

برنامه نویسی کامپیوتر

علیرضا قربی

مقدمه

شاخص TIOBE



<https://www.tiobe.com/tiobe-index/>

TIOBE شاخص

Sep 2018	Sep 2017	Change	Programming Language	Ratings	Change
1	1		Java	17.436%	+4.75%
2	2		C	15.447%	+8.06%
3	5	▲	Python	7.653%	+4.67%
4	3	▼	C++	7.394%	+1.83%
5	8	▲	Visual Basic .NET	5.308%	+3.33%
6	4	▼	C#	3.295%	-1.48%
7	6	▼	PHP	2.775%	+0.57%
8	7	▼	JavaScript	2.131%	+0.11%
9	-	▲▲	SQL	2.062%	+2.06%
10	18	▲▲	Objective-C	1.509%	+0.00%
11	12	▲	Delphi/Object Pascal	1.292%	-0.49%
12	10	▼	Ruby	1.291%	-0.64%
13	16	▲	MATLAB	1.276%	-0.35%
14	15	▲	Assembly language	1.232%	-0.41%
15	13	▼	Swift	1.223%	-0.54%
16	17	▲	Go	1.081%	-0.49%
17	9	▼▼	Perl	1.073%	-0.88%
18	11	▼▼	R	1.016%	-0.80%
19	19		PL/SQL	0.850%	-0.63%
20	14	▼▼	Visual Basic	0.682%	-1.07%

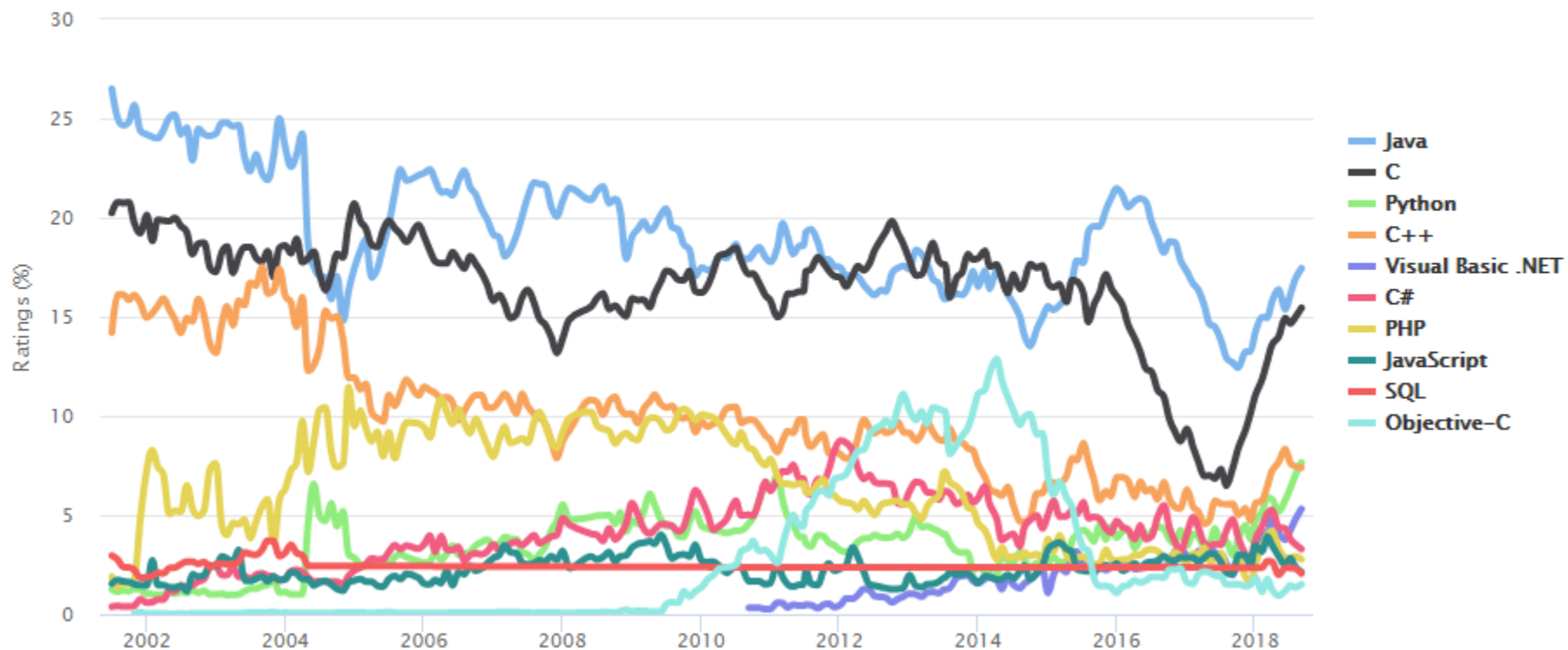
شاخص TIOBE

21	SAS	0.645%
22	Dart	0.585%
23	D	0.564%
24	Scratch	0.539%
25	F#	0.498%
26	COBOL	0.492%
27	Scala	0.471%
28	ABAP	0.433%
29	Fortran	0.415%
30	Lua	0.405%
31	Rust	0.396%
32	Transact-SQL	0.316%
33	Lisp	0.308%
34	Groovy	0.298%
35	LabVIEW	0.269%
36	Prolog	0.265%
37	Ada	0.255%
38	Logo	0.251%
39	Julia	0.242%
40	Haskell	0.218%

TIOBE شاخص

TIOBE Programming Community Index

Source: www.tiobe.com



زبان‌های با تاریخچه طولانی (شاخص TIOBE)

Programming Language	2018	2013	2008	2003	1998	1993	1988
Java	1	2	1	1	16	-	-
C	2	1	2	2	1	1	1
C++	3	4	3	3	2	2	4
Python	4	7	6	11	23	17	-
C#	5	5	7	8	-	-	-
Visual Basic .NET	6	11	-	-	-	-	-
JavaScript	7	9	8	7	20	-	-
PHP	8	6	4	5	-	-	-
Ruby	9	10	9	18	-	-	-
Delphi/Object Pascal	10	13	10	9	-	-	-
Perl	14	8	5	4	3	11	-
Objective-C	15	3	40	56	-	-	-
Ada	29	19	18	15	13	5	3
Fortran	30	25	22	12	5	3	15
Lisp	31	12	16	13	7	6	2

زبان‌های برتر هر سال (شاخص TIOBE)

Year	Winner
2017	🏆 C
2016	🏆 Go
2015	🏆 Java
2014	🏆 JavaScript
2013	🏆 Transact-SQL
2012	🏆 Objective-C
2011	🏆 Objective-C
2010	🏆 Python
2009	🏆 Go
2008	🏆 C
2007	🏆 Python
2006	🏆 Ruby
2005	🏆 Java
2004	🏆 PHP
2003	🏆 C++

گزارش Octoverse سایت Github

- 15: Objective-C

- زبانی بر پایه C برای ساخت برنامه‌های iOS

- 14: Scala

- زبانی که سال 2004 برای جایگزینی جاوا شرکت اوراکل طراحی شد و بیشتر برای برنامه‌های مقیاس بزرگ استفاده می‌شود

گزارش Octoverse سایت Github

- 13: Swift

- شرکت اپل سال 2014 زبان سوئیفت را طراحی کرد که راحت تر و سریع تر از Objective-C می باشد.

- 12: Shell

- شل دقیقا زبان برنامه نویسی نمی باشد. برای اجرای دستورات خاص در سیستم عامل از آن استفاده می شود.

گزارش Octoverse سایت Github

- 11: TypeScript

- توسط شرکت مایکروسافت در سال 2012 طراحی شد و تا حدودی نزدیک به زبان برنامه نویسی محبوب Javascript می باشد و برای برنامه های بزرگ ساخته شده است.

- 10: C

- یکی از قدیمی ترین زبان های برنامه نویسی که در اوایل 1970 ساخته شد.

گزارش Octoverse سایت Github

- 9: Go

- زبانی که توسط شرکت گوگل و برای طراحی سیستم ها در مقیاس بزرگ ساخته شده است.

- 8: C#

- این زبان توسط مایکروسافت طراحی شده و رقیبی برای جاوا می باشد.

گزارش Octoverse سایت Github

- 7: CSS (Cascading Style Sheets)

- زبانی برای طراحی وب سایت‌ها و برنامه‌های تحت وب

- 6: C++

- این زبان توسط مایکروسافت در سال 1983 طراحی شد و در سرورها و برنامه‌های دسکتاپ و ... کاربرد زیادی دارد.

گزارش Octoverse سایت Github

- 5: PHP

- بسیاری از وبسایت‌ها همانند یاهو و فیسبوک از PHP برای کدنویسی سایت خود استفاده می‌نمایند.

- 4: Ruby

- یکی از زبان‌هایی که ساده نوشته می‌شود و ساده خوانده می‌شود و شعار آن "دوست یک برنامه نویس" می‌باشد.

گزارش Octoverse سایت Github

- 3: Java

- جاوا در سال 1991 توسط شرکت Sun Microsystems برای رابط کاربری تلویزیون‌ها طراحی شد. پس از خرید این زبان توسط شرکت اوراکل، این زبان به ابرقدرتی در زبان‌های برنامه نویسی تبدیل شد. هم اکنون بیشتر برنامه‌های اندروید به این زبان نوشته می‌شوند.

- 2: Python

- در سال 1989 طراحی شد و به دلیل خوانا بودن کدنویسی آن بسیار محبوب است. بسیاری از برنامه‌نویسان بر این عقیده هستند که پایتون ساده‌ترین زبان برای یادگیری است.

گزارش Octoverse سایت Github

- 1 JavaScript

- برای طراحی صفحات مدرن وب استفاده می‌شود. بر خلاف اسم خود ارتباطی با جاوا ندارد.

مبانی اولیه

ماشین فراگیر (The Universal Machine)

یک کامپیوتر مدرن را می‌توان بدین صورت تعریف کرد :
ماشینی که اطلاعات را تحت کنترل یک برنامه قابل تغییر،
ذخیره و اداره می‌کند.

به عبارتی می‌توان اطلاعات را به کامپیوتر بدهیم و کامپیوتر
آنها را تبدیل به فرمی جدید و قابل استفاده در بیاورد و در
نهایت آنها را به عنوان خروجی برای ما نمایش دهد.

ماشین فراگیر (The Universal Machine)

کامپیوترها تنها ماشین‌های اداره و تحلیل اطلاعات نیستند به عنوان مثال :

هنگامی که از ماشین حساب برای جمع اعداد استفاده می کنید، شما در حال وارد کردن اطلاعات (اعداد) و ماشین حساب در حال اداره و تحلیل اطلاعات می باشد که در نهایت روی صفحه نمایش چاپ می شود.

ماشین فراگیر (The Universal Machine)

یک برنامه کامپیوتری مجموعه‌ای دقیق و گام به گام از فرمان‌ها است که راه را به کامپیوتر نشان می‌دهد.

اگر برنامه را تغییر دهیم، کامپیوتر عملیات دیگری را انجام می‌دهد. این انعطاف‌پذیری باعث می‌شود که کامپیوتر در لحظه‌ای ویرایشگر متن باشد و در لحظه‌ای دیگر به پخش صوت و تصویر بپردازد.

ماشین ثابت است ولی برنامه‌ای که ماشین را کنترل می‌کند تغییر می‌کند.

ماشین فراگیر (The Universal Machine)

هر کامپیوتر ماشینی است که برنامه‌ها را اجرا می‌کند. کامپیوترهای زیادی وجود دارند (PC، لپ‌تاپ، تبلت، تلفن‌های همراه و ...).

یکی از نتایج علم کامپیوتر این است که تمامی این کامپیوترها با برنامه نویسی مناسب قدرت یکسانی را دارند، به عبارتی هر کدام از آن‌ها می‌تواند تمامی کارهای دیگری را انجام می‌دهد. با این تعریف، لپ‌تاپ و یا PC که در حال حاضر از آن استفاده می‌کنید یک ماشین فراگیر و جامع است. هر کاری را که به درستی برای آن تعریف نمایید می‌تواند انجام دهد.

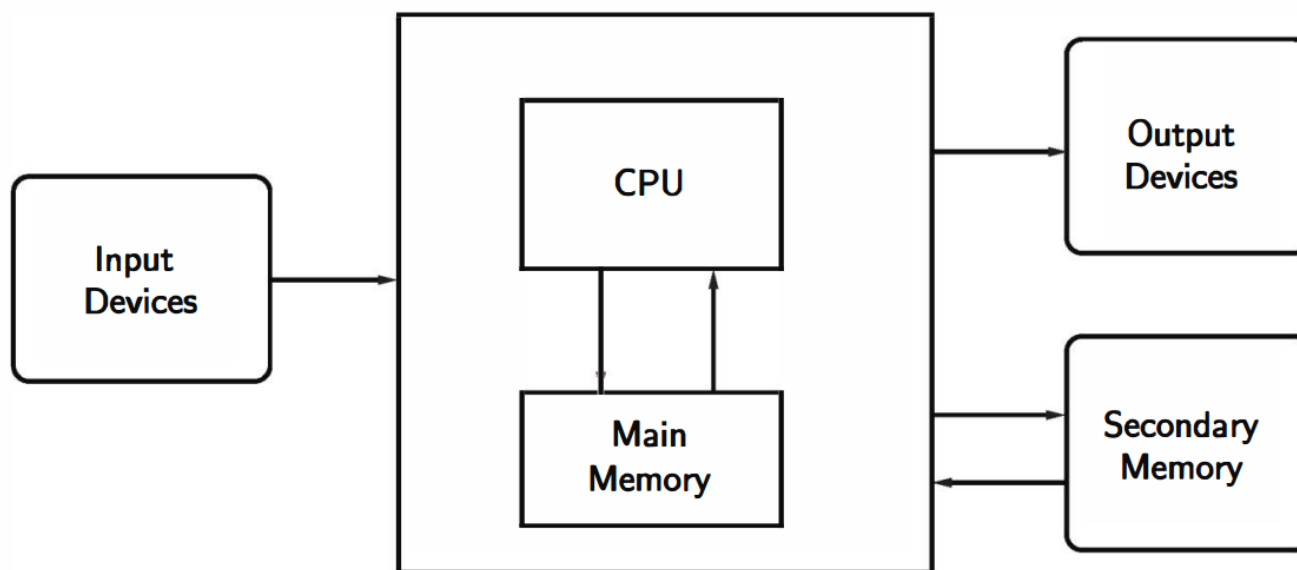
قدرت برنامه

تا به اینجا درس مهمی از محاسبات را متوجه شدید:
نرم افزار (برنامه ها) بر سخت افزار (ماشین فیزیکی) حاکمیت دارد.

نرم افزار مشخص می کند که کامپیوتر چه عملیاتی را انجام دهد.
فرآیند تولید نرم افزار را برنامه نویسی می نامند که هدف اصلی این درس می باشد.

نگاهی به سخت افزار

گرچه کامپیوترها در جزئیات با یکدیگر تفاوت دارند، اما در سطح بالا تمامی کامپیوترهای مدرن شباهت زیادی با یکدیگر دارند. شکل زیر نمایش اجرایی یک کامپیوتر مدرن است :

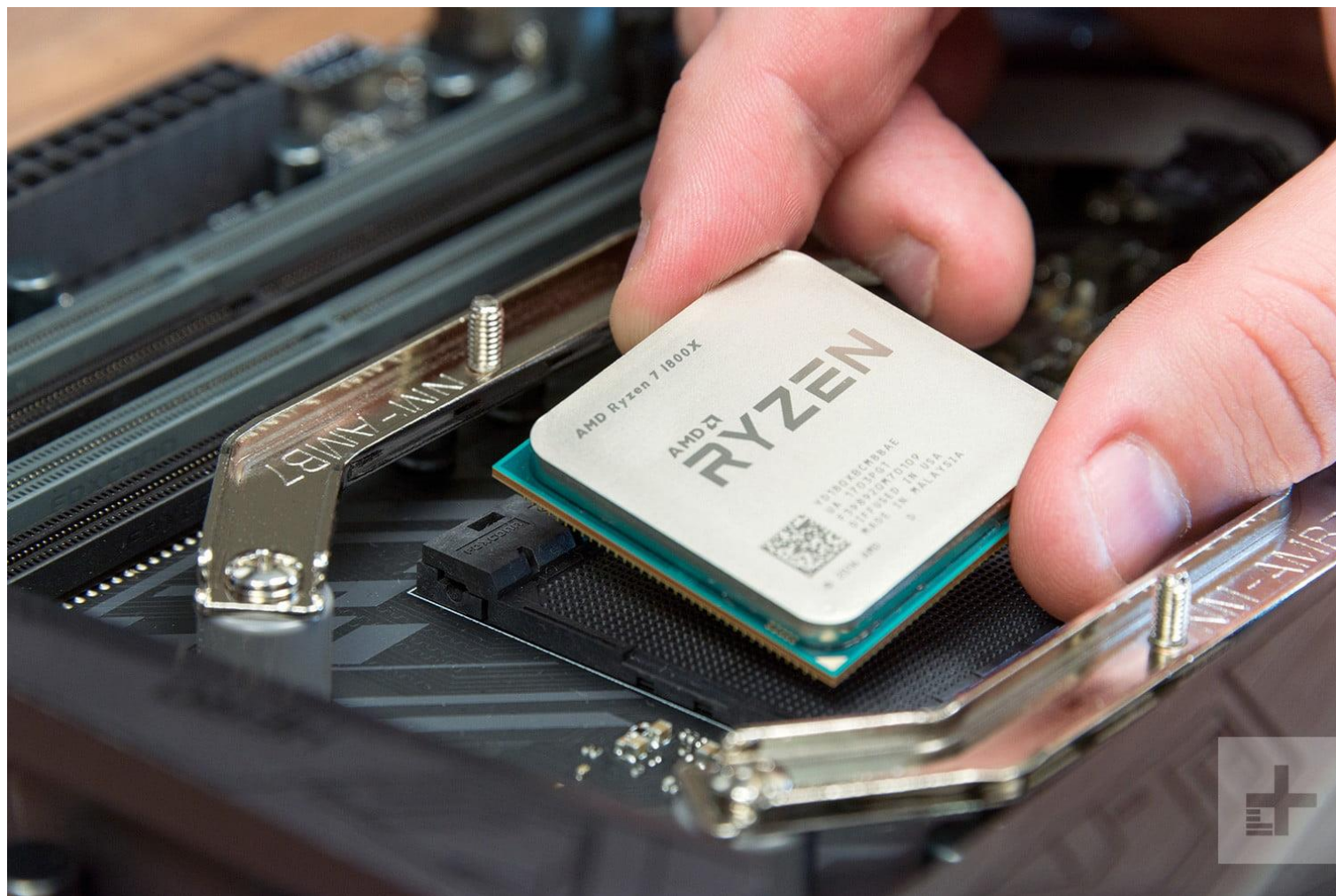


پردازنده CPU

پردازنده (Central Processing Unit) مغز یک کامپیوتر است. جایی که تمامی عملیات انجام می‌شود. پردازنده توانایی عملیات ریاضی (همانند جمع دو عدد) و عملیات منطقی (تساوی دو عدد) را دارد.



پردازنده CPU



پردازنده CPU



حافظه اصلی

حافظه برنامه و اطلاعات را در خود ذخیره می‌کند. پردازنده به طور مستقیم فقط به اطلاعات درون حافظه اصلی دسترسی دارد. به حافظه اصلی رم (Random Access Memory) RAM گفته می‌شود.

حافظه اصلی سریع ولی فرار است. بدین معنی که زمانی که کامپیوتر خاموش شود، اطلاعات درون رم از بین می‌روند. بنابراین نیاز به حافظه ثانویه که اطلاعات را در خود ذخیره کند وجود دارد.

حافظه اصلی



حافظه اصلی



حافظه ثانویه

در کامپیوترهای مدرن حافظه‌های ثانویه معمولاً به صورت دیسک سخت (HDD - Hard Disk Drive) و یا دیسک جامد (SSD - Solid State Drive) داخلی (Internal) می‌باشند. لازم به ذکر است فرم خارجی (External) آن‌ها نیز امروزه بسیار مورد استفاده قرار می‌گیرند.

HDD اطلاعات را بر روی یک دیسک چرخان ذخیره می‌کند در حالی که SSD اطلاعات را بر روی مدارهای الکترونیکی که به عنوان فلش مموری (Flash Memory) شناخته می‌شوند، ذخیره می‌کند.

HDD



SSD



حافظه ثانویه

- کامپیوترها از Removable Media (رسانه جداشدنی) به عنوان حافظه ثانویه پشتیبانی میکنند. برای مثال :
- USB Memory ها (فلش مموری USB)
 - DVD ها که اطلاعات در آن‌ها به صورت اپتیکال (Optical) توسط لیزر خوانده و نوشته می‌شود.

زبان برنامه نویسی

همانطور که گفته شد برنامه مجموعه‌ای از فرمان‌ها است که به کامپیوتر می‌گوید چکاری را انجام دهد. در نتیجه این فرمان‌ها بایستی به زبانی نوشته شود که برای کامپیوتر قابل فهم باشد.

از طرفی، طراحی برنامه‌ای برای کامپیوتر که به طور کامل زبان انسان را درک کند هنوز موضوعی حل نشده است.

حتی اگر کامپیوترها توانایی فهمیدن زبان انسان را داشتند، باز هم زبان انسان برای بیان مسائل و الگوریتم‌های پیچیده مناسب نیست. زبان انسان پر از ابهام است به طور مثال جمله زیر را در نظر بگیرید :

زبان برنامه نویسی

"مردی را در پارک با تلسکوپ دیدم."

آیا من تلسکوپ داشته‌ام یا آن مرد؟

آیا من در پارک او را دیده‌ام یا نه؟

بیشتر اوقات انسان‌ها یکدیگر را با اطلاعات قبلی، تجارب قبل و لحن و تن صدا و یا حرکات دست درک می‌کنند.

زبان برنامه نویسی

در نتیجه محققان علم کامپیوتر این مشکل را با طراحی نشانه‌گذاری‌ها و قراردادهایی برای بیان محاسبات به صورت دقیق و صریح حل کردند.

این نشانه‌گذاری‌ها و قراردادها به اسم زبان برنامه‌نویسی شناخته می‌شوند.

هر ساختاری در زبان برنامه نویسی یک فرم دقیق (دستور زبان - Syntax) و یک مفهوم دقیق (معنی - Semantic) دارد.

زبان برنامه نویسی

زبان برنامه نویسی همانند نوشتن یک Code (مجموعه قانون) می باشد که کامپیوتر از آن پیروی می کند. به همین دلیل غالباً برنامه نویسان به برنامه های خود کد کامپیوتری می گویند و فرآیند نوشتن یک الگوریتم در یک زبان برنامه نویسی را کدنویسی می نامند.

Python, C, C++, C#, Java, Fortran و هزاران زبان دیگر از جمله زبان های برنامه نویسی می باشند. گرچه این زبان ها با یکدیگر تفاوت های زیادی در جزئیات دارند ولی دستورات (Syntax) و معانی (Semantic) آن ها صریح و بدون ابهام است.

زبان برنامه نویسی

تمامی زبان‌های ذکر شده از جمله زبان‌های کامپیوتری **سطح بالا** می‌باشند. با اینکه دقیق هستند، به گونه‌ای طراحی شده‌اند که توسط انسان قابل فهم باشند..

سخت افزار کامپیوتر تنها توانایی فهم زبان **سطح پایین** که به عنوان زبان ماشین ساخته می‌شود را دارد.

به عنوان مثال اگر از کامپیوتر بخواهیم دو عدد را جمع کند، دستوراتی که پردازنده انجام می‌دهد می‌تواند به صورت زیر باشد :

زبان برنامه نویسی

"load the number from memory location 3501 into CPU"

"load the number from memory location 3502 into CPU"

"add the two numbers in the CPU"

"store the result into location 3503"

عملیات طولانی است، حتی از چیزی که در بالا نوشته شده هم پیچیده تر و طولانی تر است زیرا که تمامی دستورات و اعداد بایستی به صورت باینری (صفر و یک) ارائه شوند.

زبان برنامه نویسی

در یک زبان سطح بالا نظیر Python جمع دو عدد به صورت طبیعی تری قابل بیان است : $c = a + b$

این نوشتار بسیار ساده تر و قابل فهم تر است ولی بایستی راهی برای تبدیل زبان سطح بالا به زبان ماشین پیدا کنیم.

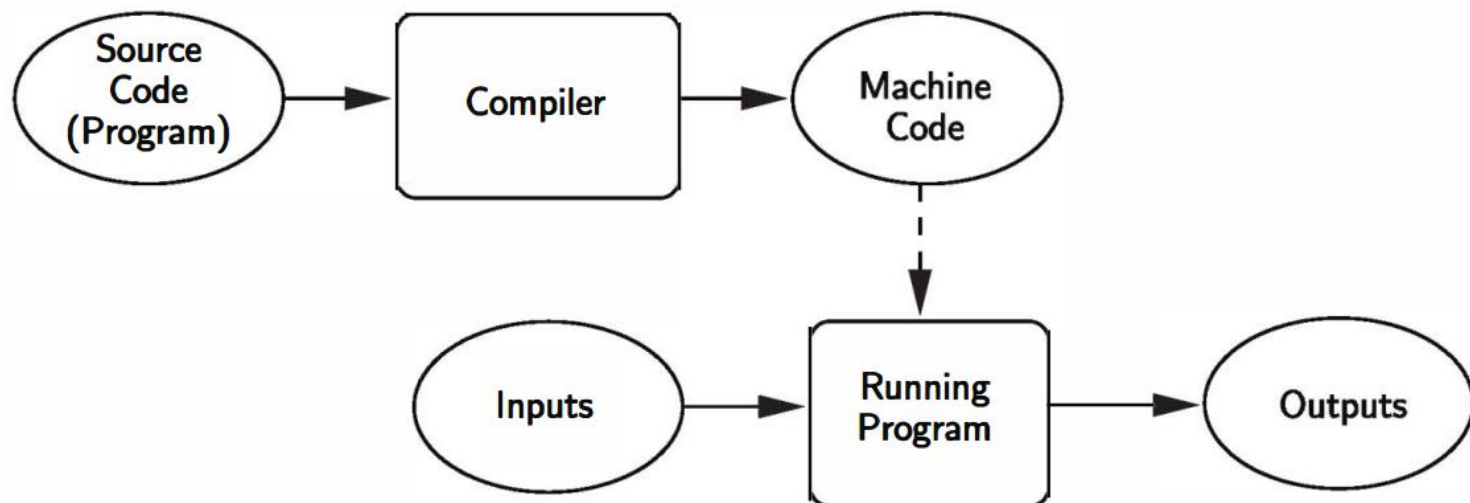
دو روش برای اینکار وجود دارد :

- استفاده از کامپایلر - مؤلف (Compiler)
- استفاده از مفسر (Interpreter)

کامپایلر - مؤلف

کامپایلر یک برنامه پیچیده کامپیوتری است که یک برنامه نوشته شده به زبان سطح بالا را ترجمه و تبدیل به برنامه‌ای معادل به زبان ماشین می‌کند.

فرآیند تالیف در دیاگرام زیر نشان داده شده است :



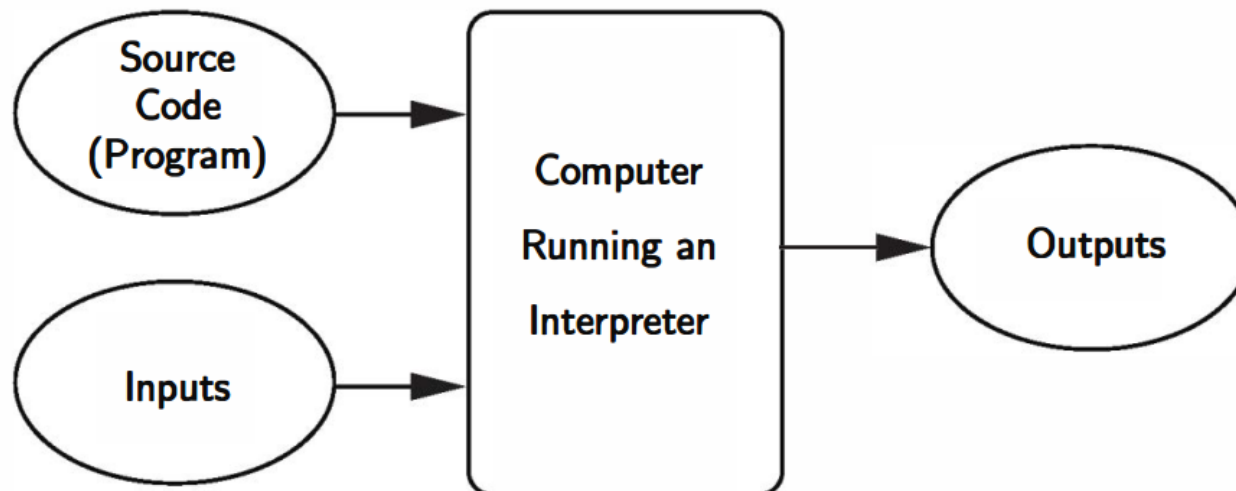
کامپایلر - مؤلف

در اینجا، برنامه سطح بالا Source Code (سورس کد) و برنامه معادل Machine Code نامیده می‌شود.

Machine Code مستقیماً قابل اجرا برای کامپیوتر است. خط تیره دیاگرام نمایانگر اجرای Machine Code است.

مفسر

مفسر کامپیوتری را شبیه سازی می‌کند که زبان سطح بالا را درک می‌کند. به جای ترجمه و تبدیل برنامه سطح بالا به برنامه معادل ماشین، مفسر برنامه سطح بالا را در صورت نیاز تحلیل و دستورات را فرمان به فرمان اجرا می‌کند. این فرآیند در دیاگرام زیر نشان داده شده است :



تفاوت مؤلف و مفسر

- تالیف (Compile) کردن ترجمه یکباری است، زمانی که برنامه تالیف شود می‌توان آن را بدون نیاز به کامپایلر و یا سورس کد، پی در پی اجرا کرد.
 - در حالت مفسر، سورس کد و مفسر برای هر اجرا نیاز می‌باشند.
- در نتیجه برنامه‌های Compile شده با توجه به ترجمه یکباره سریع تر می‌باشند، اما برنامه‌های تفسیر شده انعطاف پذیرتر می‌باشند زیرا برنامه ها به صورت متقابل نوشته و اجرا می‌شوند.

مؤلف و مفسر

فرآیند ترجمه یکی دیگر از برتری‌های زبان سطح بالا را نسبت به زبان ماشین نشان می‌دهد.

زبان ماشین توسط طراحان یک پردازنده خاص ساخته شده است، هر کامپیوتری زبان ماشین خود را دارد. به عنوان مثال یک برنامه برای پردازنده Intel i7 روی لپتاپ شما، مستقیماً بر روی یک گوشی هوشمند با پردازنده ARMv8 اجرا نمی‌شود.

از طرفی یک برنامه نوشته شده به زبان سطح بالا تا زمانی که یک Compiler و یا Interpreter مناسب برای آن تهیه شود، قابلیت اجرا بر روی بسیاری از کامپیوترها (لپتاپ، تبلت، گوشی هوشمند و ...) را خواهد داشت.

چگونگی حل مسئله

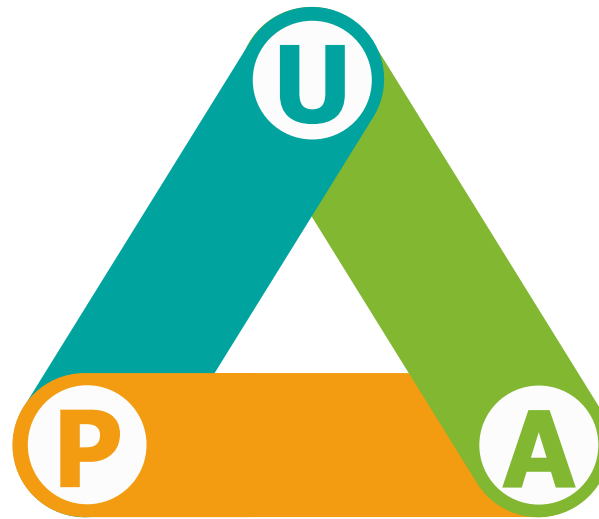
برای حل مسئله نمی‌توان یک راه کلی و عمومی ارائه داد. به عبارتی برای حل هر مسئله باید از شیوه‌های خاصی استفاده کرد که انتخاب شیوه‌های مناسب برای حل مسئله بستگی به ابتکار، خلاقیت و تجربه شخص دارد.

به طور کلی سه عامل زیر را می‌توان در انتخاب روش مناسب اتخاذ کرد :

- شناخت دقیق مسئله
- طرح نقشه حل مسئله
- آنالیز (تحلیل) کردن مسئله

چگونگی حل مسئله

شناخت دقیق مسئله
Understanding



طرح نقشه حل مسئله
Planning

آنالیز کردن مسئله
Analyzing

شناخت مسئله

در برخورد با یک مسئله باید عوامل زیر را مورد بررسی قرار داد :

□ **داده‌ها :** در یک مسئله منظور از داده‌ها، فرض‌ها و نتایجی است که باید از داده‌ها حاصل شود.

□ **مجهول‌ها :** منظور از مجهول‌ها مقادیری است که مسئله در جستجوی آن می‌باشد.

□ **ارتباط بین داده‌ها و مجهول‌ها :** منظور از ارتباط، رابطه منطقی است که توسط آن می‌توان از داده‌ها به مجهولات دست یافت که در این میان از ابزارهای ریاضی می‌توان استفاده نمود.

شناخت مسئله

مثال) فرض کنید می‌خواهیم میانگین دو عدد ۱۰ و ۵۰ را محاسبه نماییم.

❑ داده‌ها : دو عدد ۱۰ و ۵۰

❑ مجهول : میانگین دو عدد ۱۰ و ۵۰

❑ ارتباط بین داده‌ها و مجهول : فرمول میانگین دو عدد یعنی مجموع آن‌ها تقسیم بر ۲

طرح نقشه حل مسئله

پس از شناختن مسئله باید برای حل آن (بدست آوردن مجهولات) نقشه‌ای طرح کرد. به طور کلی انسان برای حل مسائل به دو صورت **منطقی و غیر منطقی (الگوریتمی و غیر الگوریتمی)** عمل می‌کند.

در روش غیر منطقی برای حل مسائل از تفکر جانبی استفاده می‌شود. تفکر جانبی نمونه‌ای تفکر است که از مجموعه راه حل‌های موجود، ساده‌ترین روش را با شیوه‌ای نامتعارف برگزیند.

در روش منطقی از شیوه الگوریتمی استفاده می‌شود.

تحلیل راه حل مسئله

منظور از تحلیل راه حل مسئله عموماً بررسی و تجزیه راه حل و در نهایت تعمیم دادن آن است.

به دلیل پیچیدگی مسئله و یا حجیم بودن و ... شناخت و پیدا کردن راه حل مناسب به سادگی امکان پذیر نیست. توصیه می شود مسئله را به شکل زیر به قسمت‌های کوچک تجزیه کنید :

- در بالاترین سطح یا سطح اول، صورت مسئله قرار دارد.
- در سطح میانی یا سطح دوم، مسئله را به چند زیر مسئله تجزیه کنید.
- در سطح آخر یا سطح سوم هر کدام از زیر مسئله ها را تک تک بررسی و در صورت پیچیده بودن به زیر مسئله‌های ساده‌تر تجزیه کنید و این عمل را آنقدر ادامه دهید تا دیگر نیازی به تقسیم کردن نباشد.
- هر کدام از زیر مسئله ها را حل کرده و با به هم پیوستن راه حل‌ها، مسئله اصلی را حل کنید.

الگوریتم

الگوریتم (Algorithm)

تعریف کلی : به مجموعه‌ای از یک یا چند دستورالعمل که اجرای آن‌ها به ترتیب تعیین شده منجر به انجام حل مسئله می‌گردد، الگوریتم گفته می‌شود.

تعریف دقیق : به مجموعه‌ای از دستورالعمل‌ها که مراحل مختلف کاری (حل یک مسئله) را به **زبان دقیق** و با **جزئیات کافی** بیان کرده و در آن **ترتیب مراحل** و **خاتمه‌پذیر بودن** عملیات کاملاً مشخص باشد، الگوریتم گفته می‌شود.

الگوریتم (Algorithm)

تعریف ماشینی : الگوریتم یک ماشین ساده است که قابلیت‌های زیر را دارد :

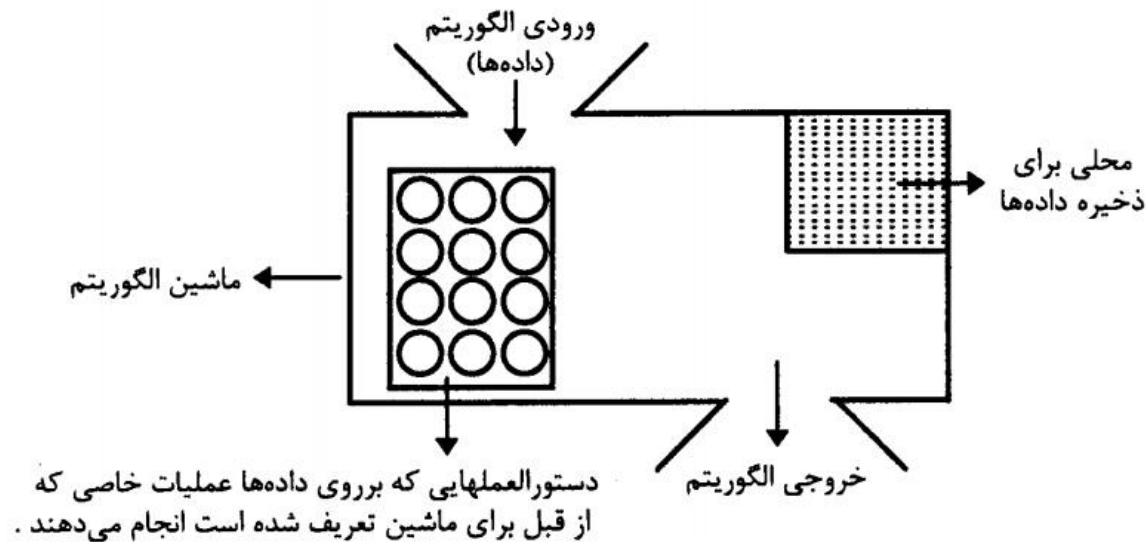
- ✓ دریافت یک یا چند داده به عنوان ورودی
- ✓ ارسال یک یا چند مقدار به عنوان خروجی
- ✓ انجام عملیات مقایسه بین دو داده دلخواه
- ✓ ذخیره داده‌ها و اطلاعات در قسمتی از ماشین

الگوریتم (Algorithm)

توجه : بین دو داده عددی همواره یکی از سه حالت زیر برقرار است:

$$\left\{ \begin{array}{ll} a < b \text{ یا } a > b \text{ یا } a = b & \text{برای دو داده} \\ a = 0 \text{ یا } a > 0 \text{ یا } a < 0 & \text{برای یک داده} \end{array} \right.$$

که با توجه به تعریف فوق می‌توانیم شکل زیر را برای ماشین الگوریتم در نظر بگیریم :



الگوریتم (Algorithm)

برای فهم دقیق یک مسئله باید از قواعدی استفاده شود، یعنی برای مسئله مدل ریاضی خاصی ایجاد و آن را در قالب یک فرمول ریاضی بیان کنید.

در علم الگوریتم نویسی نکات زیر را رعایت کنید :

- در ابتدای هر الگوریتم کلمه شروع و در انتهای آن کلمه پایان قرا دهید.
- حتما برای هر یک از دستورالعمل‌ها، شماره‌ای در نظر بگیرید.
- برای محاسبه یک عبارت ریاضی یا انجام عملیات روی چند داده، حتما مکانی برای ذخیره داده‌ها و نتایج حاصله در نظر بگیرید.
- برای قراردادن مقداری در یک مکان از حافظه ماشین فرضی از علامت ← استفاده کنید.

الگوریتم (Algorithm)

مثال : مفهوم هر یک از عبارات زیر را برای ماشین الگوریتم بنویسید :

$$1) A \leftarrow 2$$

این عبارت بدین مفهوم است که خانه ای را به اسم A در نظر بگیرید و مقدار عددی ۲ را در آن ذخیره کند.

$$2) B \leftarrow 2*3-5$$

این عبارت بدین معنی است که ابتدا ۲ را در ۳ ضرب کن و سپس ۵ را از حاصل آن کم کرده و در خانه B قرار بده.

$$3) C \leftarrow B+1$$

این عبارت به این معناست که به مقدار خانه B یک واحد اضافه کن و حاصل را در خانه ای به اسم C ذخیره کن .

$$4) I \leftarrow I+1$$

این عبارت بدین مفهوم است که به مقدار قبلی خانه I یک واحد اضافه کن و سپس مقدار نهایی را دوباره در خانه I ذخیره کن .

الگوریتم (Algorithm)

مثال : مفهوم هر یک از عبارات زیر را برای ماشین الگوریتم بنویسید :

$$1) A \leftarrow 2$$

این عبارت بدین مفهوم است که خانه ای را به اسم A در نظر بگیرید و مقدار عددی ۲ را در آن ذخیره کند.

$$2) B \leftarrow 2*3-5$$

این عبارت بدین معنی است که ابتدا ۲ را در ۳ ضرب کن و سپس ۵ را از حاصل آن کم کرده و در خانه B قرار بده.

$$3) C \leftarrow B+1$$

این عبارت به این معناست که به مقدار خانه B یک واحد اضافه کن و حاصل را در خانه ای به اسم C ذخیره کن .

$$4) I \leftarrow I+1$$

این عبارت بدین مفهوم است که به مقدار قبلی خانه I یک واحد اضافه کن و سپس مقدار نهایی را دوباره در خانه I ذخیره کن .

الگوریتم (Algorithm)

مثال ۱) الگوریتمی بنویسید که سه عدد ۲ و ۳ و ۵ را در سه خانه A و B و C ذخیره کند و سپس میانگین آن‌ها را محاسبه و چاپ کند.

الگوریتم (Algorithm)

شناخت مسئله : داده‌های مسئله ۲ و ۳ و ۵ هستند و نتیجه حاصل از آن‌ها میانگین سه عدد می باشد.

طرح نقشه مسئله : فرمول محاسبه میانگین سه عدد :

$$\text{میانگین سه عدد} = \frac{\text{مجموع سه عدد}}{۳}$$

الگوریتم (Algorithm)

الگوریتم مسئله :

1. شروع
2. عدد ۲ را در خانه A قرار بده.
3. عدد ۳ را در خانه B قرار بده.
4. عدد ۵ را در خانه C قرار بده.
5. خانه‌های A و B و C را با هم جمع کن و حاصل را در خانه S قرار بده.
6. مقدار خانه S را بر ۳ تقسیم کن و در AVE قرار بده.
7. مقدار خانه AVE را بنویس.
8. پایان

الگوریتم (Algorithm)

عموماً برای ساده تر شدن الگوریتم و قابل فهم تر بودن آن از فرم نوشتاری در ریاضیات برای دستورالعملها، بجای فرم فارسی استفاده می‌کنیم. به عنوان نمونه اگر الگوریتم مثال ۱ را بخواهیم بفرم ریاضی بنویسیم، داریم:

۱- شروع

۲- $A \leftarrow 2$

۳- $B \leftarrow 3$

۴- $C \leftarrow 5$

۵- $S \leftarrow C+B+A$

۶- $AVE \leftarrow \frac{S}{3}$

۷- AVE را بنویس.

۸- پایان .

الگوریتم (Algorithm)

اگر بخواهیم راه حل این مسئله را تعمیم دهیم به طوریکه برای سه عدد دلخواه عمل میانگین را انجام دهد ان را به شکل زیر می نویسیم:

مثال (۲)

۱- شروع .

۲- سه عدد را به عنوان ورودی بگیر

۳- حاصل جمع سه عدد را در S قرار بده

۴- S را بر ۳ تقسیم کن و در AVE قرار بده

۵- AVE را بنویس

۶- پایان .

۱- شروع

۲- A و B و C را بگیر .

۳- $S \leftarrow A+B+C$

۴- $AVE \leftarrow \frac{S}{3}$

۵- AVE را بنویس.

۶- پایان.

فرم ساده شده



الگوریتم (Algorithm)

برای اینکه همواره مراحل مختلف اجرای یک الگوریتم را بهتر درک کنید، آنرا بطریقه دستی اجرا کنید. برای این منظور به هریک از متغیرها مقداری داده و برای خروجی نیز محلی را در نظر گرفته و با توجه به روند الگوریتم آنرا اجرا کنید. اجرای دستی الگوریتم مثال ۱ بصورت زیر است:

$\frac{A}{2}$	$\frac{B}{3}$	$\frac{C}{5}$	$\frac{S}{10}$	$\frac{AVE}{10}$	$\frac{\text{چاپ}}{10}$
				$\frac{3}{3}$	$\frac{3}{3}$

برای مثال ۲ نیز:

$\frac{A}{4}$	$\frac{B}{6}$	$\frac{C}{2}$	$\frac{S}{12}$	$\frac{AVE}{12}$	$\frac{\text{چاپ}}{4}$
				$\frac{3}{3}$	$\frac{4}{4}$

دلیل اجرای دستی الگوریتم بررسی صحت درستی آن می باشد. در ادامه، الگوریتمها را بفرم خلاصه شده نوشته و سعی می کنیم اسامی متغیرها را طوری انتخاب کنیم که تناسبی با راه حل مسئله و مقداری که قرار است در آن ذخیره شود، داشته باشند.

الگوریتم (Algorithm)

توجه : تساوی در ریاضیات به مفهوم برابر بودن دو مقدار است اما در کامپیوتر به مفهوم جایگزینی است به همین دلیل از علامت \leftarrow برای جایگزینی استفاده می کنیم. نکته دیگری که حائز اهمیت است، اسامی خانه ها یا متغیرهایی است که داده ها یا نتایج محاسبات در آنها ذخیره می شوند. این اسامی باید ترکیبی از حروف و ارقام بوده و حتماً اولین کاراکتر آن یکی از حروف الفباء باشد و از کاراکترهای خاص مثل (+, -, ×, ÷, ?, !, و ... برای نامگذاری استفاده نکنید.

مثلاً اگر بخواهیم مقدار ۵ را در خانه I ذخیره کنیم حتماً باید بنویسید :

$$I \leftarrow 5$$

چراکه اگر برعکس آن یعنی $5 \leftarrow I$ را بنویسید بدین مفهوم است که مقدار I را در خانه ای بنام 5 قرار بده که نادرست است. بدلیل اینکه 5 نام آن خانه است که با توجه به قواعد گفته شده نادرست می باشد.

و یا $D \leftarrow b^2 - 4ac$ که درست بوده ولی برعکس آن یعنی $D \leftarrow b^2 - 4ac$ نادرست می باشد چرا که $b^2 - 4ac$ نام خانه می باشد که با توجه به قواعد گفته شده نادرست می باشد.

الگوریتم (Algorithm)

مثال ۳) حقوق کارگری A ریال است و هر ماه ۱۰٪ از حقوق وی بابت بیمه و ۵٪ آن بابت حق مسکن کسر می‌گردد. الگوریتمی بنویسید که حقوق یک کارگر را به‌عنوان ورودی دریافت کرده و حقوق خالص وی را با کسر بیمه و حق مسکن محاسبه و چاپ نماید.

$$\left. \begin{array}{l} B \text{ مقدار بیمه} \\ M \text{ مقدار حق مسکن} \\ H \text{ مقدار حقوق خالص} \end{array} \right\}$$

۱- شروع

۲- A را بگیر.

$$B \leftarrow \frac{10A}{100} \quad -3$$

$$M \leftarrow \frac{5A}{100} \quad -4$$

$$S \leftarrow M+B \quad -5$$

$$H \leftarrow A-S \quad -6$$

۷- H را بنویس.

اجرای آن :

A	B	M	S	H	چاپ
10000	1000	500	1500	8500	8500

۸- پایان. ۸۵۰۰ ریال حقوق خالص است.

الگوریتم (Algorithm)

مثال ۴) الگوریتمی بنویسید که زمان T بر حسب ثانیه را به عنوان ورودی دریافت نموده و معین کند که چند ساعت، چند دقیقه و چند ثانیه است. با فرض اینکه هر ساعت ۶۰ دقیقه و هر دقیقه ۶۰ ثانیه است. حل :

برای بدست آوردن ساعت ابتدا T را بر ۳۶۰۰ تقسیم کرده و جزء صحیح آنرا بدست می آوریم که معرف ساعت است، سپس این مقدار را در ۳۶۰۰ ضرب کرده از T کم کرده و حاصل را بر ۶۰ تقسیم و جزء صحیح آنرا بدست می آوریم که معرف دقیقه است و این عمل را برای ثانیه نیز انجام می دهیم. با توجه به توضیحات بالا الگوریتم مسئله بصورت زیر می باشد.

۱- شروع .

۲- T را بگیر .

$$H \leftarrow \left[\frac{T}{3600} \right] \quad ۳-$$

۴- H را بنویس .

H : ساعت
 M : دقیقه
 S : ثانیه

الگوریتم (Algorithm)

$$R \leftarrow T - 3600 \times H \quad \text{۵-}$$

$$M \leftarrow \left[\frac{R}{60} \right] \quad \text{۶-}$$

۷- M را بنویس.

$$S \leftarrow R - 60 \times M \quad \text{۸-}$$

۹- S را بنویس.

۱۰- پایان.

اجرای دستی آن :

$$\begin{array}{r} T \\ \hline 4210 \end{array} \quad \begin{array}{r} H \\ \hline 1 \end{array} \quad \begin{array}{r} R \\ \hline 610 \end{array} \quad \begin{array}{r} M \\ \hline 10 \end{array} \quad \begin{array}{r} S \\ \hline 10 \end{array}$$

$$\begin{array}{r} \text{چاپ} \\ \hline 1 \text{ ساعت} \\ 10 \text{ دقیقه} \\ 10 \text{ ثانیه} \end{array}$$

الگوریتم (Algorithm)

انواع جملات

در الگوریتم نویسی جملات به چهار نوع تقسیم می‌شوند :

۱- جملات شرطی

۲- جملات محاسباتی

۳- جملات توضیحی

۴- جملات مربوط به ورودی و خروجی (I/O) Input/Output

جملات شرطی

این جملات به دو دسته تقسیم می‌شوند :

شرطی نوع ساده : فرم کلی این جملات به صورت زیر است :

< یک یا چند دستور > THEN < یک یا چند شرط > IF

الگوریتم (Algorithm)

در این گونه جملات شرطی، اگر شرط بعد از IF درست باشد دستورات مقابل THEN را اجرا و به خط بعد منتقل می‌شود. اما اگر شرط نادرست باشد دستورات جلوی THEN را انجام نداده و مستقیماً به خط بعد می‌رود.

تذکر: دستورات الگوریتم به ترتیب نوشتن آنها اجرا می‌شوند ولی ما می‌توانیم ترتیب اجرای دستورات را از خطی به خط دیگر انتقال دهیم. در دستورات شرطی اگر شرط ما درست باشد دستورات مقابل THEN اجرا شده و در این حالت می‌توانیم اجرای الگوریتم را به چند خط بالاتر و یا چند خط پایین‌تر ارجاع دهیم.

الگوریتم (Algorithm)

مثال ۵) الگوریتمی بنویسید که اعداد زوج دو رقمی را یکی یکی محاسبه و چاپ نماید.

چاپ	I
10	10
12	12
14	14
.	.
.	.
.	.
98	98
100	100

۱- شروع

۲- $I \leftarrow 10$

۳- I را بنویس

۴- $I \leftarrow I + 2$

۵- اگر $I \leq 98$ سپس برو به خط ۳

۶- پایان .

دراین مثال ملاحظه می شود که مقدار نهائی I برابر 100 می باشد.

الگوریتم (Algorithm)

مثال ۶) الگوریتمی بنویسید که عدد طبیعی n را به عنوان ورودی دریافت و اعداد فرد کوچکتر یا مساوی عدد n را یکی یکی محاسبه و چاپ نماید.

n	I	چاپ
8	1	1
	3	3
	5	5
	7	7
	9	

۱- شروع

۲- n را بگیر

۳- $I \leftarrow 1$

۴- I را بنویس

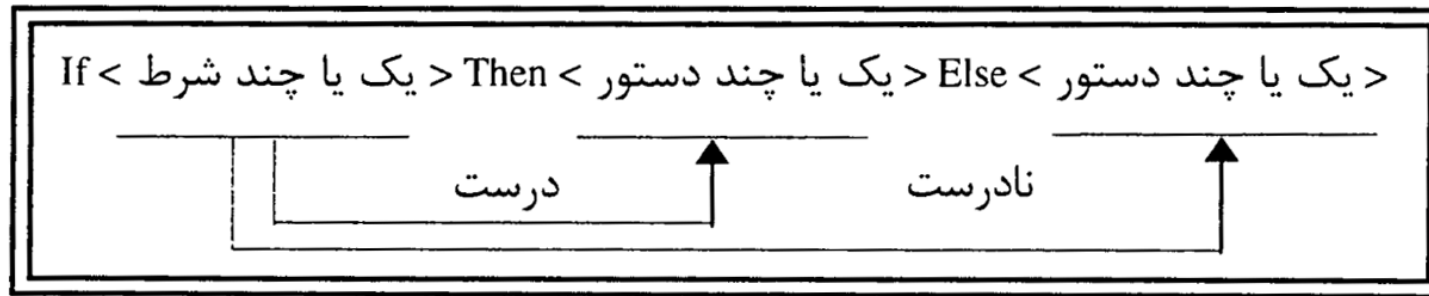
۵- $I \leftarrow I + 2$

۶- اگر $I \leq n$ سپس برو به خط ۴

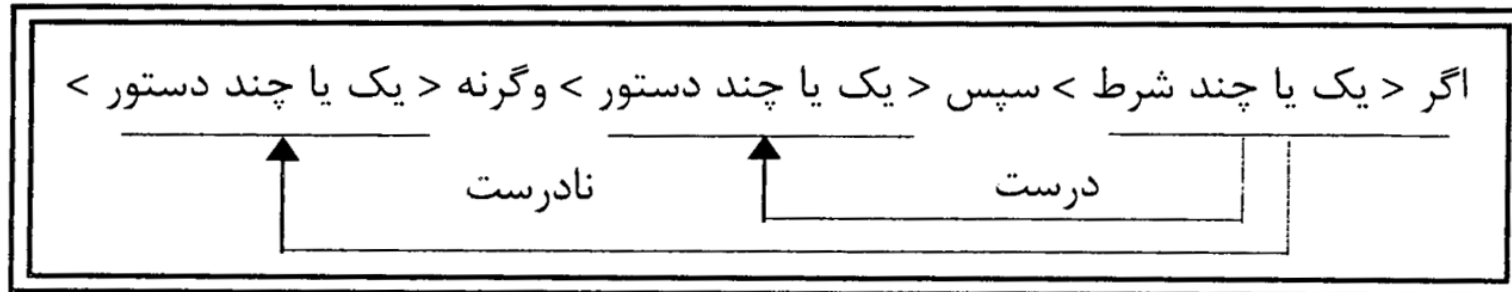
۷- پایان.

الگوریتم (Algorithm)

شرطی نوع دوم: فرم کلی این جملات بصورت زیر است:



یا



تذکر: هرگاه بخواهیم دو دستورالعمل را در یک شماره بنویسیم بین دو دستورالعمل (و) قرار می دهیم.

الگوریتم (Algorithm)

مثال ۷) الگوریتمی بنویسید که اعداد زوج بین 1000 و 2000 را یکی یکی تولید و چاپ نماید. در ضمن مجموع آنها را نیز چاپ کند.

<u>I</u>	<u>S</u>	<u>چاپ</u>
1000	0	1000
1002	1000	
1004	3006	1002
1006	.	1004
1008	.	.
1010	.	.
1012	.	.
1014	750000	2000
2002		750000

۱- شروع

۲- $I \leftarrow 1000$ و $S \leftarrow 0$

۳- بنویس I و $S \leftarrow S + I$

۴- $I \leftarrow I + 2$

۵- اگر $I \leq 2000$ سپس برو به ۳ وگرنه S را بنویس

۶- پایان

تذکر: هر گاه بخواهیم مجموعی را محاسبه کنیم، ابتدا متغیری را در نظر می‌گیریم که مقدار اولیه آن صفر باشد (مانند S)، سپس تک تک جملاتی را که قرار است در خانه S ذخیره شوند را تولید و با مقدار قبلی S جمع و دوباره در خود S ذخیره می‌کنیم و این مراحل را تا پایان تولید جملاتی که قرار است تولید شوند ادامه می‌دهیم.

الگوریتم (Algorithm)

مثال ۸) الگوریتمی بنویسید که عدد طبیعی N را دریافت و مجموع زیر را محاسبه و چاپ نماید.

چاپ	S	I	N	
شروع	0	1	7	$S = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{N}$
۲- N را بگیر	1	2		
۳- $S \leftarrow 0$ و $I \leftarrow 1$	1.5	3		
۴- $S \leftarrow S + \frac{1}{I}$	1.84	4		
۵- $I \leftarrow I + 1$	2.09	5		
۶- اگر $I \leq N$ سپس برو به خط ۴ وگرنه S را بنویس	2.29	6		
۷- پایان		7		

الگوریتم (Algorithm)

مثال ۹) الگوریتمی بنویسید که عدد طبیعی n را دریافت و مجموع زیر را محاسبه و چاپ نماید.

$$S = \frac{1}{2} + \frac{1}{4} + \dots + \frac{1}{n}$$

$\frac{n}{8}$	$\frac{I}{2}$	$\frac{S}{0}$	چاپ 1.045
	2	0	
	4	0.5	
	6	0.75	
	8	0.94	
10		1.045	

۱- شروع

۲- n را بگیر

۳- $S \leftarrow 0$ و $I \leftarrow 2$

۴- $S \leftarrow S + \frac{1}{I}$

۵- $I \leftarrow I + 2$

۶- اگر $I \leq n$ سپس برو به خط ۴ وگرنه S را بنویس

۷- پایان

الگوریتم (Algorithm)

مثال ۱۰) الگوریتمی بنویسید که عدد طبیعی n را دریافت و مجموع زیر را محاسبه و چاپ نماید.

$$P = \frac{1}{3^1} + \frac{2}{3^2} + \frac{3}{3^2} + \dots + \frac{n}{3^n}$$

$\frac{n}{4}$	$\frac{I}{1}$	$\frac{S}{0}$	چاپ
	1	0	0.7160
	2	0.34	
	3	0.56	
	4	0.67	
5		0.7160	

۱- شروع

۲- n را بگیر

۳- $I \leftarrow 1$ و $P \leftarrow 0$

۴- $p \leftarrow p + \frac{I}{3^I}$

۵- $I \leftarrow I + 1$

۶- اگر $I \leq n$ سپس برو به خط ۴ وگرنه p را بنویس

۷- پایان

الگوریتم (Algorithm)

مثال (۱۱) الگوریتمی بنویسید که ۳ عدد a ، b و c را دریافت و معین کند که با این سه عدد می‌توان یک مثلث ساخت یا خیر؟

فرض: شرط آنکه سه عدد a و b و c طول اضلاع مثلثی باشند، آنستکه بین a و b و c روابط زیر برقرار باشند.

$$\left. \begin{array}{l} a \leq b + c \\ b \leq a + c \\ c \leq a + b \end{array} \right\}$$

۱- شروع

۲- a و b و c را بگیر

۳- اگر $a \leq b + c$ بود برو به خط ۴ و گرنه برو به ۷

۴- اگر $b \leq a + c$ بود برو به خط ۵ و گرنه برو به ۷

۵- اگر $c \leq a + b$ بود برو به خط ۶ و گرنه برو به ۷

۶- بنویس می‌توان یک مثلث ساخت و برو به ۸

۷- بنویس نمی‌توان یک مثلث ساخت

۸- پایان

الگوریتم (Algorithm)

مثال ۱۲) الگوریتمی بنویسید که سه عدد a ، b و c که طول اضلاع مثلث هستند، را دریافت و معین کند که مثلث قائم الزاویه است یا خیر؟

یکی از شرایط قائم الزاویه بودن اینست که برای اضلاع a و b و c یکی از سه شرط زیر برقرار باشد.

$$\left\{ \begin{array}{l} a^2 = b^2 + c^2 \\ \text{یا} \\ b^2 = a^2 + c^2 \\ \text{یا} \\ c^2 = a^2 + b^2 \end{array} \right.$$

۱- شروع

۲- a و b و c را بگیر

۳- اگر $a^2 = b^2 + c^2$ بود برو به خط ۶ وگرنه برو به خط ۴

۴- اگر $b^2 = a^2 + c^2$ بود برو به خط ۶ وگرنه برو به خط ۵

۵- اگر $c^2 = a^2 + b^2$ بود برو به خط ۶ وگرنه برو به خط ۷

۶- بنویس می‌توان یک مثلث قائم الزاویه ساخت و برو به ۸

۷- بنویس نمی‌توان یک مثلث قائم الزاویه ساخت

۸- پایان

اگر این قسمت نوشته نشود
نیز تاثیری در نتیجه ندارد.

(یا می‌نویسیم پایان)

الگوریتم (Algorithm)

مثال ۱۳) الگوریتمی بنویسید که ۱۰۰ عدد دلخواه را یکی یکی دریافت و چاپ نماید و در نهایت جمع آن اعداد را نیز چاپ کند.

۱- شروع

۲- $I \leftarrow 1$ و $S \leftarrow 0$

۳- a را بگیر

۴- a را بنویس

۵- $S \leftarrow S + a$

۶- $I \leftarrow I + 1$

۷- اگر $I \leq 100$ بود پس برو به خط ۳

۸- S را بنویس

۹- پایان .

الگوریتم (Algorithm)

تذکر : در کلیه الگوریتمهایی که روند اجرا را از مرحله ای به مرحله ای دیگر ارجاع داده ایم، دراصل یک حلقه ساخته ایم. نکته ای که در مورد این حلقه ها قابل توجه می باشد، این است که همواره متغیری در ابتدای الگوریتم وجود دارد که یک مقدار اولیه دارد سپس در حین اجرا به مقدار آن افزوده شده و دوباره درخودش ذخیره می شود (البته این عمل در داخل حلقه انجام می شود) و با یک مقدار مشخص سنجیده می شود. اگر شرط درست بود، دوباره حلقه تکرار می گردد که به این گونه متغیر ها **شمارنده** نیز گفته می شود.

باید دقت داشت زمانیکه از حلقه خارج می شویم ، همواره مقدار شمارنده حلقه از مقداری که با آن سنجیده شده است، بیشتر است . (اگر شمارنده حالت نزولی داشته باشد مقدار شمارنده از مقداری که با آن سنجیده می شود ، کمتر است) .

در مورد مثال ۱۳، متغیر I نقش شمارنده حلقه را دارد که مقدار اولیه آن یک است و هر بار در داخل حلقه یک واحد به آن اضافه شده و با عدد ۱۰۰ مقایسه می گردد، زمانیکه از حلقه خارج می شویم، مقدار شمارنده I همواره از عدد ۱۰۰ بزرگتر است یعنی ۱۰۱ می باشد .

الگوریتم (Algorithm)

درباره هر مسئله دلخواهی این مقدار بفرم زیر محاسبه می شود :

میزان تغییرات در داخل حلقه + مقدار نهایی آن در داخل حلقه = مقدار نهایی شمارنده در خارج از حلقه

مثال (۱۴) در الگوریتم زیر معین کنید ، شمارنده حلقه چه متغیری بوده و مقدار نهایی آن در داخل حلقه و خارج حلقه چقدر است ؟

۱- شروع

۲- $N \leftarrow 2$

۳- i و z و k را بگیر

۴- $t \leftarrow i + z + k$

۵- t را بنویس

۶- $N \leftarrow N + 2$

۷- اگر $N \leq 6$ سپس برو به ۳

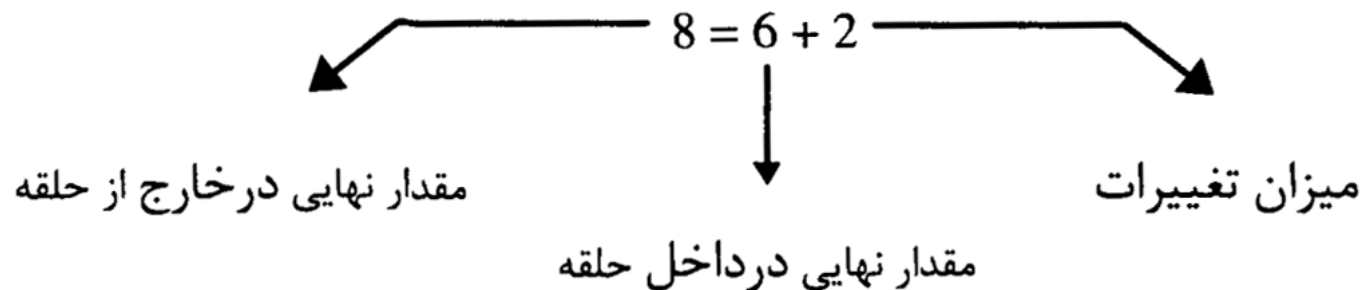
۸- پایان

الگوریتم (Algorithm)

<u>N</u>	<u>i</u>	<u>j</u>	<u>k</u>	<u>t</u>	<u>چاپ</u>
2	1	2	3	6	6
4	5	7	1	13	13
6	8	0	-1	7	7
8					

اجرای دستی آن بصورت زیر است .

در این مثال N شمارنده حلقه است که سه بار تکرار می شود و مقدار نهائی آن در داخل حلقه با توجه به جدول فوق، برابر ۶ و خارج از حلقه برابر ۸ است .



الگوریتم (Algorithm)

مثال ۱۵) الگوریتمی بنویسید که دو عدد A و B را به عنوان ورودی دریافت و بزرگترین و کوچکترین عدد را محاسبه و چاپ نماید. (اگر A و B مساوی بودند ، دو عدد دیگر را بگیرد).
این مسئله را در دو حالت حل می کنیم.
حالت اول : در این قسمت فرض می کنیم که هیچوقت دو عدد مساوی نیستند که الگوریتم آن بصورت زیر می شود.

۱- شروع

۲- A و B را بگیر

۳- اگر $A > B$ بود سپس بنویس A بزرگترین عدد و B کوچکترین عدد است و پایان .

۴- B را به عنوان بزرگترین عدد بنویس .

۵- A را به عنوان کوچکترین عدد بنویس .

۶- پایان.

A	B	چاپ
3	5	۵ بزرگترین و ۳ کوچکترین عدد

اجرای دستی آن بصورت زیر است .

الگوریتم (Algorithm)

حالت دوم: این قسمت تعمیم یافته حالت اول است، با این فرض که قبل از مقایسه کوچکتري یا بزرگتری A و B، حالت تساوی آنها مقایسه شود و در صورت برابری، دو عدد دیگر دریافت شود، که الگوریتم آن بصورت زیر است.

۱- شروع

۲- A و B را بگیر

۳- اگر $A=B$ سپس برو به خط ۲

(برو به خط ۶)

۴- اگر $A>B$ بود سپس A را بنویس بزرگترین و B را بنویس کوچکتري عدد و پایان

۵- B را بنویس بزرگترین عدد

۶- A را بنویس کوچکتري عدد

۷- پایان .

اجرای دستی آن بصورت زیر است:

A	B	چاپ
6	6	3 بزرگترین و -1
3	-1	کوچکتري عدد

الگوریتم (Algorithm)

مثال ۱۶) الگوریتم مثال ۱۵ را برای سه عدد بنویسید به طوریکه فقط بزرگترین را چاپ کند. (فرض بر این است که سه عدد با هم برابر نیستند).

۱- شروع

۲- A و B و C را بگیر

۳- اگر $A > B$ و $A > C$ سپس A را بنویس و پایان

۴- اگر $A > B$ و $C > A$ سپس C را بنویس و پایان

۵- اگر $B > A$ و $B > C$ سپس B را بنویس و پایان

۶- اگر $B > A$ و $C > B$ سپس C را بنویس و پایان

که اجرای دستی آن بصورت زیر خواهد بود :

۷- پایان

← به جای پایان می‌توان نوشت برو به خط 7

<u>A</u>	<u>B</u>	<u>C</u>	<u>چاپ</u>
-5	1	-3	1

<u>A</u>	<u>B</u>	<u>C</u>	<u>چاپ</u>
3	2	6	6

الگوریتم (Algorithm)

مثال ۱۷) الگوریتمی بنویسید که یک عدد صحیح مثبت را به عنوان ورودی دریافت و معین کند عدد زوج یا فرد می باشد.

توضیح: اگر عددی بر ۲ تقسیم شود، باقیمانده آن یکی از دو عدد صفر یا یک می باشد. اگر صفر بود عدد زوج است و اگر یک بود عدد فرد است.

۱- شروع

۲- عدد N را بگیر

(برو به خط ۶)

۳- $R \leftarrow N - 2 \times [N/2]$

۴- اگر $R=0$ سپس بنویس N زوج است و پایان

۵- بنویس N فرد است

۶- پایان .

الگوریتم (Algorithm)

که می‌توان آنرا به‌فرم زیر با دو شرط نوشت :

۱- شروع

۲- عدد N را بگیر

۳- $R \leftarrow N - 2 \times [N/2]$

۴- اگر $R=0$ سپس بنویس N زوج است و پایان

۵- اگر $R=1$ سپس بنویس N فرد است

۶- پایان .

که اجرای دستی آن به شکل مقابل خواهد بود :

<u>چاپ</u>	<u>R</u>	<u>N</u>	<u>چاپ</u>	<u>R</u>	<u>N</u>
۱۳ فرد است	1	13	۶ زوج است	0	6

الگوریتم (Algorithm)

مثال ۱۸) الگوریتمی بنویسید که عدد طبیعی N را دریافت و مجموعه مقسوم علیه‌های آن و همچنین ، تعداد آنها و نیز مجموع آنها را محاسبه و چاپ نماید .

توضیح : فرض کنیم N عددی طبیعی باشد، مقسوم علیه‌های N ، اعداد صحیح کوچکتر از N هستند که اگر N به هر کدام از آنها تقسیم شود باقیمانده مساوی با صفر می‌شود. برای مثال عدد ۱۲ را در نظر بگیریم که مقسوم علیه‌های آن ۱، ۲، ۳، ۴، ۶ و ۱۲ هستند.

۱- شروع

۲- N را بگیر

۳- $I \leftarrow 1$

۴- $R \leftarrow N - I \times [N/I]$

۵- اگر $R=0$ سپس I را بنویس

۶- $I \leftarrow I + 1$

۷- اگر $I \leq N$ سپس برو به خط ۴

۸- پایان

الگوریتم (Algorithm)

الگوریتم بالا فقط مقسوم علیه‌ها را حساب می‌کند برای محاسبه مجموع آنها خانه‌ای مانند S را مساوی صفر در نظر گرفته و هر بار که یک مقسوم علیه را تولید کرده و می‌نویسیم آنرا با S جمع می‌کنیم و برای تعداد W، ای را مساوی صفر قرارداده و هر بار که یک مقسوم علیه را می‌نویسیم یکی به W اضافه می‌کنیم و در آخر S و W را چاپ می‌کنیم.

الگوریتم کامل مسئله بصورت زیر است :

چاپ	W	S	R	I	N
	0	0	0	1	6
1	1	1			
2	2	3	0	2	
3	3	6	0	3	
			2	4	
			1	5	
6	4	10	0	6	
10				7	
4					

۱- شروع

۲- N را بگیر

۳- $S \leftarrow 0$

۴- $W \leftarrow 0$

۵- $I \leftarrow 1$

۶- $R \leftarrow N - I \times [N/I]$

۷- اگر $R=0$ سپس I را بنویس و $W \leftarrow W+1$ و $S \leftarrow S+I$

۸- $I \leftarrow I+1$

الگوریتم (Algorithm)

۹- اگر $I \leq N$ سپس برو به خط ۶

۱۰- S و W را بنویس

۱۱- پایان .

الگوریتم (Algorithm)

در قسمت مربوط به چگونگی حل مسئله ذکر شد که بهترین راه حل برای یک مسئله راه حلی است که ساده و قابل فهم و در عین حال کوتاه نیز باشد، در مورد الگوریتم نیز باید بگوئیم که یک الگوریتم زمانی بهترین حالت را دارد که :

- ساده و قابل فهم باشد .
- روان باشد .
- کوتاه باشد .

همچنین از عملیاتی که وقت زیادی گرفته و در عین حال زائد هستند، استفاده نشده باشد. در مورد الگوریتم مثال ۱۸ می‌دانیم که برای هر عددی یک و خود عدد، مقسوم علیه‌های آن هستند، پس عمل محاسبه مقسوم علیه برای ۱ و N اضافه و زائد است یعنی می‌توان از ۲ شروع کرد و تا $N-1$ پیش رفت، حتی قویتر از آن می‌توان گفت که کفایت از ۲ تا نصف عدد پیش برویم چرا که قضیه زیر بیان این موضوع است که از نصف عدد تا خود عدد (منظور کوچکتر از خود عدد) هیچ مقسوم علیه‌ای برای عدد وجود ندارد.

الگوریتم (Algorithm)

پس الگوریتم مثال ۱۸ را می‌توانیم در حالت بهینه بصورت زیر بنویسیم :

۱- شروع

۲- N را بگیر

۳- ۱ را بنویس

۴- $S \leftarrow N+1$ و $W \leftarrow 2$

۵- $I \leftarrow 2$

۶- $R \leftarrow N - I \times [N/I]$

۷- اگر $R=0$ سپس I را بنویس و $S \leftarrow S+I$ و $W \leftarrow W+1$

۸- $I \leftarrow I+1$

۹- اگر $I \leq \frac{N}{2}$ سپس برو به ۶

۱۰- N را بنویس

۱۱- S و W را بنویس

۱۲- پایان.

N	I	R	S	W	چاپ
12	2	0	18	3	1
	3	0	18	4	
	4	2	32	5	
	5	0			
	6				
	7		28	6	

تعداد 6 مجموع 28

در اینجا فرض شده که N عدد 1 نباشد.

چون یک و خود عدد را برای محاسبه مقسوم علیه در نظر نمی‌گیریم

لذا از ابتدا مقدار S را $N+1$ و W را ۲ در نظر گرفتیم.

الگوریتم (Algorithm)

مثال ۱۹) الگوریتمی بنویسید که عدد طبیعی N را دریافت و معین کند این عدد تام است یا نه ؟
توضیح : عدد طبیعی N را **تام** یا **کامل** می گوئیم هرگاه مجموع مقسوم علیه های کوچکتر از عدد N ، با خود عدد N برابر شود مانند:

$\{1, 2, 3, 6\} =$ مجموع مقسوم علیه های عدد 6

$\{1, 2, 3\} =$ مجموعه مقسوم علیه های کوچکتر از 6 عدد 6

که مجموع آنها $1+2+3=6$ است که با خود عدد برابر است لذا عدد 6 تام است.

الگوریتم (Algorithm)

۱- شروع

۲- N را بگیر

۳- $S \leftarrow 0$

۴- $I \leftarrow 1$

۵- $R \leftarrow N - I \times \left[\frac{N}{I} \right]$

۶- اگر $R=0$ پس $S \leftarrow S+I$

۷- $I \leftarrow I+1$

۸- اگر $I \leq \frac{N}{2}$ پس برو به ۵

۹- اگر $S = N$ سپس بنویس عدد N تام است و پایان

۱۰- بنویس عدد N تام نیست.

۱۱- پایان

<u>N</u>	<u>I</u>	<u>R</u>	<u>S</u>	<u>چاپ</u>
14	1	0	0	عدد 14
	2	0	1	تام نیست
	3	2	3	
	4	2	0	
	5	2		
	6	2		
	7	0	10	
	8			

الگوریتم (Algorithm)

مثال ۲۰) الگوریتمی بنویسید که عدد طبیعی N را به عنوان ورودی گرفته و معین کند اول است یا نه ؟
تعریف عدد اول : عدد طبیعی N را اول گوئیم هرگاه به جز یک و خودش هیچ مقسوم-
علیه دیگری نداشته باشد.

۱- شروع

۲- N را بگیر

۳- $I \leftarrow 2$

۴- $R \leftarrow N - I \times \left[\frac{N}{I} \right]$

۵- اگر $R = 0$ سپس بنویس

N اول نیست و پایان

۶- $I \leftarrow I + 1$

۷- اگر $I < N$ سپس برو به ۴

۸- بنویس عدد N اول است .

۹- پایان.

چاپ	R	I	N	چاپ	R	I	N
عدد ۱۱	1	2	11	۶ عدد	0	2	6
اول است	2	3		اول			
	.	4		نیست			
	.	.					
	.	.					
	.	.					
	1	10					
		11					

این الگوریتم فقط برای محاسبه اول بودن یا نبودن اعداد بزرگتر از 2 درست عمل می کند. از طرفی بهینه نیز نمی باشد. الگوریتم کامل تری در مبحث فلوجارت ارائه می شود.

الگوریتم (Algorithm)

مثال (۲۱) الگوریتمی بنویسید که عدد طبیعی N را دریافت و فاکتوریل آنرا محاسبه کند.
فرض کنید که N یک عدد طبیعی باشد، فاکتوریل N را با نماد $N!$ نمایش داده و مقدار آن طبق روابط زیر بدست می‌آید.

$$N! = \begin{cases} 1 & \text{اگر } N = 0 \\ 1 & \text{اگر } N = 1 \\ N \times (N - 1) \times (N - 2) \times \dots \times (3) \times (2) \times (1) & \text{اگر } N \neq 0, 1 \end{cases}$$

برای مثال :

- 1) $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$
- 2) $1! = 1$
- 3) $0! = 1$

الگوریتم (Algorithm)

			۱- شروع
$\frac{N}{6}$	$\frac{P}{1}$	$\frac{I}{1}$	۲- N را بگیر
	$1 \times 1 = 1$		۳- $P \leftarrow 1$
	$1 \times 2 = 2$	۲	۴- $I \leftarrow 1$
	$1 \times 2 \times 3 = 6$	۳	۵- $P \leftarrow P \times I$
	$1 \times 2 \times 3 \times 4 = 24$	۴	۶- $I \leftarrow I + 1$
	$1 \times 2 \times 3 \times 4 \times 5 = 120$	۵	۷- اگر $I \leq N$ سپس برو به ۵
	$1 \times 2 \times 3 \times 4 \times 5 \times 6 = 720$	۶	۸- P را بنویس
		۷	۹- پایان .

الگوریتم (Algorithm)

تذکر: در کلیه مثالهای قبل به نوعی از حلقه استفاده شده است. با توجه به نوع مسئله می توان از چند حلقه استفاده کرد بفرمی که هر حلقه به طور کامل داخل حلقه دیگری قرار گرفته باشد.

مثال ۲۲) الگوریتمی بنویسید که عدد طبیعی N را دریافت و حاصل جمع زیر را محاسبه و چاپ نماید.

$$S = 1! + 2! + 3! + \dots + N!$$

برای حل، حلقه ای با یک شمارنده برای جمع کردن و حلقه دیگری برای محاسبه فاکتوریل تک تک جملاتی که قرار است تولید و با هم جمع شوند، در نظر بگیرید.

الگوریتم (Algorithm)

چاپ	I	P	t	S	N
	1	1	1	0	3
	2	1		1	
	1	1	2		
	2	1			
	3	2		3	
	1	1	3		
	2	1			
	3	2			
	4	6		9	
			4		
9					

۱- شروع

۲- N را بگیر

۳- $S \leftarrow 0$

۴- $t \leftarrow 1$

۵- $P \leftarrow 1$

۶- $I \leftarrow 1$

۷- $P \leftarrow P \times I$

۸- $I \leftarrow I + 1$

۹- اگر $t \leq I$ سپس برو به ۷

۱۰- $S \leftarrow S + P$

۱۱- $t \leftarrow t + 1$

۱۲- اگر $t \leq N$ سپس برو به ۵

۱۳- S را بنویس

۱۴- پایان .

این فرم حلقه ها را حلقه های تودرتو می گویند که در حل مسائل متنوع، کاربرد دارد

فلوچارت

فلوچارت

در قسمت قبل با چگونگی حل مسئله و نوشتن الگوریتم برای آن آشنا شدیم. برای بیان الگوریتم ها از جملات فارسی استفاده شد و سپس آن ها را خلاصه و از عبارات ریاضی برای بیان جملات استفاده کردیم.

این روش ها برای الگوریتم هایی که ساده هستند و تعداد دستورالعمل هایشان کم است بسیار کارآمد و مناسب است. ولی برای مسائل پیچیده که از چند قسمت تشکیل شده اند و هر قسمت دارای تعداد زیادی دستورالعمل می باشد، مناسب نیستند. همچنین اگر از حلقه های تودرتو و پیچیده استفاده شده باشد، دیگر درک الگوریتم بسیار مشکل خواهد بود.

فلوچارت

در این قسمت از روش دیگری که تصویری است برای بیان الگوریتم استفاده می کنیم که به آن فلوچارت یا نمودار گردش یا نمودار عملیاتی می گویند.

همان گونه که در الگوریتم از چند نوع جمله برای بیان عملکرد استفاده کردیم، در فلوچارت نیز از نمادهای تصویری خاصی استفاده می کنیم (که به آنها مؤلفه های تصویری می گویند) که از به هم پیوستن آنها یک نمودار یا چارت خاصی به وجود می آید.

فلوچارت

تعریف فلوچارت

برای درک بهتر الگوریتم و سهولت در دنبال کردن دستورالعملهای آن از یکسری اشکال خاص برای نشان دادن الگوریتم استفاده می‌کنیم که به آن فلوچارت می‌گویند. یا به عبارت ساده‌تر :

به مجموعه‌ای از علائم ساده که الگوریتم را بصورت نمادهای تصویری یا نموداری تبدیل می‌کنند ، فلوچارت گفته می‌شود.

توضیح : اولین مرحله در حل یک مسئله با کامپیوتر، نوشتن الگوریتم و مرحله دوم رسم فلوچارت می‌باشد (رسم فلوچارت درک روش حل را ساده‌تر می‌کند)، مرحله سوم نوشتن برنامه به یکی از زبانهای برنامه نویسی است. لازم بذکر است که اگر بتوانید برای یک مسئله الگوریتم نوشته و فلوچارت مربوط به آن را رسم کنید، نوشتن برنامه آن از روی فلوچارت بسیار ساده خواهد بود، لذا با دقت هر یک از فلوچارتهایی را که در این فصل بیان می‌شوند بررسی کنید سپس آنها را بدرستی درک نمایید و در نهایت سعی کنید حداقل یک بار از روی آنها بنویسید .

علائم فلوچارت

علامت شروع و پایان

در ابتدای هر فلوچارت از علامت بیضی به مفهوم شروع و خاتمه عملیات استفاده می‌کنیم.

پایان

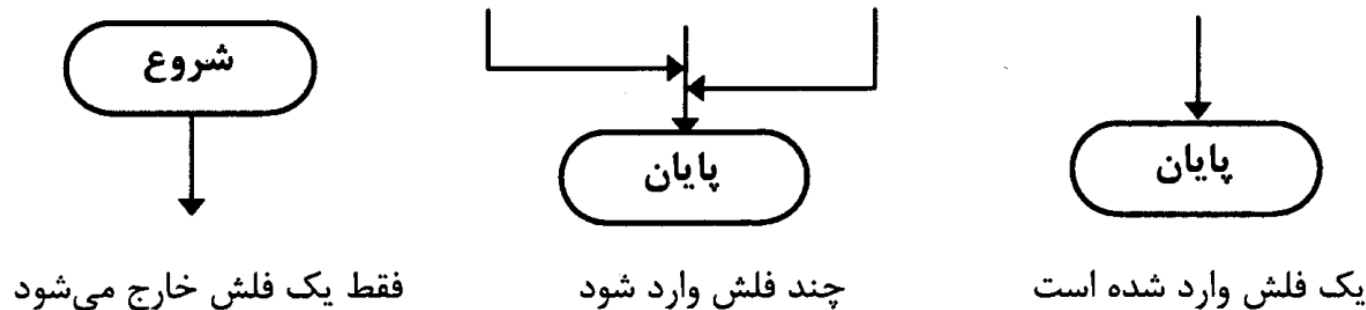
شروع

نکته‌ای قابل توجه این است که چون هر الگوریتمی فقط دارای یک شروع و پایان است لذا فلوچارت مربوط به آن نیز فقط دارای یک شروع و پایان خواهد بود.

علائم فلوچارت

برای اتصال دادن مؤلفه‌های تصویری در ترسیم فلوچارت از علامت \rightarrow استفاده کنید (با توجه به این که چگونه آنها را به هم متصل کنیم جهت و اندازه آن اختیاری خواهد بود).

از توضیحات بالا در می‌یابیم که از بیضی مربوط به شروع فقط یک فلش به بیرون منشعب شده ولی به بیضی مربوط به پایان می‌تواند چند فلش از جاهای مختلف منتهی شود که باز متذکر می‌شویم که این به مفهوم داشتن چند پایان نیست .



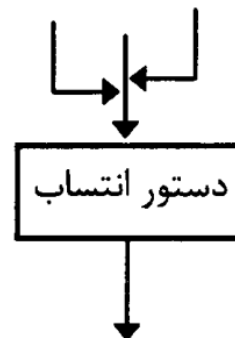
علائم فلوچارت

علامت جایگزینی و انتساب

از علامت مستطیل برای نمایش یک عمل، بخصوص عملیات محاسبه‌ای یا جایگزینی مقداری در یک خانه و کلاً عبارتهای انتسابی استفاده می‌کنیم.

مانند: $A \leftarrow 2$ $S \leftarrow S-1$ $C \leftarrow 2 \times 3$ دستور محاسبه و انتساب

در مورد این علامت نیز می‌توانید چند فلش به آن وارد ولی فقط یک فلش از آن خارج کنید.



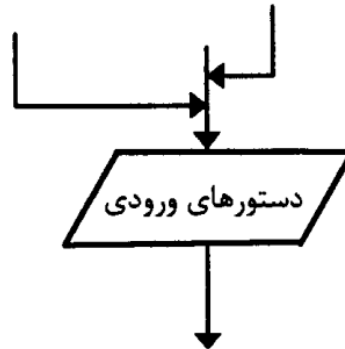
علائم فلوچارت

علامت ورودی

از علامت متوازی الاضلاع برای نمایش عملیات مربوط به گرفتن مقادیر و داده‌ها به عنوان ورودی استفاده کنید.



در مورد این علامت نیز می‌توانید چند فلش به آن منتهی ولی فقط یک فلش از آن خارج کنید.



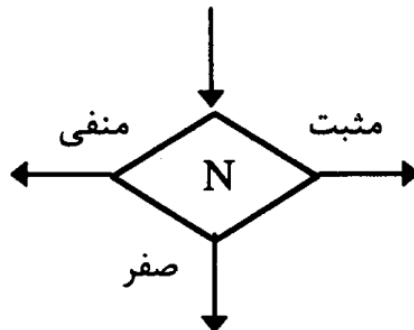
علائم فلوچارت

علامت شرطی

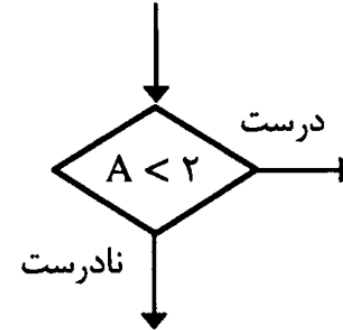
می‌دانید که در عبارات شرطی، انجام عملیات منوط به برقرار بودن شرط یا شرط‌هایی می‌باشد، به همین منظور از علامت لوزی برای دستورات شرطی استفاده کنید. به دستورات شرطی عبارتهای تصمیم‌گیری نیز گفته می‌شود. شرط یا شرط‌ها را داخل لوزی بنویسید.



در این علامت برخلاف علامتهای قبلی می‌توان چند فلش را وارد کرده و دو یا سه فلش از آن با توجه به ارزش شرط (ها) (یعنی درست یا نادرست بودن) خارج کرد. مانند:



سه فلش به عنوان خروجی

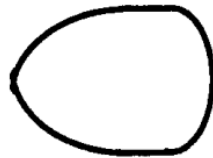


دو فلش به عنوان خروجی

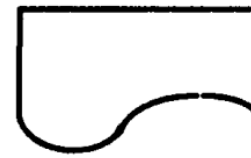
علائم فلوچارت

علامت چاپ

اگر بخواهید مقداری را روی صفحه کاغذ یا صفحه نمایش، نشان دهید از علائم زیر استفاده کنید و مقادیری که می‌خواهید چاپ شوند را داخل آن بنویسید.

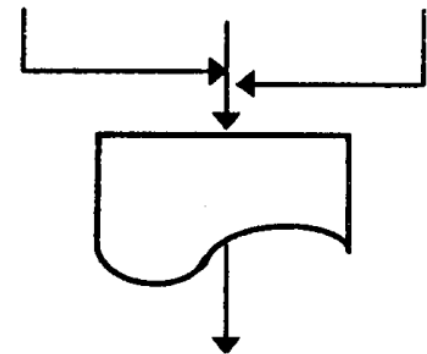
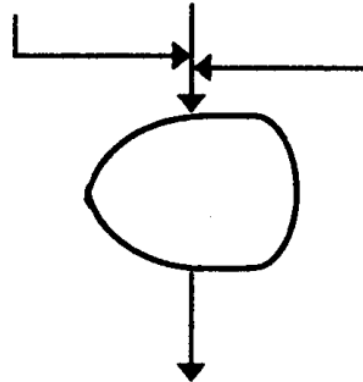
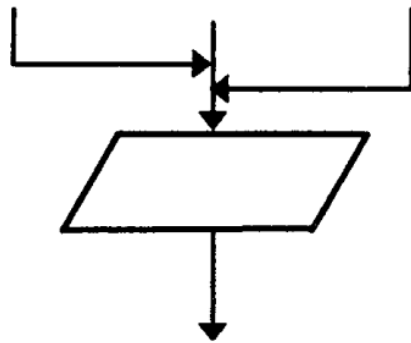


برای چاپ روی صفحه نمایش

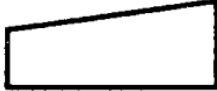


برای چاپ روی صفحه کاغذ

اگر هدف فقط نمایش دادن اطلاعات باشد، می‌توانید از همان علامت متوازی الاضلاع استفاده کنید. در این علامتها مانند بیضی و مستطیل فقط یک فلش از آنها خارج ولی چند فلش به آنها وارد کرد.

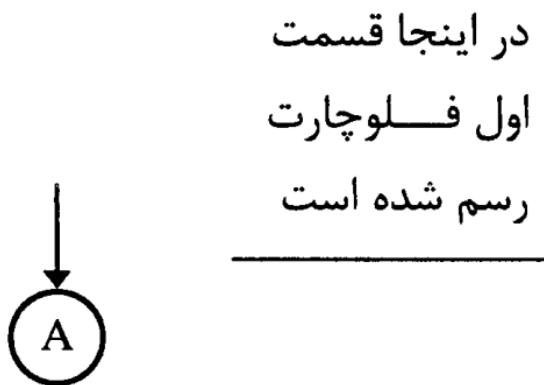


علائم فلوچارت

توضیح: در بعضی از مسائل گفته می‌شود که مقداری را از صفحه کلید، به عنوان ورودی دریافت نماید که به جای استفاده از علامت متوازی الاضلاع از علامت  استفاده کنید.

علامت ادامه

اگر فلوچارت را از یک قسمت قطع کرده و بقیه آن را در محل دیگری بنویسید، برای اتصال دادن این قسمت‌ها به هم از علامت زیر استفاده کنید.



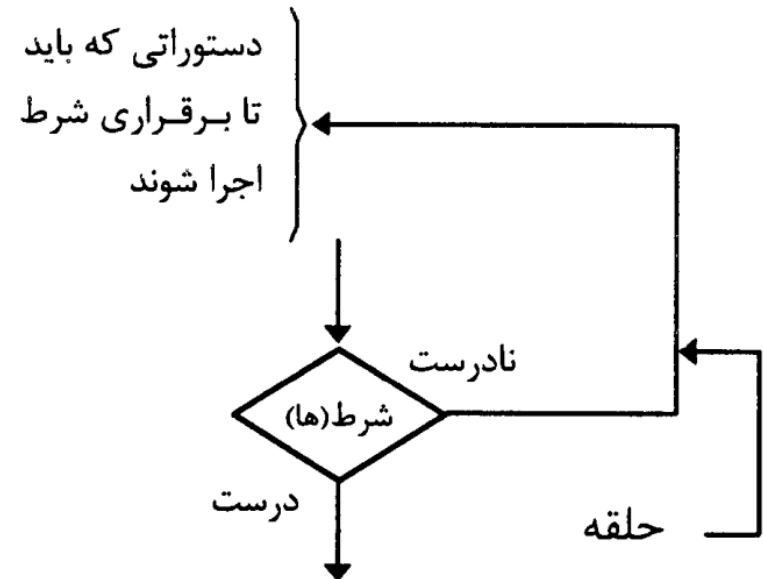
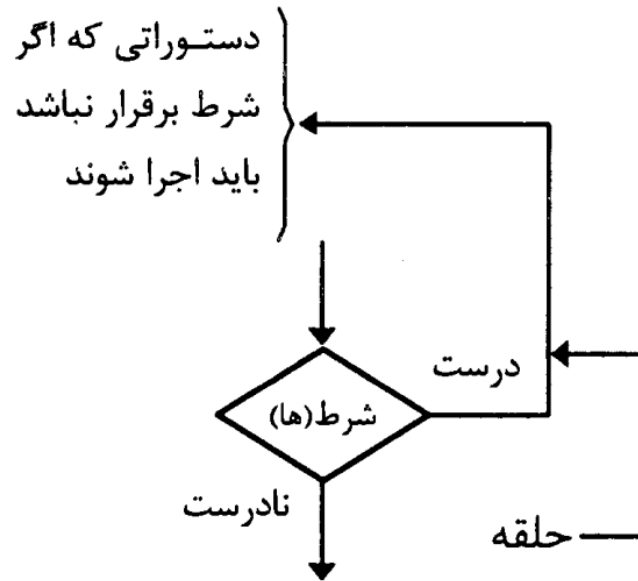
فرض کنید می‌خواهید در این محل ادامه آنرا
بنویسید



علائم فلوچارت

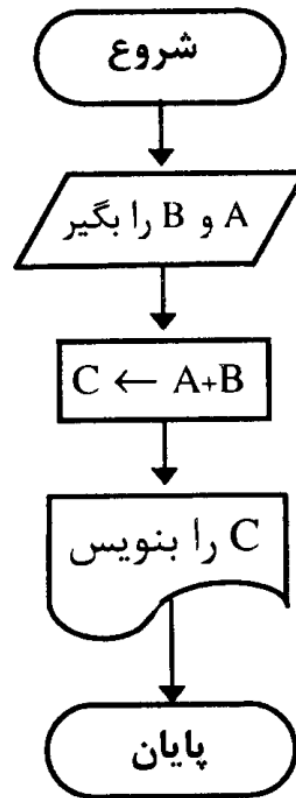
علامت حلقه

برای ساخت حلقه‌ها از ترکیب یک شرط و یک سری دستورالعمل استفاده می‌کنیم که علائم فلوچارت آن‌ها نیز بصورت زیر است .



فلوچارت

مثال ۱) فلوچارتی رسم کنید که دو عدد A و B را به عنوان ورودی گرفته و حاصل جمع آنها را چاپ نماید (روی صفحه کاغذ).



۱ - شروع

۲ - A و B را بگیر

۳ - $C \leftarrow A + B$

۴ - C را چاپ کن

۵ - پایان

فلوچارت

مثال ۲) مطلوب است نوشتن الگوریتم و رسم فلوچارتی که عدد حقیقی x را دریافت و مقدار تابع چند ضابطه‌ای زیر را محاسبه و چاپ کند.

$$f(x) = \begin{cases} x+1 & x < 0 \\ -x^2 & 0 \leq x < 1 \\ \frac{1}{x} & x \geq 1 \end{cases}$$

۱- شروع

۲- x را بگیر

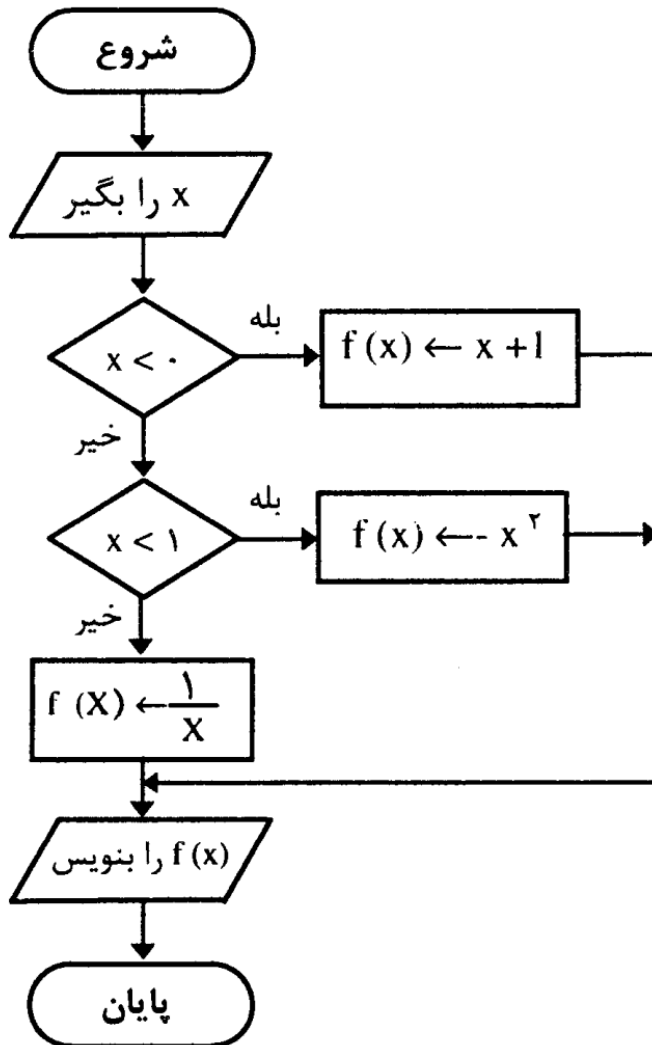
۳- اگر $x < 0$ سپس $f(x) \leftarrow x+1$ و برو به خط ۶

۴- اگر $x < 1$ سپس $f(x) \leftarrow -x^2$ و برو به خط ۶

۵- $f(x) \leftarrow \frac{1}{x}$

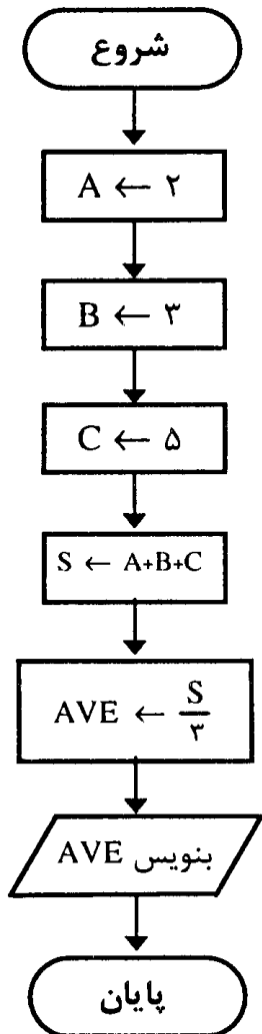
۶- $f(x)$ را بنویس

۷- پایان



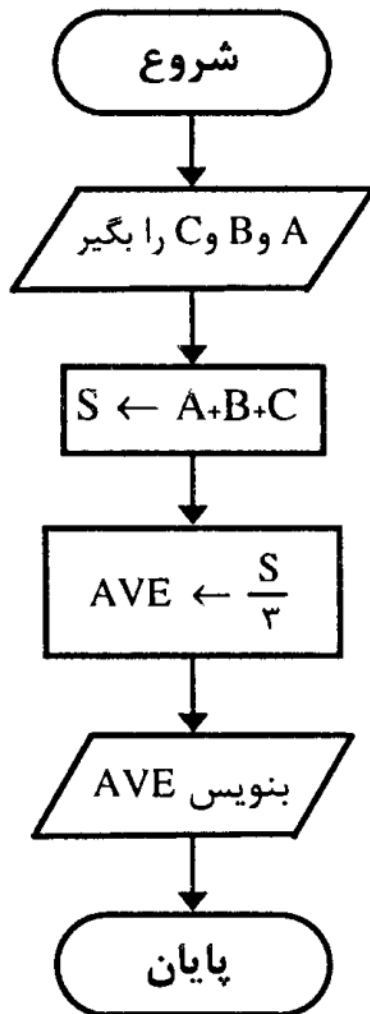
فلوچارت

مثال ۳) فلوچارت مثال ۱ مبحث الگوریتم را رسم نمایید.



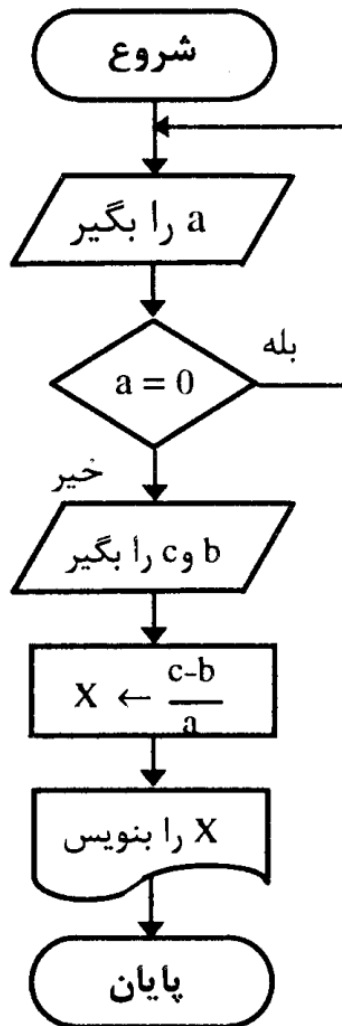
فلوچارت

مثال ۴) فلوچارت مثال ۲ مبحث الگوریتم را رسم نمایید.



فلوچارت

مثال ۵) مطلوب است نوشتن الگوریتم و رسم فلوچارتی که جواب معادله درجه اول یک مجهولی $aX+b=c$ در صورتی که $a \neq 0$ را محاسبه و روی صفحه کاغذ چاپ نماید.



۱ - شروع

۲ - a را بگیر

۳ - اگر $a = 0$ سپس برو به ۲

۴ - b و c را بگیر

۵ - $X \leftarrow \frac{c-b}{a}$

۶ - X را چاپ کن

۷ - پایان

فلوجارت

مثال ۶) مطلوبست نوشتن الگوریتم و رسم فلوجارتی که جوابهای حقیقی معادله درجه دوم یک مجهولی $ax^2 + bx + c = 0$ را در صورتی که $a \neq 0$ را محاسبه و چاپ نماید.

۱ - شروع

۲ - A را بگیر

۳ - اگر $a = 0$ سپس برو به ۲

۴ - b و c را بگیر

۵ - $D \leftarrow b^2 - 4ac$

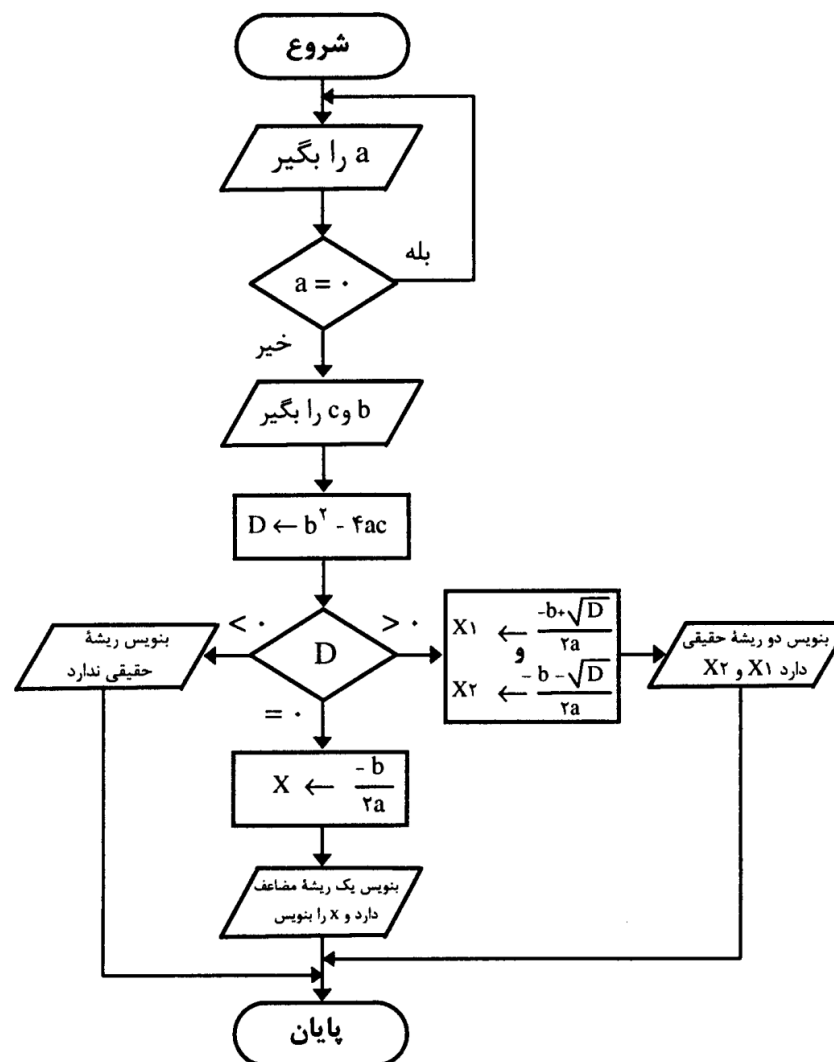
۶ - اگر $D > 0$ سپس $X_1 \leftarrow \frac{-b + \sqrt{D}}{2a}$ و $X_2 \leftarrow \frac{-b - \sqrt{D}}{2a}$ و X_1 و X_2 را بنویس و برو به ۹

۷ - اگر $D = 0$ سپس $X \leftarrow \frac{-b}{2a}$ و بنویس یک ریشه مضاعف دارد و X را بنویس و برو به ۹

۸ - بنویس ریشه حقیقی ندارد

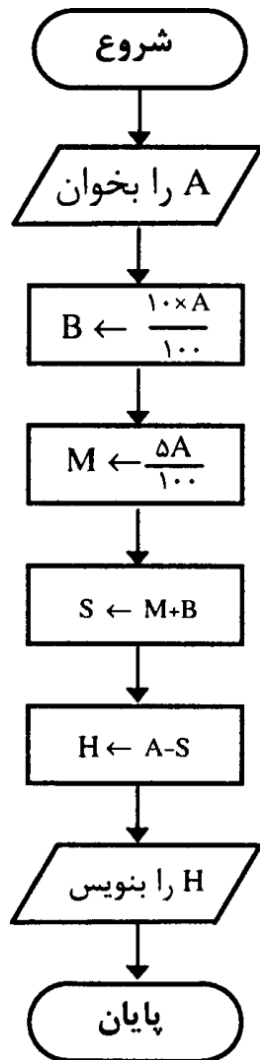
۹ - پایان

فلوچارت



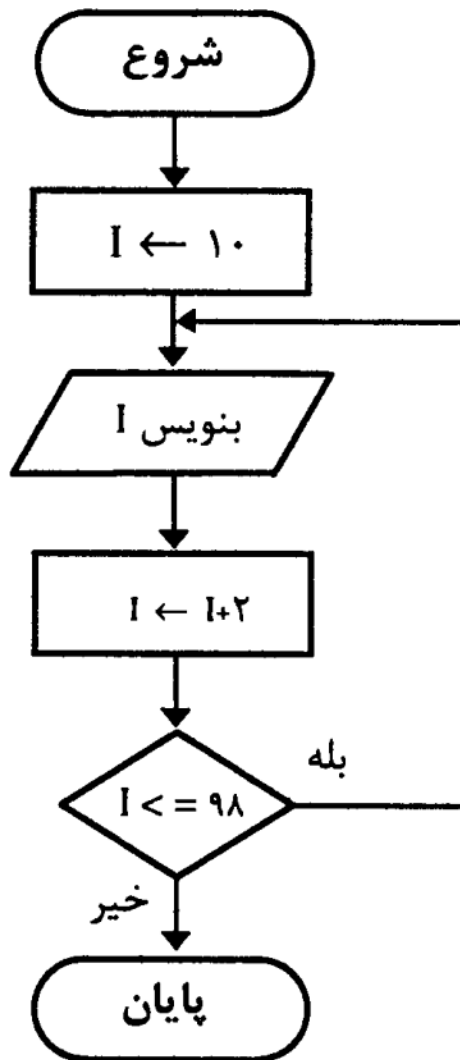
فلوچارت

مثال ۷) فلوچارت مثال ۳ مبحث الگوریتم را رسم نمایید.



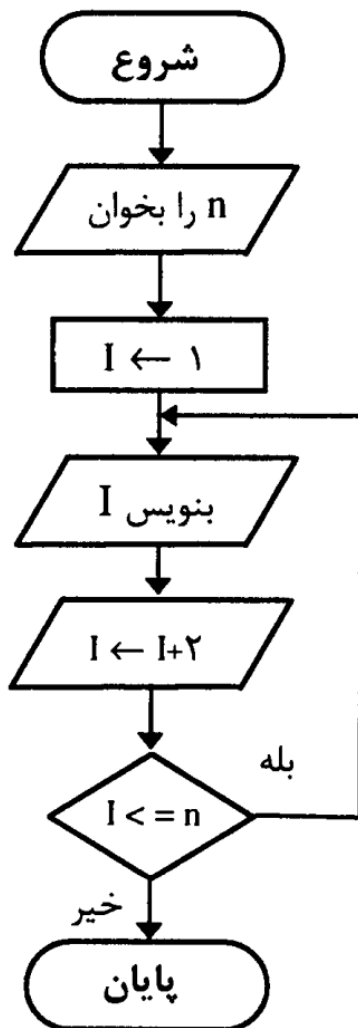
فلوچارت

مثال ۸) فلوچارت مثال ۵ مبحث الگوریتم را رسم نمایید



فلوچارت

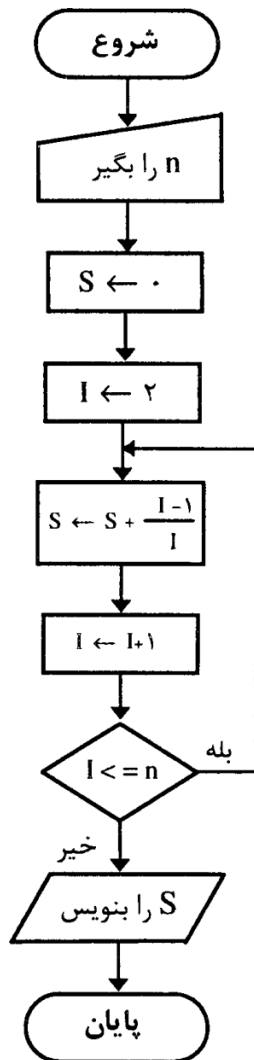
مثال ۹) فلوچارت مثال ۶ مبحث الگوریتم را رسم نمایید.



فلوچارت

مثال ۱۰) مطلوب است نوشتن الگوریتم و رسم فلوچارتی که عدد صحیح و مثبت N را به عنوان ورودی از طریق صفحه کلید دریافت و سپس حاصل جمع زیر را محاسبه و روی صفحه کاغذ چاپ نماید.

$$S = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \dots + \frac{N-1}{N}$$



۱ - شروع

۲ - n را بگیر

۳ - $S \leftarrow ۰$

۴ - $I \leftarrow ۲$

۵ - $S \leftarrow S + \frac{I-1}{I}$

۶ - $I \leftarrow I + ۱$

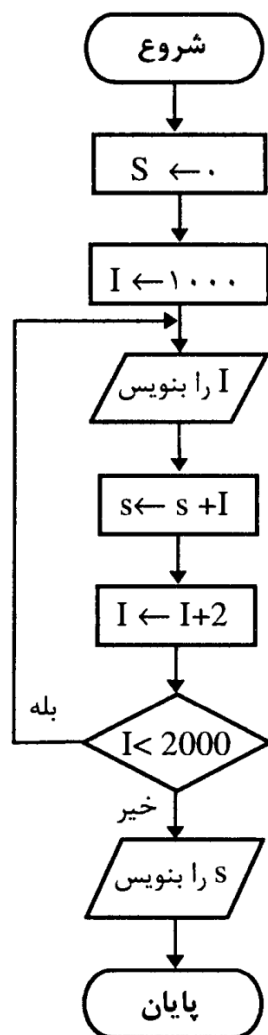
۷ - اگر $I \leq n$ سپس برو به ۵

۸ - S را چاپ کن

۹ - پایان

فلوچارت

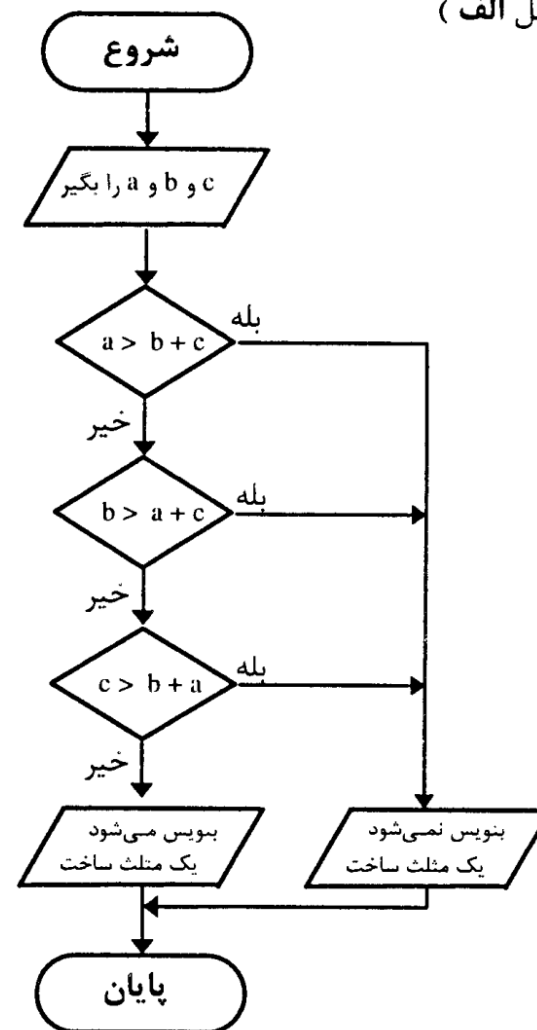
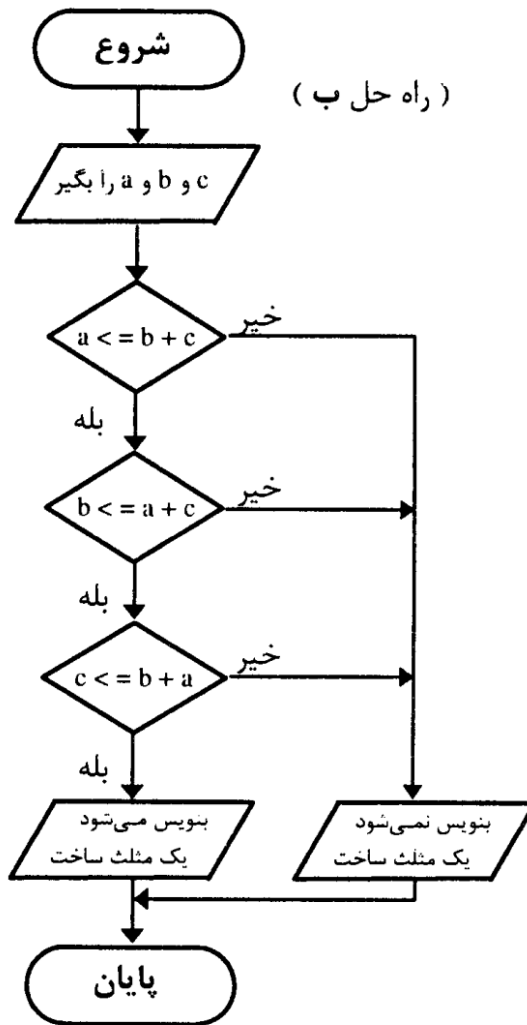
مثال ۱۱) فلوچارت مثال ۷ مبحث الگوریتم را رسم نمایید.



فلوچارت

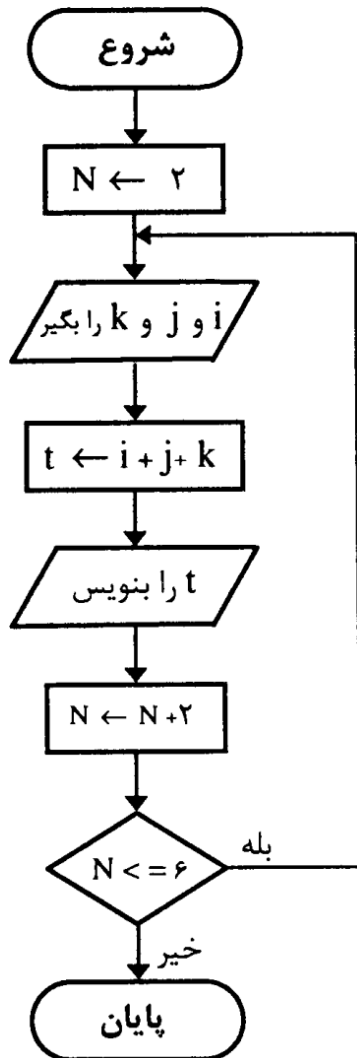
مثال ۱۲) فلوچارت
مثال ۱۱ مبحث
الگوریتم را رسم
نمایید.

(راه حل الف)



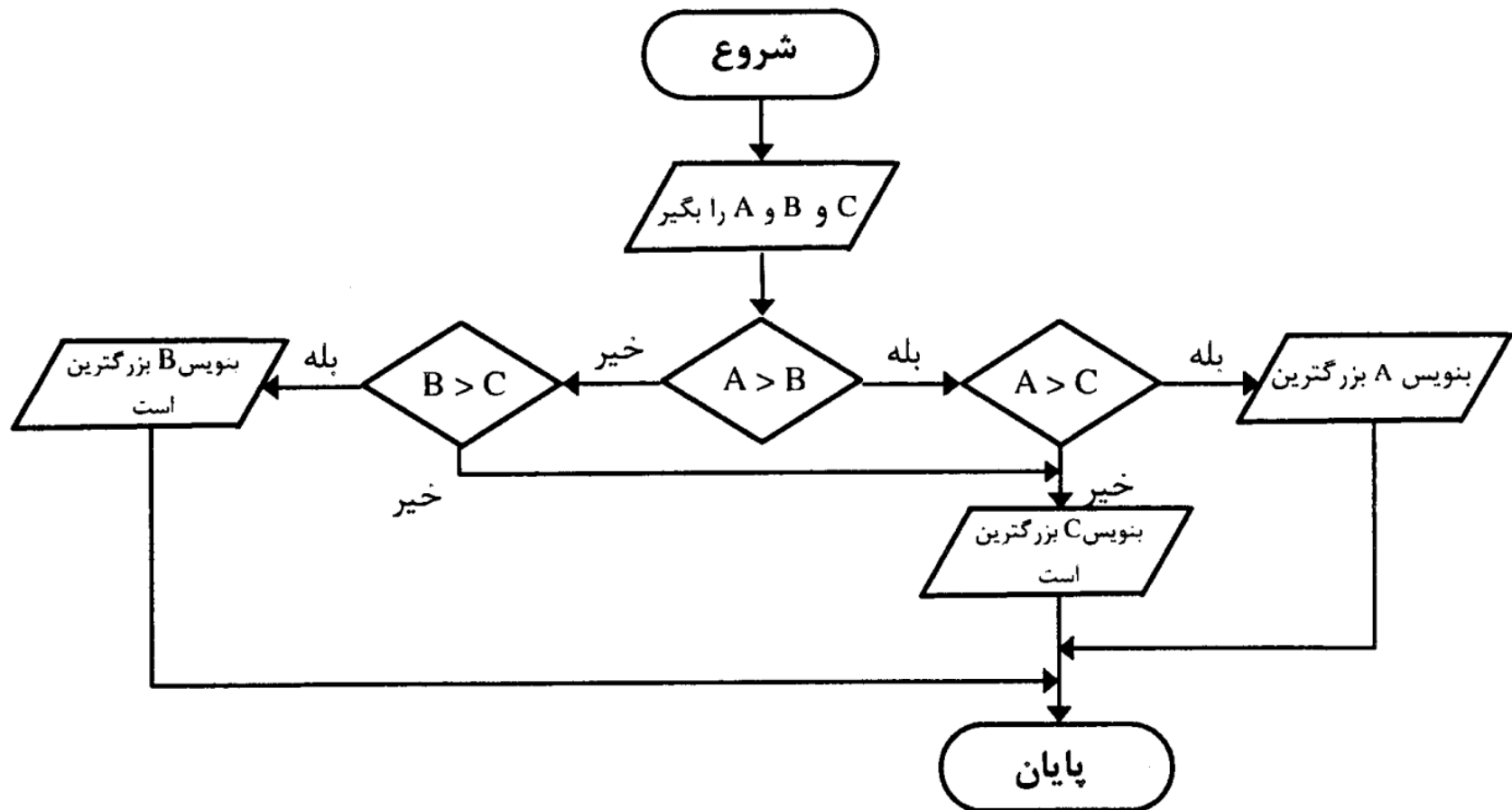
فلوچارت

مثال ۱۳) فلوچارت مثال ۱۴ مبحث الگوریتم را رسم نمایید.



فلوچارت

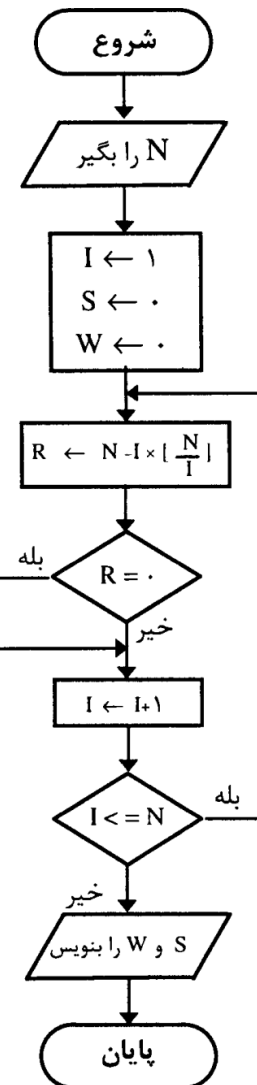
مثال ۱۴) فلوچارت مثال ۱۶ مبحث الگوریتم را رسم نمایید.



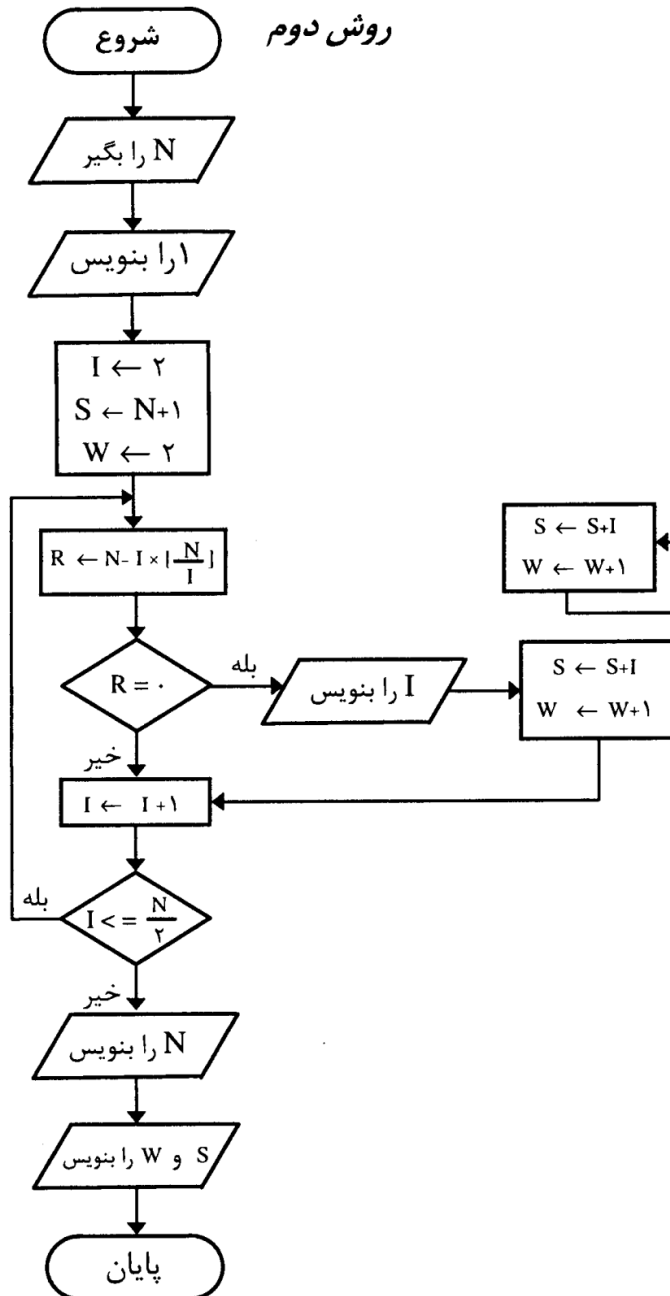
فلوچارت

مثال ۱۵) فلوچارت مثال
۱۸ مبحث الگوریتم را
رسم نمایید.

روش اول

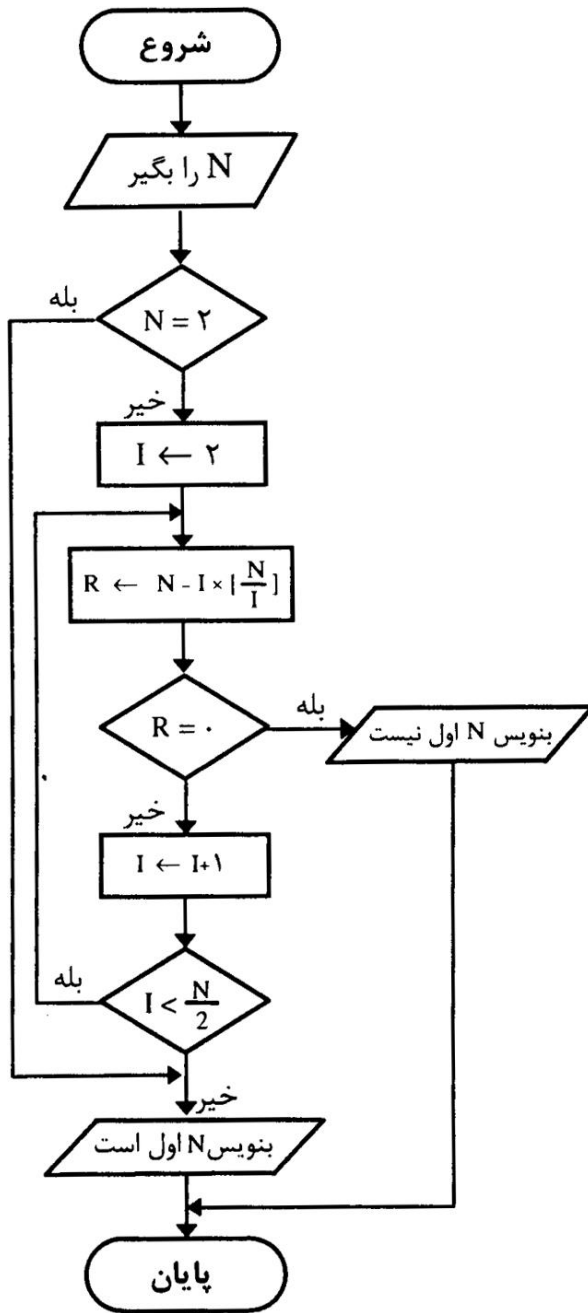


روش دوم



فلوچارت

مثال ۱۶) مطلوب است نوشتن الگوریتم و رسم فلوچارتی که عدد طبیعی N را به عنوان ورودی گرفته و معین کند اول است یا نه ؟ عدد ۲ را نیز در نظر بگیرید و پاسخ به صورت بهینه نوشته شده باشد.



۱ - شروع

۲ - N را بگیر

۳ - اگر $N = 2$ سپس بنویس N اول است و برو به ۱۰

۴ - $I \leftarrow 2$

۵ - $R \leftarrow N - I \times \left[\frac{N}{I} \right]$

۶ - اگر $R = 0$ پس بنویس N اول نیست و برو به ۱۰

۷ - $I \leftarrow I + 1$

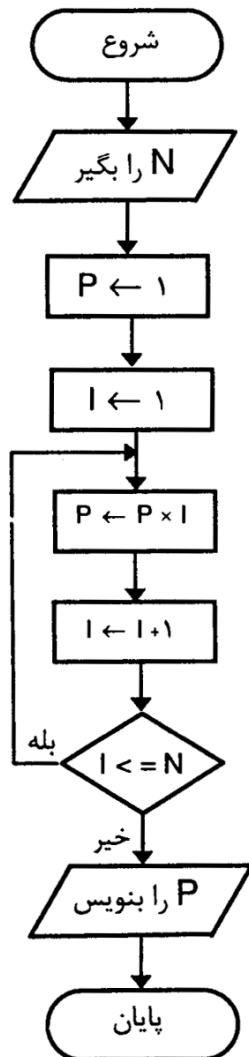
۸ - اگر $I \leq \frac{N}{2}$ پس برو به ۵

۹ - بنویس N اول است

۱۰ - پایان

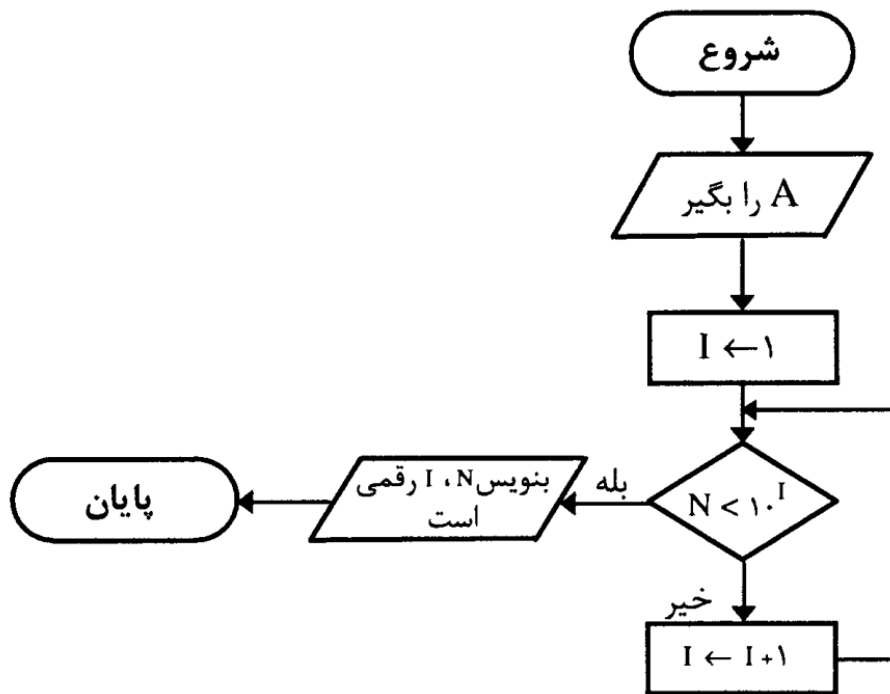
فلوچارت

مثال ۱۷) فلوچارت مثال ۲۱ مبحث الگوریتم را رسم نمایید.



فلوچارت

مثال ۱۸) مطلوبست نوشتن الگوریتم و رسم فلوچارتی یک عدد از ورودی بگیرد و معین کند چند رقمی است.



۱- شروع

۲- A را بگیر

۳- $I \leftarrow 1$

۴- اگر $I < 10.N$ پس بنویس I رقمی است و برو به ۷

۵- $I \leftarrow I + 1$

۶- برو به ۴

۷- پایان



PYTHON

پایتون (PYTHON)

پایتون به انگلیسی Python یک زبان برنامه‌نویسی همه منظوره، مفسر، سطح بالا، شی‌گرا، اسکریپتی و متن باز است که توسط خودو فان راسام به هلندی: Guido van Rossum در سال ۱۹۹۱ در کشور هلند طراحی شد.

فلسفه ایجاد آن تأکید بر دو هدف اصلی **خوانایی بالای برنامه‌های** نوشته شده و **کوتاهی و بازدهی نسبی بالای** آن است. کلمات کلیدی و اصلی این زبان به صورت حداقلی تهیه شده‌اند و در مقابل کتابخانه‌هایی که در اختیار کاربر است بسیار وسیع هستند.

پایتون مدل‌های مختلف برنامه‌نویسی (از جمله شی‌گرا و برنامه‌نویسی دستوری و تابع محور) را پشتیبانی می‌کند و برای مشخص کردن نوع متغیرها از یک سامانه پویا استفاده می‌کند.

پایتون پروژه‌ای آزاد و متن‌باز توسعه‌یافته‌است و توسط بنیاد نرم‌افزار پایتون مدیریت می‌گردد.

<https://www.python.org/>

پایتون (PYTHON)

دانلود پایتون خالص :

<https://www.python.org/downloads/>

دانلود توزیع Anaconda (مورد استفاده در این درس) :

<https://www.anaconda.com/downloads>

دانلود توزیع Canopy :

<https://store.enthought.com/downloads/>

IDE

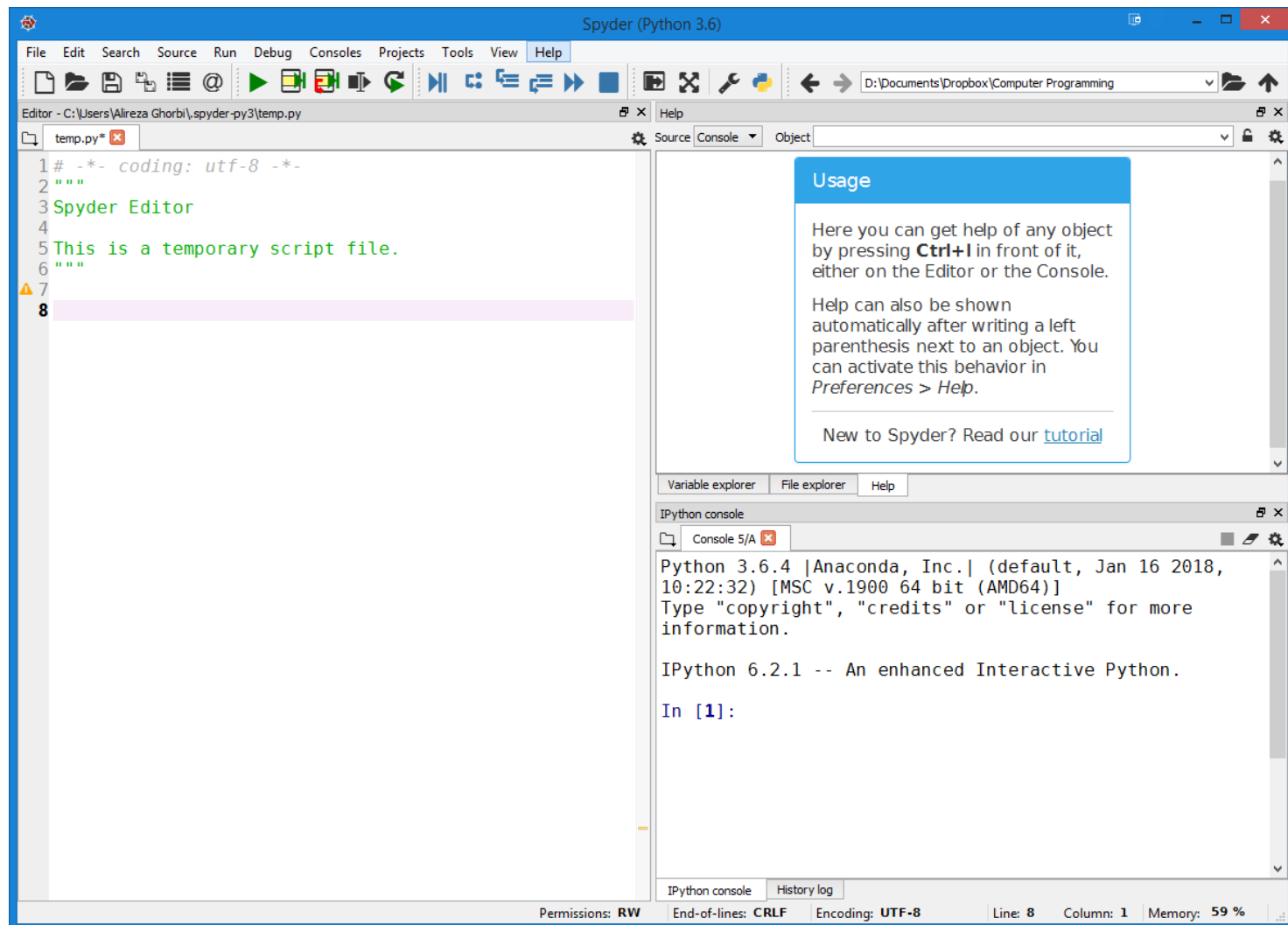
IDE مخفف Integrated development environment به فارسی محیط توسعه یکپارچه

IDE عبارت است از محیطی عمدتاً گرافیکی که تمام یا شماری از ابزارهای لازم برای توسعه نرم افزار (بخش هایی یا تمام زنجیره ابزار توسعه) را خود دارد. در IDE دسترسی به ابزارها و اعمال آنها در پروژه جاری تسهیل شده است.

امکاناتی که به طور معمول در IDE ها وجود دارد:

- ویرایش و نوشتن کد به صورت پیشرفته با استفاده از امکانات پیشنهاد دهنده اتوماتیک که با نوشتن حرف اول یک دستور نام کامل دستورهایی که وجود دارد لیست می شود.
- نمایش کدها به صورت رنگی.
- کمک به رفع عیب های نرم افزار و حل مشکلات آن (Debug).

ANACONDA – SPYDER IDE



OBJECTS (اجزاء، اشیا)

یکی از پایه‌های اولیه نمایش داده‌ها OBJECT ها می باشند. برنامه‌ها داده‌ها را دریافت و روی آن‌ها عملیات را انجام می‌دهند.

OBJECT ها انواع (Type) مختلفی دارند. نوع یک object به برنامه می‌گوید که چه عملیاتی را می‌توان روی داده انجام داد.

OBJECT ها به دو صورت هستند :

- اسکالر (به جزیهای کوچک‌تر قابل تقسیم نمی‌باشد)
- غیر اسکالر (یک ساختار درونی دارد که قابل دسترسی است)

SCALAR OBJECTS

int	معرف اعداد صحیح می باشد. مثلا 5
float	معرف اعداد حقیقی می باشد. مثلا 3.27
bool	معرف Boolean ها یا همان جواب یک مقایسه می باشد که مقادیر آن True و False است.
NoneType	نوعی خاص که یک مقدار None را دارد.

با استفاده از دستور `type()` می توان نوع یک object را متوجه شد.

```
In [1]: type(4)
Out[1]: int
```

```
In [2]: type(4.0)
Out[2]: float
```

SCALAR OBJECTS

```
In [1]: type(5+2)
Out[1]: int
```

```
In [2]: type(5.0+2.0)
Out[2]: float
```

```
In [3]: type(5.0+2)
Out[3]: float
```

```
In [4]: type(5/2)
Out[4]: float
```

```
In [5]: type(5.0/2.0)
Out[5]: float
```

```
In [6]: type(5.0/2)
Out[6]: float
```

```
In [7]: type(5//2)
Out[7]: int
```

```
In [8]: type(5.0//2.0)
Out[8]: float
```

```
In [9]: type(5.0//2)
Out[9]: float
```

```
In [10]: type(True)
Out[10]: bool
```

```
In [11]: type(False)
Out[11]: bool
```

```
In [12]: type(print(5+2))
7
Out[12]: NoneType
```

```
In [13]: type(5**2)
Out[13]: int
```

```
In [14]: type(5.0**2.0)
Out[14]: float
```

```
In [15]: type(5.0*2)
Out[15]: float
```

نمایش خروجی در کنسول

برای نمایش خروجی در کنسول از دستور `print` استفاده می‌شود.

```
In [4]: float(5)
```

```
Out[4]: 5.0
```

```
In [5]: int(3.0)
```

```
Out[5]: 3
```

```
In [6]: 3+2
```

```
Out[6]: 5
```

```
In [7]: print(3+2)
```

```
5
```

```
In [8]: type(print(3+2))
```

```
5
```

```
Out[8]: NoneType
```


عملیات بر روی int ها و float ها

$i+j$	جمع	❑ اگر هر دو int باشند، نتیجه int است.
$i-j$	تفریق	❑ اگر هر دو float باشند، نتیجه float است.
$i*j$	ضرب	
i/j	تقسیم	❑ نتیجه همیشه float می باشد.
$i//j$	خارج قسمت صحیح تقسیم	❑ اگر هر دو int باشند، نتیجه int است.
$i\%j$	باقیمانده تقسیم	❑ اگر هر دو float باشند، نتیجه float است.
$i**j$	توان	

عبارات (EXPRESSIONS)

با ترکیب object ها با operator ها (عملیات و علائم ریاضی) عبارات ساخته می‌شوند.

هر عبارت یک نتیجه دارد، و آن نتیجه یک type دارد.

syntax (گرامر) یک عبارت ساده بدین صورت است :

<object> <operator> <object>

عملیات ساده

از پرانتزها برای تقدم عملیات استفاده کنید :

$$3*5+1=16$$

$$3*(5+1)=18$$

تقدم علائم بدون پرانتز :

**

*

/

+ و -

انجام عملیات از چپ به راست

PYTHON

اجزای اصلی برنامه

انتساب متغیرها و مقادیر

از علامت مساوی برای انتساب مقدار به متغیر استفاده می‌شود.

```
pi=3.14159
```

```
pi_approx=22/7
```

خلاصه سازی عبارات

چرا برای مقادیر اسم تعریف کنیم ؟

- استفاده از اسامی به جای مقادیر
- ساده بودن تغییر کد در آینده

```
pi=3.14159
```

```
radius = 2
```

```
area = pi*(radius**2)
```

برنامه نویسی در مقابل ریاضی

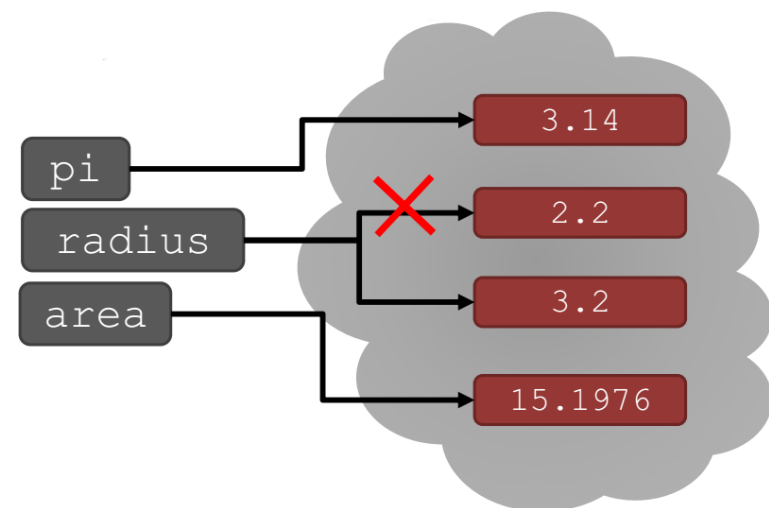
در برنامه نویسی گام‌های رسیدن به نتیجه مهم می‌باشند. به عبارتی، به سادگی مسئله را بر x حل نمی‌کنیم.

```
pi=3.14159
radius = 2
# area of circle
area = pi*(radius**2)
radius = radius+1
```

تغییر انتساب

- می‌توان اسامی متغیرها را دوباره انتساب کرد.
 - مقدار ممکن است هنوز در حافظه باشد ولی handle (هویت) خود را از دست داده است.
- در مثال زیر مقدار مساحت تا زمانی که به آن دستور مجدد ندهید تغییر نمی‌کند.

```
pi=3.14159  
radius = 2.2  
area = pi*(radius**2)  
radius = radius+1
```



انتساب متغیرها با =

- محاسبه سمت راست ← مقدار (VALUE)
- ذخیره (انتساب) آن در حافظه سمت چپ ← متغیر (VARIABLE)
- سمت چپ با مقدار جدید جایگزین می‌شود.

$X = 2$

$X = X * X$

$Y = X + 1$

مثالی از انتساب

▪ تغییر متغیر

❖ آیا بدین شکل درست است ؟

```
x = 1
```

```
y = 2
```

```
y = x
```

```
x = y
```

▪ تغییر متغیر

✓ شکل صحیح!

```
x = 1
```

```
y = 2
```

```
temp = y
```

```
y = x
```

```
x = temp
```

عملیات و علائم مقایسه بر روی int ها و float ها

$i > j$	مقایسه i بزرگتر از j
$i \geq j$	مقایسه i بزرگتر یا مساوی j
$i < j$	مقایسه i کوچکتر از j
$i \leq j$	مقایسه i کوچکتر یا مساوی j
$i == j$	مقایسه i مساوی j
$i != j$	مقایسه i نامساوی j

```
In [16]: 2>1  
Out[16]: True
```

```
In [17]: 2>=2  
Out[17]: True
```

```
In [18]: 2>3  
Out[18]: False
```

```
In [19]: 3==3  
Out[19]: True
```

```
In [20]: 3!=4  
Out[20]: True
```

```
In [21]: 3<4  
Out[21]: True
```

```
In [22]: 3<=3  
Out[22]: True
```

عملیات منطقی بر روی bool ها

```
In [63]: a = 2
```

```
In [64]: b = 3
```

```
In [65]: not a > 1  
Out[65]: False
```

```
In [66]: not a > 2  
Out[66]: True
```

```
In [67]: not a == 2  
Out[67]: False
```

```
In [68]: not a != 2  
Out[68]: True
```

```
In [69]: 2 is not a  
Out[69]: False
```

```
In [70]: 3 is not a  
Out[70]: True
```

```
In [71]: 2 is not b  
Out[71]: True
```

```
In [72]: 3 is not b  
Out[72]: False
```

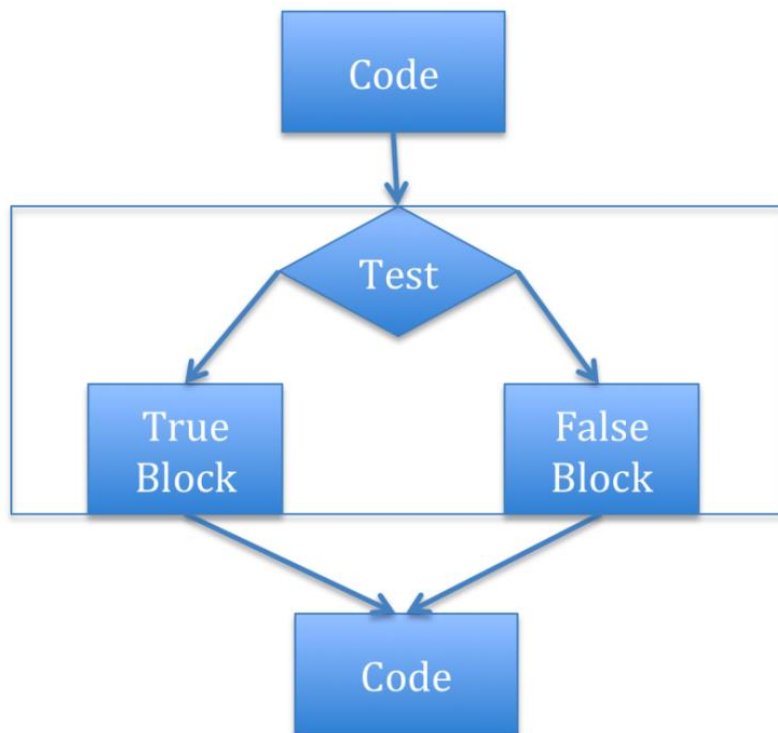
not a	<input type="checkbox"/> True اگر a غلط باشد. <input type="checkbox"/> False اگر a درست باشد.
a and b	شرط برای مقایسه a و b <input type="checkbox"/> True اگر هر دو صحیح باشند.
a or b	شرط برای مقایسه a یا b <input type="checkbox"/> True اگر یکی صحیح باشند.

```
In [73]: a > 1 and b <= 4  
Out[73]: True
```

```
In [74]: a > 2 and b <= 4  
Out[74]: False
```

```
In [75]: a > 2 or b <= 4  
Out[75]: True
```

شاخه‌سازی در برنامه



ساده‌ترین روش شاخه‌سازی عبارات شرطی می‌باشند.

❖ مقایسه و تست (True یا False)

❖ بلوکی از کد برای اجرا در صورت صحیح بودن نتیجه مقایسه (True)

❖ بلوک اختیاری از کد برای اجرا در صورت غلط بودن نتیجه مقایسه (False)

TYPES

▪ متغیرها و عبارات

int ○

float ○

bool ○

string (جدید) ○

○ ... و غیره که در ادامه درس به آنها اشاره می‌شود.

STRINGS

- حروف، کاراکترهای خاص، فاصله و اعداد
- درون " " و یا ' ' قرار می گیرد (quotation marks or single quotes)

```
hi = "hello there"
```

```
greetings = 'hello'
```

- ترکیب string ها

```
name = "alireza"
```

```
greet = hi + name
```

```
greeting = hi + " " + name
```

عملیات بر روی STRING ها

- 'ab' + 'cd' → ترکیب
- 3 * 'alireza' → ترکیب پی در پی
- len('alireza') → طول عبارت
- 'alireza'[1] → اندیسی از عبارت
- 'alireza'[1:3] → برش عبارت
- str(14) → تبدیل عدد به متن

```
In [76]: "ali" + "reza"  
Out[76]: 'alireza'
```

```
In [77]: 3 * "alireza"  
Out[77]: 'alirezaalirezaalireza'
```

```
In [78]: len("alireza")  
Out[78]: 7
```

```
In [79]: 'alireza'[0]  
Out[79]: 'a'
```

```
In [80]: 'alireza'[2]  
Out[80]: 'i'
```

```
In [81]: 'alireza'[1:4]  
Out[81]: 'lir'
```

```
In [82]: 'alireza'[2:]  
Out[82]: 'ireza'
```

```
In [83]: 'alireza'[:5]  
Out[83]: 'alire'
```

```
In [84]: 'alireza'[:]  
Out[84]: 'alireza'
```


خروجی : print

✓ برای نمایش خروجی در کنسول

✓ پرینت ترکیبی

کد :

```
x = 1
print(x)
x_str = str(x)
print("my fav num is", x, ". x =", x)
print("my fav num is " + x_str + ". " + "x = " + x_str)
```

نتیجه :

```
1
my fav num is 1 . x = 1
my fav num is 1. x = 1
```

ورودی : input(" ")

- هر چیزی که داخل " " قرار بگیرد را نشان می‌دهد.
- کاربر چیزی را تایپ می‌کند و Enter را میزند.
- می‌توان آن را به متغیری انتساب داد که بعداً به آن ارجاع شود.

```
text = input("Type anything... ")  
print(5*text)
```

- دستور input خروجی **string** می‌دهد، در نتیجه اگر نیاز به عدد باشد باید آن را تبدیل کرد.

```
num = int(input("Type a number... "))  
print(5*num)
```

کنترل جریان کد و شاخه‌سازی (شرط)

```
if <condition>:  
    <expression>  
    <expression>  
    ...
```

```
if <condition>:  
    <expression>  
    <expression>  
    ...  
else:  
    <expression>  
    <expression>  
    ...
```

```
if <condition>:  
    <expression>  
    <expression>  
    ...  
elif <condition>:  
    <expression>  
    <expression>  
    ...  
else:  
    <expression>  
    <expression>  
    ...
```

<condition> مقدار True یا False دارد.

<expression> ها زمانی اجرا می‌شوند که در آن بلوک نتیجه مقایسه صحیح True باشد.

کنترل جریان کد و شاخه‌سازی

مثالی ساده برای چک کردن زوج یا فرد بودن یک عدد صحیح :

```
x = int(input('Enter an integer: '))
if (x % 2) == 0:
    print('')
    print('Even')
else:
    print('')
    print('Odd')
print('Done with conditional')
```

دو نمونه از اجرای کد :

Enter an integer: 4

Even

Done with conditional

Enter an integer: 5

Odd

Done with conditional

بررسی مثال گذشته

- عبارت $x \% 2 == 0$ مقدار **True** را می‌دهد زمانی که باقیمانده x بر 2 عدد 0 باشد.
- توجه شود که برای مقایسه از $==$ و برای تعریف متغیر از $=$ استفاده شده است.
- فاصله‌ها و تورفتگی‌های کد حائز اهمیت هستند. هر فرورفتگی نشان دهنده یک بلوک از دستورات می‌باشد. (برای مثال اگر خط آخر نیز دارای تورفتگی باشد، دستورات مربوط به آن خط جزئی از بلوک `else` از کد خواهد شد)
- دقت شود که این تورفتگی‌ها ساختار تصویری را به کد می‌دهند که بازتاب دهنده معنا (Semantic) برنامه می‌باشد.

دستورات شرطی تو در تو

مثال : چک کردن مضرب 2 و 3 بودن یک عدد صحیح.

کد :

```
x = int(input('Enter an integer: '))
if (x % 2) == 0:
    if (x % 3) == 0:
        print('Divisible by 2 and 3')
    else:
        print('Divisible by 2 and not by 3')
elif (x % 3) == 0:
    print('Divisible by 3 and not by 2')
else:
    print('Not divisible by 2 or 3')
```

اگر به جای `==` از `=` استفاده کنید، چه اتفاقی خواهد افتاد ؟

پاسخ : کد با خطا مواجه می‌شود (invalid syntax یا گرامر (نحو، دستور) اشتباه)

دستورات شرطی تو در تو

نمونه نتیجه مثال گذشته :

```
Enter an integer: 6  
Divisible by 2 and 3
```

```
In [98]: runfile('D:/Documents/Dropbox/Computer Programming/nestedConds.py',  
wdir='D:/Documents/Dropbox/Computer Programming')
```

```
Enter an integer: 8  
Divisible by 2 and not by 3
```

```
In [99]: runfile('D:/Documents/Dropbox/Computer Programming/nestedConds.py',  
wdir='D:/Documents/Dropbox/Computer Programming')
```

```
Enter an integer: 9  
Divisible by 3 and not by 2
```

```
In [100]: runfile('D:/Documents/Dropbox/Computer Programming/  
nestedConds.py', wdir='D:/Documents/Dropbox/Computer Programming')
```

```
Enter an integer: 11  
Not divisible by 2 or 3
```

مقایسه‌های ترکیبی

مثال : مقایسه 3 عدد غیر یکسان و چاپ کوچکترین آن‌ها

کد :

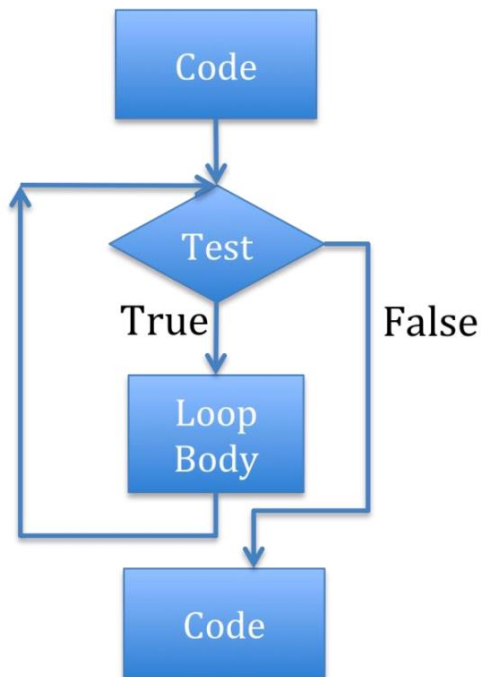
```
x = int(input('Enter an integer: '))
y = int(input('Enter an integer: '))
z = int(input('Enter an integer: '))
if (x != y) and (x != z) and (y != z):
    if x < y and x < z:
        print('x is the least')
    elif y < z:
        print('y is the least')
    else:
        print('z is the least')
else:
    print('at least two numbers were equal')
```


مقایسه‌های ترکیبی

حالت دیگری از کد مثال گذشته :

```
x = int(input('Enter an integer: '))
y = int(input('Enter an integer: '))
z = int(input('Enter an integer: '))
if (x is not y) and (x is not z) and (y is not z):
    if x < y and x < z:
        print('x is the least')
    elif y < z:
        print('y is the least')
    else:
        print('z is the least')
else:
    print('at least two numbers were equal')
```

کنترل با حلقه‌ها



- شاخه‌سازی ساده در برنامه فقط حق انتخاب می‌دهد، ولی همچنان مسیر روند کد خطی است.
- ممکن است نیاز به اجرای دفعات نامشخصی از یک قسمت کد باشد.
- حلقه‌ها امکان توسعه و گسترش الگوریتم‌های ساده شاخه‌سازی به الگوریتم‌های پیچیده را می‌دهند.

- با یک مقایسه و تست شروع می‌شوند.
- اگر مقایسه True باشد، بدنه حلقه یکبار اجرا می‌شود و دوباره به شرط اولیه باز می‌گردد.
- عملیات تا زمانی که مقایسه False شود ادامه پیدا می‌کند.
- پس از آن از حلقه خارج شده و دستورات پس از حلقه اجرا خواهند شد.

کنترل جریان با حلقه‌های while

```
while <condition>:  
    <expression>  
    <expression>  
    ...
```

- <condition> ارزیابی مقایسه
- اگر <condition> درست باشد (True)، تمامی دستورات درون بلوک while انجام می‌شود.
- دوباره <condition> بررسی می‌شود.
- این روند تا زمانی که <condition> غلط شود (False) تکرار خواهد شد.

کنترل جریان با حلقه‌های for

```
for <variable> in range(<some_number>):
```

```
    <expression>
```

```
    <expression>
```

```
    ...
```

- در هر بار اجرای حلقه، <variable> مقداری را به خود می‌گیرد.
- دفعه اول، <variable> با کمترین مقدار شروع می‌شود.
- دفعه بعد، <variable> مقدار قبلی به علاوه یک خواهد شد.
- و تکرار

کنترل جریان با حلقه‌های while

خروجی :

10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48

مثال : چاپ اعداد صحیح و زوج دو رقمی کوچکتر از 50

کد :

```
n = 10
while n < 50:
    print(n)
    n = n+2
```

کنترل جریان با حلقه‌های for

خروجی :

10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48

مثال : چاپ اعداد صحیح و زوج دو رقمی کوچکتر
از 50

کد :

```
for n in range(10,50,2):  
    print(n)
```

دستور range (بازه)

```
sum_a = 0
for i in range(7, 10):
    sum_a += i
print("sum_a value is", sum_a)
```

```
sum_b = 0
for i in range(5, 11, 2):
    sum_b += i
print("sum_b value is", sum_b)
```

```
sum_a value is 24
sum_b value is 21
```

فرم کلی دستور بازه به صورت `range(start, stop, step)` است.

- تعریف start و step اختیاری است، ولی تعریف مقدار stop اجباری است.
- در صورتی که مقدار start (شروع) وارد نشود، مقدار اولیه آن `start = 0` در نظر گرفته می‌شود.
- در صورتی که مقدار step (گام) وارد نشود، مقدار اولیه آن `step = 1` در نظر گرفته می‌شود.
- بازه ابتدا بسته و انتها باز است، بدین معنی که از مقدار start شروع می‌شود و تا مقدار `stop - 1` ادامه پیدا خواهد کرد.

دستور break

- فوراً از حلقه خارج می‌شود.
- از سایر عبارات و دستوراتی که در آن بلوک قرار دارد صرف نظر می‌کند.
- تنها از درونی‌ترین حلقه خارج می‌شود.

```
while <condition_1>:  
    while <condition_2>:  
        <expression_a>  
        break  
        <expression_b>  
    <expression_c>
```


دستور break

مثال : خروجی کد زیر چه مقداری می باشد ؟

```
sum_b = 0
for i in range(5, 11, 2):
    sum_b += i
    if sum_b == 5:
        break
print("sum_b value is", sum_b)
```

for در مقابل while

while	for
تعداد حلقه‌ها نامحدود است.	تعداد حلقه‌ها را می‌دانیم.
با دستور break می‌توان زودتر از موعد از حلقه بیرون آمد.	با دستور break می‌توان زودتر از موعد از حلقه بیرون آمد.
می‌توان از شمارنده استفاده کرد ولی بایستی قبل از حلقه مقدار اولیه آن را مشخص کرد و درون حلقه مقدار آن را افزایش داد.	از شمارنده استفاده می‌کند.
ممکن است نتوان حلقه while را با for بازنویسی کرد.	حلقه for را می‌توان با while بازنویسی کرد.

حلقه‌ها

مثال : مربع یک عدد صحیح با استفاده از جمع پی در پی
کد:

```
x = int(input("Enter an integer: "))
ans = 0
itersLeft = x
while (itersLeft != 0):
    ans = ans + x
    itersLeft = itersLeft - 1
print(str(x) + '*' + str(x) + ' = ' + str(ans))
```

نمونه نتیجه :

```
Enter an integer: 2
2*2 = 4
```

حلقه‌ها

بررسی گام به گام کد مثال گذشته زمانی که کاربر عدد $x = 3$ را وارد کرده باشد:

x	ans	itersLeft
3	0	3
	3	2
	6	1
	9	1

```
x = int(input("Enter an integer: "))
ans = 0
itersLeft = x
while (itersLeft != 0):
    ans = ans + x
    itersLeft = itersLeft - 1
print(str(x) + '*' + str(x) + ' = ' + str(ans))
```

برخی از خواص حلقه‌ها

- ❑ نیاز به تعریف مقدار اولیه متغیر تکرار شونده در خارج از حلقه
- ❑ نیاز به تست مقدار متغیر برای تشخیص پایان عملیات
- ❑ نیاز به تغییر مقدار متغیر درون حلقه در کنار سایر عملیات

برخی از خواص حلقه‌ها

- نیاز به تعریف متغیر حلقه
 - تعریف مقدار اولیه متغیر در خارج از حلقه
 - تغییر مقدار متغیر درون حلقه
 - تست مقدار متغیر برای تشخیص پایان عملیات

شاخه‌ها (شروط) و حلقه‌ها

□ شاخه‌ها (دستورات شرطی) به ما امکان پرش به مکان دیگری از کد را با استفاده از بررسی یک مقایسه (شرط) می‌دهند.

➤ برنامه‌ها از نظر زمان ثابت (constant) هستند.

□ حلقه‌ها (مثل while) به ما امکان تکرار قسمتی از کد را می‌دهند تا زمانی که یک شرط برقرار باشد.

➤ در این حالت زمان اجرای برنامه بسته به مقدار متغیرها و حجم دستورات متغیر است.

PYTHON

الگوریتم‌های ساده

روش حدس و بررسی

- الگوریتم‌های تکرار شونده (Iterative) به ما امکان اجرای کارهای پیچیده‌تری را می‌دهند. یکی از این نوع الگوریتم‌ها **روش حدس و بررسی** می‌باشد :
 - توانایی **حدس مقدار** نتیجه
 - توانایی **بررسی درست بودن** نتیجه
 - حدس مجدد تا زمانی که حل پیدا شود یا اینکه تمامی مقادیر حدس زده شده باشند.
 - به این فرآیند **شمارش جامع** و به انگلیسی EXHAUSTIVE ENUMERATION گفته می‌شود.
- روش حدس و بررسی برای مسائلی که تعداد محدودی امکان دارند قابل استفاده است.
- شمارش جامع روش مناسبی برای تولید حدس‌ها در حالتی منظم می‌باشد.

روش حدس و بررسی

مثال : پیدا کردن ریشه سوم عدد صحیح به روش حدس و بررسی

- ✓ برای پیدا کردن ریشه سوم x یک راه حل این است که مقادیر $0^{**}3$ و $1^{**}3$ و $2^{**}3$ و ... را بررسی نماییم.
- ✓ زمانی که به k رسیدیم دست نگه داریم به صورتی که $k^{**}3 > x$
- ✓ محدود کردن کیس‌های مورد بررسی

روش حدس و بررسی

کد اولیه :

```
x = int(input('Enter an integer: '))
guess = 0
while guess**3 < x:
    guess = guess + 1
if guess**3 != x:
    print(x, 'is not a perfect cube')
else:
    print('Cube root of', x, 'is', guess)
```

روش حدس و بررسی

توسعه طرح نهایی مثال قبل :

- کد فقط برای اعداد مثبت کار می‌کند.
- راه حل ساده آن دنبال کردن علامت حل و انجام عملیات برای حالت مثبت می باشد.

روش حدس و بررسی

کد دوم مثال قبل :

```
x = int(input('Enter an integer: '))
guess = 0
while guess**3 < abs(x):
    guess = guess + 1
if guess**3 != abs(x):
    print(x, 'is not a perfect cube')
else:
    if x < 0:
        guess = -guess
    print('Cube root of', x, 'is', guess)
```

اگر یکی از شروط حلقه جا انداخته شود چه اتفاقی خواهد افتاد ؟

□ فرض کنید متغیر حلقه را در خارج از حلقه تعیین نکنیم ؟

- احتمالا با خطای `NameError` مواجه می‌شویم و در شرایط بدتر بدون متوجه شدن ما، کد از متغیری که از قبل در حافظه است استفاده کند.

□ فرض کنید که متغیر حلقه را درون حلقه تغییر ندهیم ؟

- به حلقه نامتناهی بر می‌خوریم که هیچ‌گاه به پایان نمی‌رسد.

روش حدس و بررسی

کد سوم و تمیزتر مثال قبل :

```
x = int(input('Enter an integer: '))
for guess in range(x + 1):
    if guess**3 == x:
        print('Cube root of', x, 'is', guess)
```

روش حدس و بررسی

کد چهارم و کامل تر مثال قبل :

```
x = int(input('Enter an integer: '))
for guess in range(abs(x + 1)):
    if guess**3 >= abs(x):
        break
if guess**3 != abs(x):
    print(x, 'is not a perfect cube')
else:
    if x < 0:
        guess = - guess
    print('Cube root of', x, 'is', guess)
```


نکات تکمیلی STRING ها

- ❑ STRING ها را می‌توان با `<`, `>`, `==` و غیره مقایسه کرد.
- ❑ دستور `len()` برای بدست آوردن طول عبارت داخل پرانتز استفاده می‌شود.
- ❑ از براکت در انتهای STRING برای شمارش اندیس‌ها و بدست آوردن مقدار یک اندیس/موقعیت استفاده می‌شود.
 - `'alireza'[1] → 'l'`
 - `'ali'[-1] → 'i'`
 - `len('alireza') → 7`

نکات تکمیلی STRING ها

▪ STRING را می‌توان با `[start: stop: step]` برش زد.

➤ `'alireza'[1:3] → 'li'`

➤ `'alireza'[1:7:2] → 'lrz'`

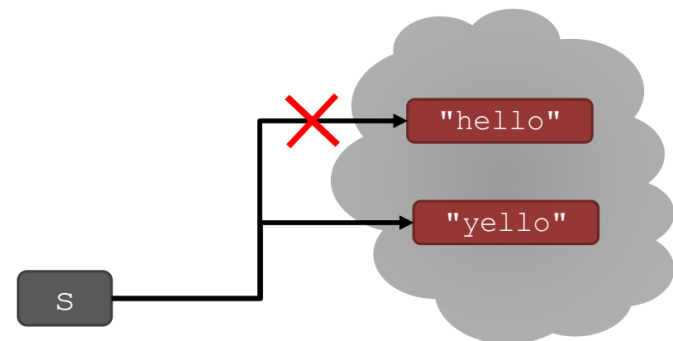
➤ `'alireza'[::-1] → 'azerila'`

▪ STRING ها در اصطلاح **immutable** هستند، بدین معنی که تغییر ناپذیرند.

➤ `s = "hello"`

➤ `s[0] = 'y' →` خطا می‌دهد

➤ `s = 'y' + s[1:] →` مجاز است



STRING ها و حلقه‌های for

- range یک راه برای تکرار روی اعداد است، اما متغیر حلقه for می‌تواند روی هر نوعی از مقادیر تکرار شود.

```
s = 'abcdefghi'
```

```
for index in range(len(s)):
    if s[index] == "i" or s[index] == "u":
        print("found i or u")
```

```
for char in s:
    if char == 'i' or char == 'u':
        print("found i or u")
```

تمرین کلاسی 1

فرض کنید `s` یک `string` از حروف کوچک می باشد. برنامه ای بنویسید که تعداد حروف صدادار در `string` را حساب کند.

حروف صدادار: `a, e, i, o, u` می باشند.

برای مثال اگر `s = 'azcbobobegghakl'` باشد، خروجی برنامه به صورت زیر بایستی شود :

Number of vowels: 5

تمرین کلاسی 2

فرض کنید `s` یک `string` از حروف کوچک می‌باشد. برنامه‌ای بنویسید که تعداد دفعاتی که کلمه 'ok' در `s` وجود دارد را محاسبه کند.

برای مثال اگر `s = 'oklhgksokko'` باشد، خروجی برنامه به صورت زیر بایستی شود :

Number of times ok occurs is: 2

تمرین کلاسی 3

فرض کنید `s` یک `string` از حروف کوچک می‌باشد. برنامه‌ای بنویسید که طولانی‌ترین زیر `string` از `s` را که حروف آن به ترتیب حروف الفبا می‌باشد را پیدا کند و آن را چاپ کند. به طور مثال اگر `s = 'azcbobobegghakl'` باشد، خروجی برنامه به صورت زیر است :

Longest substring in alphabetical order is: beggh

و اگر به تعداد حروف برابری رسید اولین زیر `string` را چاپ کند.
به طور مثال اگر `s = 'abcbcd'` باشد، خروجی به صورت زیر است :

Longest substring in alphabetical order is: abc

روش‌های تقریبی

- فرض کنید حال می‌خواهیم ریشه سوم هر عدد غیر منفی را بدست آوریم.
- نمی‌توانیم پاسخ دقیق را ضمانت کنیم، اما پاسخ نزدیک کافی است.
- از روش شمارش جامع (EXHAUSTIVE ENUMERATION) استفاده می‌کنیم.
 - با استفاده از گام‌های کوچک، حدس‌ها را تولید می‌کنیم.
 - تست برای چک کردن پاسخ نزدیک

روش‌های تقریبی

- پاسخ خوب مناسب
- شروع با یک حدس و افزایش آن با یک مقدار کوچک
- $|guess^3 - \epsilon| \leq \epsilon$ برای یک اپسیلون کوچک
- کاهش مقدار افزایش گام \leftarrow برنامه کندتر
- افزایش مقدار اپسیلون \leftarrow پاسخ تقریبی‌تر

روش‌های تقریبی

محاسبه ریشه سوم هر عدد غیر منفی :

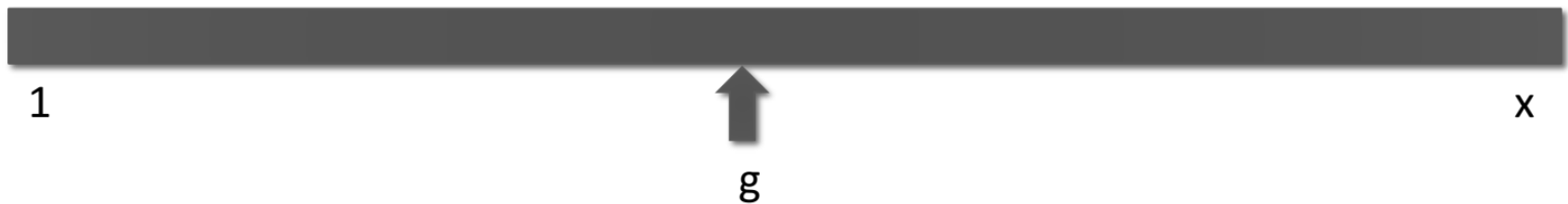
```
cube = 29
epsilon = 0.001
guess = 0.0
increment = 0.0001
num_guesses = 0
while abs(guess**3 - cube) >= epsilon and guess <= cube:
    guess += increment
    num_guesses += 1
print('num_guesses =', num_guesses)
if abs(guess**3 - cube) >= epsilon:
    print('Failed on cube root of', cube)
else:
    print('%.4f' % guess, 'is close to the cube root of', cube)
```

روش‌های تقریبی

- گام می‌تواند هر عدد کوچکی باشد
- اگر خیلی کوچک باشد، زمان بیشتری برای پیدا کردن ریشه سوم نیاز است.
- اگر خیلی بزرگ باشد، ممکن است از روی پاسخ بپریم و از آن دور شویم.
- در نتیجه به روش بهینه‌تری برای این مسئله نیاز داریم.

جستجوی دو بخشی (Bisection Search)

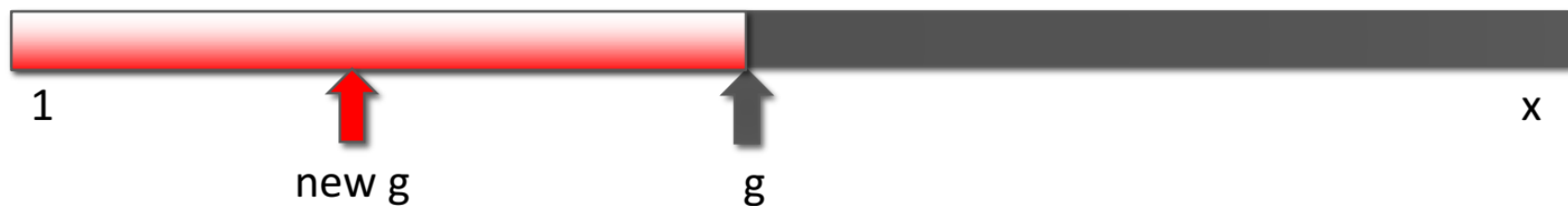
- از ریاضیات می‌دانیم که ریشه سوم هر عدد x بزرگ‌تر مساوی یک بین 1 و x قرار دارد.
- به جای آن که به صورت جامع شمارش را از 1 شروع کنیم، فرض کنید حدسی را وسط بازه انتخاب کنیم.



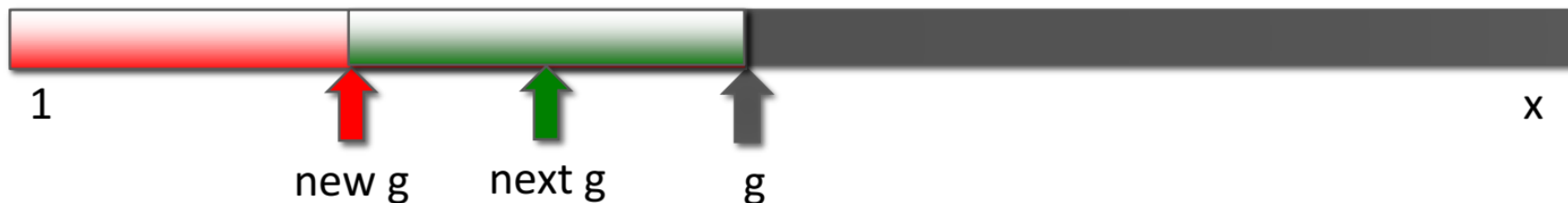
- اگر خوش‌شانس باشیم، پاسخ به اندازه کافی نزدیک خواهد بود.

جستجوی دو بخشی (Bisection Search)

- اگر به پاسخ نزدیک نباشد، آیا حدس بزرگتر بوده یا کوچکتر؟
- اگر $x > g^{**3}$ ، می‌دانیم که g بزرگ بوده و دوباره جستجو می‌کنیم :



- و به طور مثال اگر g جدید به گونه‌ای باشد که $x < g^{**3}$ ، می‌دانیم که g کوچک بوده و دوباره جستجو می‌کنیم :



- در هر مرحله، بازه جستجو به نصف مرحله قبل کاهش پیدا می‌کند.

جستجوی دو بخشی (Bisection Search)

محاسبه ریشه سوم هر عدد غیر منفی به روش Bisection Search :

```
x = 29
epsilon = 0.001
numGuesses = 0
low = 0.0
high = x
ans = (high + low)/2.0

while abs(ans**3 - x) >= epsilon:
    print('low = ' + str(low) + ' high = ' + str(high) + ' ans = ' + str(ans))
    numGuesses += 1
    if ans**3 < x:
        low = ans
    else:
        high = ans
    ans = (high + low)/2.0
print('numGuesses = ' + str(numGuesses))
print(str(ans) + ' is close to cube root of ' + str(x))
```

جستجوی دو بخشی (Bisection Search)

کد اسلاید قبل فقط برای مقادیر بزرگتر مساوی ۱ کار می‌کند، چرا؟

تمرین ← کد قبل را به گونه‌ای اصلاح کنید که برای مقادیر منفی نیز کار کند.

تمرین ← کد قبل را به گونه‌ای اصلاح کنید که برای مقادیر $x < 1$ کار کند.

جستجوی دو بخشی (Bisection Search)

- جستجوی دو بخشی به طور اساسی زمان محاسبه را کاهش می‌دهد.
- انتخاب حدس‌های هوشمندانه بسیار اهمیت دارد.



PYTHON

FUNCTIONS

چگونه یک کد را می‌نویسیم؟

- تا به اینجا...
- ساز و کار زبان برنامه‌نویسی پوشش داده شد
- می‌دانیم که برای هر محاسبه چگونه فایل مجزا ایجاد کنیم
- هر فایل قسمتی از کد می‌باشد
- و هر کد مجموعه‌ای از دستورات پی در پی است
- مشکلات این روش :
 - برای برنامه‌های مقیاس کوچک مناسب است
 - برای برنامه‌های بزرگ به هم ریخته است
 - دنبال کردن جزئیات سخت است
 - خطایابی سخت تر است.

برنامه‌نویسی خوب

- برنامه نویس خوب با تعداد خطوطی که نوشته مشخص نمی‌شود.
- برنامه‌نویس خوب با مقدار خدماتی که آورده ارزیابی می‌شود.
- برای بسته بندی خدمات از **Function** ها (توابع) استفاده می‌کنیم.
- توابع دو جنبه بسیار مهم از تفکر محاسباتی را به ما ارائه می‌دهند: (۱) تجزیه کردن (۲) خلاصه سازی

تجزیه کردن و خلاصه سازی

■ تجزیه

- تقسیم یک مسئله به قسمت‌های مختلف با مفهوم مجزا با استفاده از تابع

■ خلاصه سازی

- خلاصه سازی قسمتی از حل کلی با استفاده از تابع

تجزیه کردن و خلاصه سازی

تجزیه

- تقسیم یک مسئله به قسمت‌های مختلف
 - با مفهوم مجزا
 - و قابل استفاده مجدد
 - برای نظم بخشیدن به کد
 - و ایجاد ارتباط بین اجزای کد

خلاصه سازی

- قسمتی از کد که مثل جعبه سیاه می‌ماند
 - جزئیات را نمی‌بینیم
 - نیازی به دیدن جزئیات نداریم
 - نمی‌خواهیم جزئیات را ببینیم
 - پنهان سازی قسمت‌های کم اهمیت کد

تجزیه کردن و خلاصه سازی

- با یکدیگر قوی هستند.
- کد به مراتب قابل استفاده است ولی تنها یکبار نیاز به خطایابی آن داریم.

توابع (FUNCTIONS)

- نوشتن قسمتی از کد با قابلیت استفاده مجدد، به عنوان تابع
- توابع در برنامه تا زمانی که فرا خوانده نشوند اجرا نمی‌شوند
- خصوصیات تابع
 - اسم (name)
 - پارامترهای ورودی (• یا بیشتر) (parameters)
 - توضیحات (اختیاری است) (docstring)
 - بدنه (body)

نحوه نوشتن تابع و فراخوانی آن

```
def is_even(i):  
    """  
    Input: i, a positive int  
    Return True if i is even, otherwise False  
    """  
    print('Hi')  
    return i % 2 == 0  
  
print(is_even(3))
```

نحوه نوشتن تابع و فراخوانی آن

Keyword
کلیدواژه

Name
اسم

Parameters
ورودی‌ها

```
def is_even(i):  
    """  
    Input: i, a positive int  
    Return True if i is even, otherwise False  
    """  
    print('Hi')  
    return i % 2 == 0
```

docstring
توضیحات

body
بدنه

```
print(is_even(3))
```

نحوه فراخوانی
تابع در کد

درون بدنه تابع

```
def is_even(i):  
    """  
    Input: i, a positive int  
    Return True if i is even, otherwise False  
    """  
    print('Hi')  
    return i % 2 == 0  
  
print(is_even(3))
```

محاسبه یک
سری عبارات
Keyword
کلیدواژه

محاسبه عبارت
و خروجی تابع

تابع بدون عبارت return !

- اگر عبارت return در تابع قرار نداشته باشد، پایتون مقدار **None** را به عنوان خروجی تابع ارائه می‌دهد.

```
def is_even(i):  
    """  
    Input: i, a positive int  
    Return True if i is even, otherwise False  
    """  
    i % 2 == 0
```

print و return

return	print
فقط درون تابع معنی و کاربرد دارد	بیرون تابع نیز قابل استفاده است
فقط یک return درون تابع اجرا می‌شود	به هر تعداد که بخواهیم می‌توانیم از آن درون تابع استفاده کنیم
کدهای درون تابع که بعد از return قرار گرفته‌اند اجرا نمی‌شوند.	کدهای درون تابع بعد از دستور print می‌توانند اجرا شوند
مقداری را به خود اختصاص می‌دهد و به تماس‌گیرنده ارائه می‌دهد (زمانی تابع فراخوانده می‌شود)	مقداری را به خود اختصاص می‌دهد که بر روی کنسول نمایش داده می‌شود

میدان متغیر

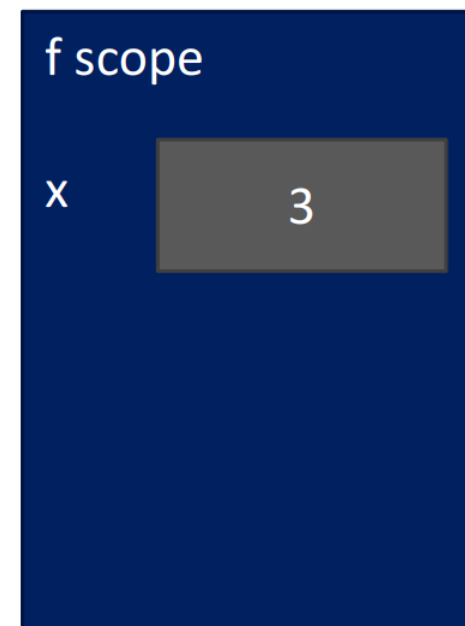
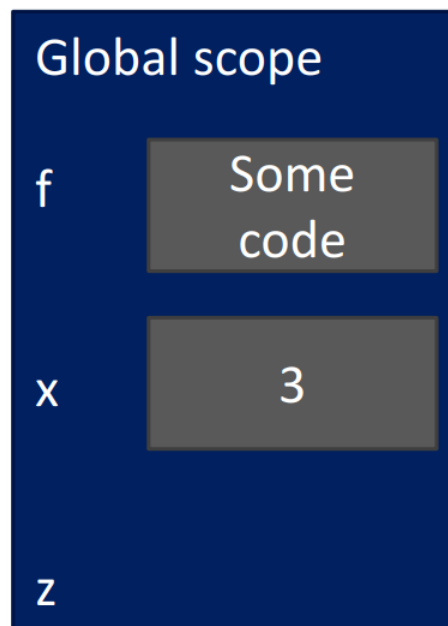
- زمان فراخوانی تابع، پارامتر ورودی تابع به پارامتر اصلی وصل می‌شود.
- هنگام ورود به تابع یک میدان/محیط/قالب (scope/environment/frame) جدید ساخته می‌شود.

پارامتر ورودی

```
def f(x):  
    x = x + 1  
    print('in f(x): x =', x)  
    return x
```

```
x = 3  
z = f(x)
```

فراخوانی f ← پارامتر اصلی

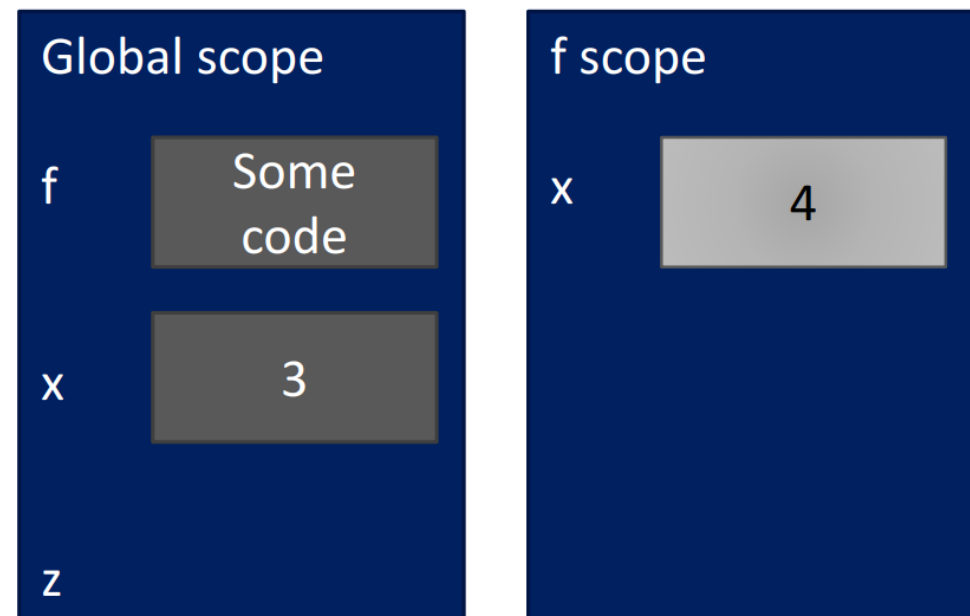


میدان متغیر

- زمان فراخوانی تابع، پارامتر ورودی تابع به پارامتر اصلی وصل می‌شود.
- هنگام ورود به تابع یک میدان/محیط/قالب (scope/environment/frame) جدید ساخته می‌شود.

```
def f(x):  
    x = x + 1  
    print('in f(x): x =', x)  
    return x
```

```
x = 3  
z = f(x)
```

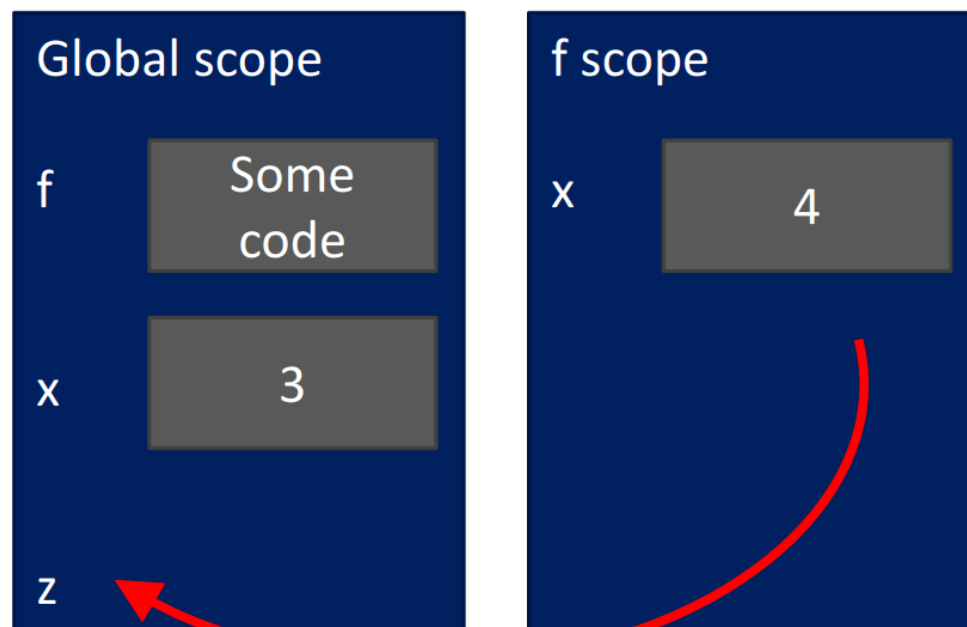


میدان متغیر

- زمان فراخوانی تابع، پارامتر ورودی تابع به پارامتر اصلی وصل می‌شود.
- هنگام ورود به تابع یک میدان/محیط/قالب (scope/environment/frame) جدید ساخته می‌شود.

```
def f(x):  
    x = x + 1  
    print('in f(x): x =', x)  
    return x
```

```
x = 3  
z = f(x)
```



میدان متغیر

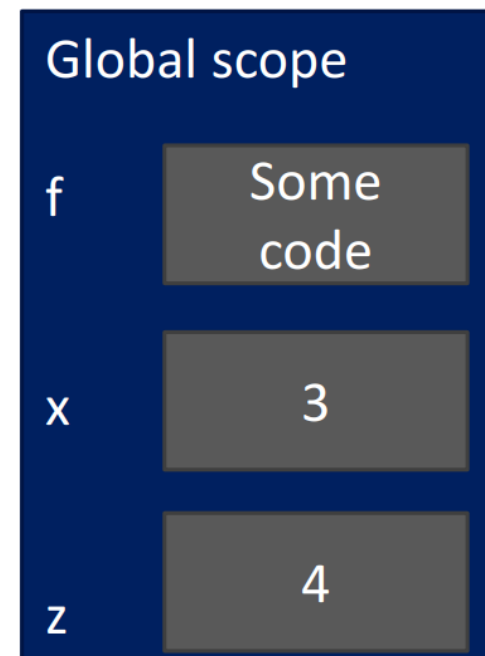
- زمان فراخوانی تابع، پارامتر ورودی تابع به پارامتر اصلی وصل می‌شود.
- هنگام ورود به تابع یک میدان/محیط/قالب (scope/environment/frame) جدید ساخته می‌شود.

```
def f(x):  
    x = x + 1  
    print('in f(x): x =', x)  
    return x
```

```
x = 3  
z = f(x)
```



انتساب مقدار خروجی به متغیر z



ورودی‌های تابع

▪ ورودی‌های تابع می‌توانند به هر فرمی باشند :

```
def func_a():  
    print("inside func_a")
```

```
def func_b(y):  
    print("inside func_b")  
    return y
```

```
def func_c(z):  
    print("inside func_c")  
    return z()
```

```
print(func_a())  
print(5 + func_b(2))  
print(func_c(func_a))
```

فراخوانی تابع func_a ، بدون ورودی

فراخوانی تابع func_b ، یک ورودی

فراخوانی تابع func_c ، یک ورودی (که خود یک تابع دیگر است)

مثالی از میدان (scope)

- درون تابع، به متغیر تعریف شده در خارج از آن می‌توان دسترسی داشت.
- درون تابع، متغیر تعریف شده در خارج از آن قابل تغییر و ویرایش نمی‌باشد.

<p>x مجداً در میدان f تعریف می‌شود</p> <pre>def f(y): x = 1 x += 1 print(x)</pre> <p>x = 5 f(x) print(x)</p> <p>X های متفاوت</p>	<p>x از خارج g خوانده می‌شود</p> <pre>def g(y): print(x) print(x + 1)</pre> <p>x = 5 g(x) print(x)</p> <p>X درون g از روی میدانی که g را فراخوانده انتخاب می‌شود</p>	<pre>def h(y): x = x + 1</pre> <p>x = 5 h(x) print(x)</p> <p>خطای UnboundLocalError : متغیر محلی 'x' قبل از انتساب ارجاع داده شده است.</p>
---	---	--

مثالی از میدان (scope)

- درون تابع، به متغیر تعریف شده در خارج از آن می‌توان دسترسی داشت.
- درون تابع، متغیر تعریف شده در خارج از آن قابل تغییر و ویرایش نمی‌باشد.

<pre>def f(y): x = 1 x += 1 print(x) x = 5 f(2) print(x)</pre>	<pre>def g(y): print(x) print(x + 1) x = 5 g(2) print(x)</pre>	<pre>def h(y): x = x + 1 x = 5 h(2) print(x)</pre>
---	---	---

← X از میدان کلی (global) برنامه

مثالی از میدان (scope)

```
def g(x):  
    def h():  
        x = 'abc'  
    x = x + 1  
    print('in g(x): x =', x)  
    h()  
    return x
```

قطعه‌ای کد

```
x = 3  
z = g(x)
```

Global scope

g

Some
code

x

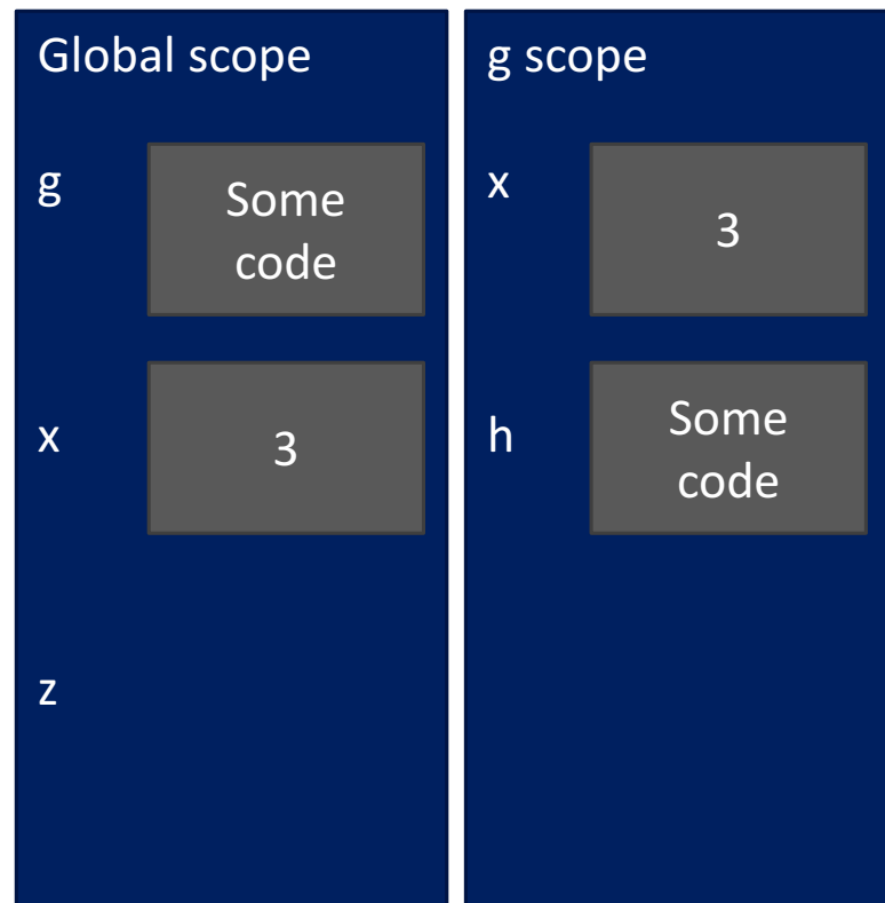
3

z

مثالی از میدان (scope)

```
def g(x):  
    def h():  
        x = 'abc' ←  
    x = x + 1  
    print('in g(x): x =', x)  
    h()  
    return x
```

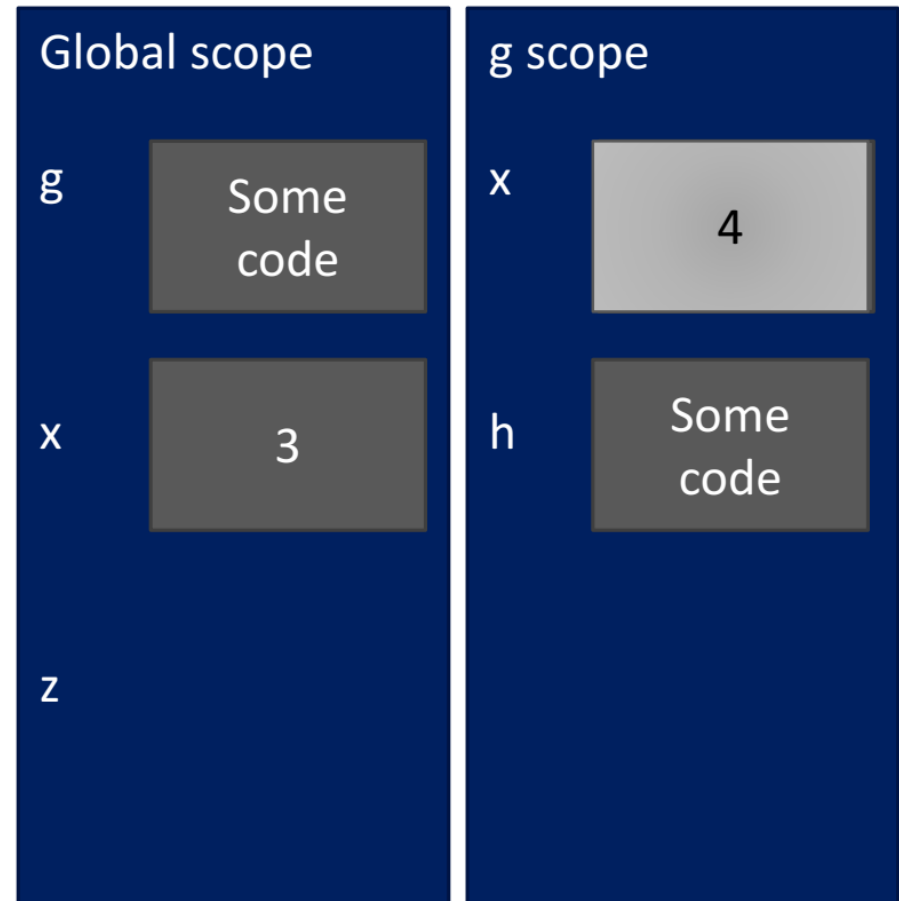
```
x = 3  
z = g(x)
```



مثالی از میدان (scope)

```
def g(x):  
    def h():  
        x = 'abc'  
    x = x + 1  
    print('in g(x): x =', x)  
    h()  
    return x
```

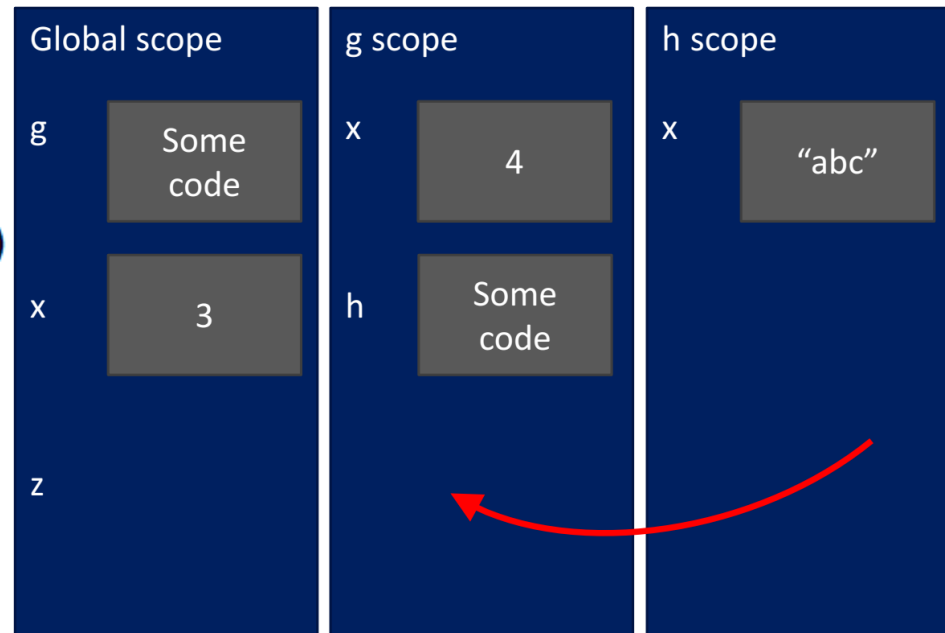
```
x = 3  
z = g(x)
```



مثالی از میدان (scope)

```
def g(x):  
    def h():  
        x = 'abc' ←  
    x = x + 1  
    print('in g(x): x =', x)  
    h() ←  
    return x
```

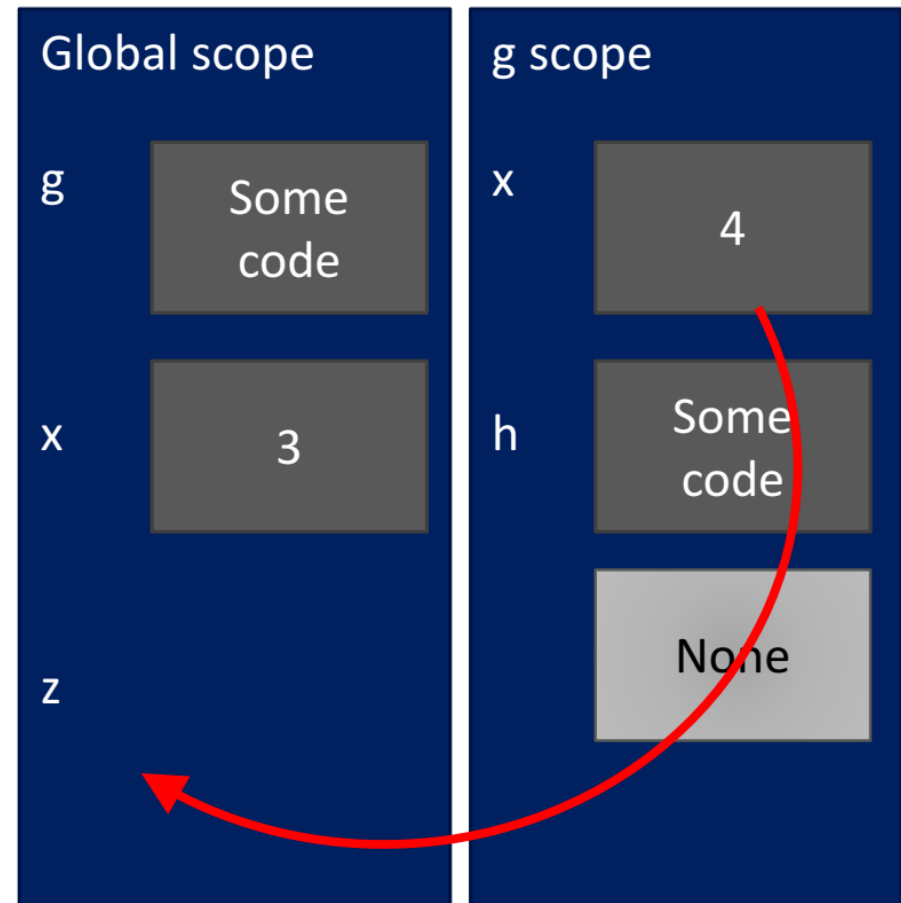
```
x = 3  
z = g(x)
```



مثالی از میدان (scope)

```
def g(x):  
    def h():  
        x = 'abc'  
    x = x + 1  
    print('in g(x): x =', x)  
    h()  
    return x
```

```
x = 3  
z = g(x)
```



مثالی از میدان (scope)

```
def g(x):  
    def h():  
        x = 'abc'  
    x = x + 1  
    print('in g(x): x =', x)  
    h()  
    return x
```

```
x = 3  
z = g(x)
```

Global scope

g

Some
code

x

3

z

4

کلیدواژه‌های پارامترهای ورودی و مقادیر پیش‌فرض

- تابع ساده زیر را برای چاپ نام و نام خانوادگی در نظر بگیرید:
- اگر آخرین ورودی TRUE باشد، بنابراین «نام خانوادگی، نام» چاپ می‌شود، در غیر این صورت «نام و نام خانوادگی» چاپ خواهد شد.

```
def printName(firstName, lastName, reverse):  
    if reverse:  
        print(lastName + ', ' + firstName)  
    else:  
        print(firstName, lastName)
```

کلیدواژه‌های پارامترهای ورودی و مقادیر پیش‌فرض

- تمامی حالت‌های زیر نتیجه یکسانی دارند :

```
def printName(firstName, lastName, reverse):  
    if reverse:  
        print(lastName + ', ' + firstName)  
    else:  
        print(firstName, lastName)
```

```
printName('Alireza', 'Ghorbi', False)  
printName('Alireza', 'Ghorbi', reverse=False)  
printName('Alireza', lastName='Ghorbi', reverse=False)  
printName(lastName='Ghorbi', firstName='Alireza', reverse=False)
```

کلیدواژه‌های پارامترهای ورودی و مقادیر پیش‌فرض

- می‌توان مقادیر پیش‌فرضی را برای ورودی‌ها در نظر گرفت، که اگر مقداری برای آن تعریف نشده باشد از مقدار پیش‌فرض استفاده کند.

```
def printName(firstName, lastName, reverse=False):  
    if reverse:  
        print(lastName + ', ' + firstName)  
    else:  
        print(firstName, lastName)
```

```
printName('Alireza', 'Ghorbi')  
printName('Alireza', 'Ghorbi', True)
```

بازگشت (Recursion) چیست؟

- روشی برای طراحی پاسخ مسائل با ۳ مرحله زیر :
 - تقسیم مسئله به زیر بخش‌های مشابه
 - حل بازگشتی و پی در پی زیر بخش‌ها
 - ترکیب مناسب پاسخ‌ها
- روشی از برنامه‌نویسی که در آن **تابع خود را فرا می‌خواند**.
- در برنامه‌نویسی هدف داشتن بازگشت بی‌نهایت نمی‌باشد.
- بایستی **یک یا چند حالت پایه** داشته باشیم که حل آن بسیار ساده است.
- با هدف ساده‌سازی مسئله بزرگتر، بایستی یک مسئله را با **ورودی‌های متفاوت** حل کنیم

الگوریتم‌های تکراری (Iterative) تا به اینجا

- استفاده از ساختار for و while باعث ایجاد الگوریتم‌های تکراری می‌شدند.
- توانایی انجام محاسبات روی یک سری متغیر وضعیتی که در هر تکرار از حلقه بروزرسانی می‌شدند.

حاصل ضرب ۲ عدد - حل به روش تکرار

- حاصل ضرب "a * b" معادل است با جمع a در خودش به تعداد b مرتبه
- متغیرهای وضعیت :
- شمارنده (b) که از b شروع می شود و به مقدار صفر متوقف می شود. $b-1 \rightarrow b$
- مقدار فعلی محاسبات (result) به صورت $result + a \rightarrow result$

```
def mult_iter(a, b):  
    result = 0  
    while b > 0:  تکرار ، iteration  
        result += a  مقدار فعلی محاسبات  
        b -= 1  مقدار فعلی شمارنده  
    return result
```

حاصل ضرب ۲ عدد - حل به روش بازگشتی

- گام بازگشتی
 - فکر کنید چگونه می‌توان مسئله را به فرم **ساده‌تر/کوچک‌تر** کاهش داد.
- حالت پایه
 - مسئله را تا جایی کاهش دهید تا به حالت ساده‌ای برسید که **مستقیماً حل** شود.
 - زمانی که $b = 1$ باشد $a * b = a$

حاصل ضرب ۲ عدد - حل به روش بازگشتی

$$a * b = \underbrace{a + a + a + a + a + \dots + a}_{b \text{ مرتبه}}$$

b مرتبه

$$= \underbrace{a + a + a + a + a + \dots + a}_{b-1 \text{ مرتبه}}$$

b-1 مرتبه

$$= a + a * (b-1)$$

```
def mult_recur(a, b):
```

```
    if b == 1:  
        return a
```

حالت پایه

```
    else:
```

```
        return a + mult_recur(a, b-1)
```

گام بازگشتی

- $n! = n * (n-1) * (n-2) * (n-3) * ... * 1$
- حالت پایه، اگر $n = 1$ باشد حاصل فاکتوریل نیز یک است.

```
n = 1          →      if n == 1:
                        return 1
```

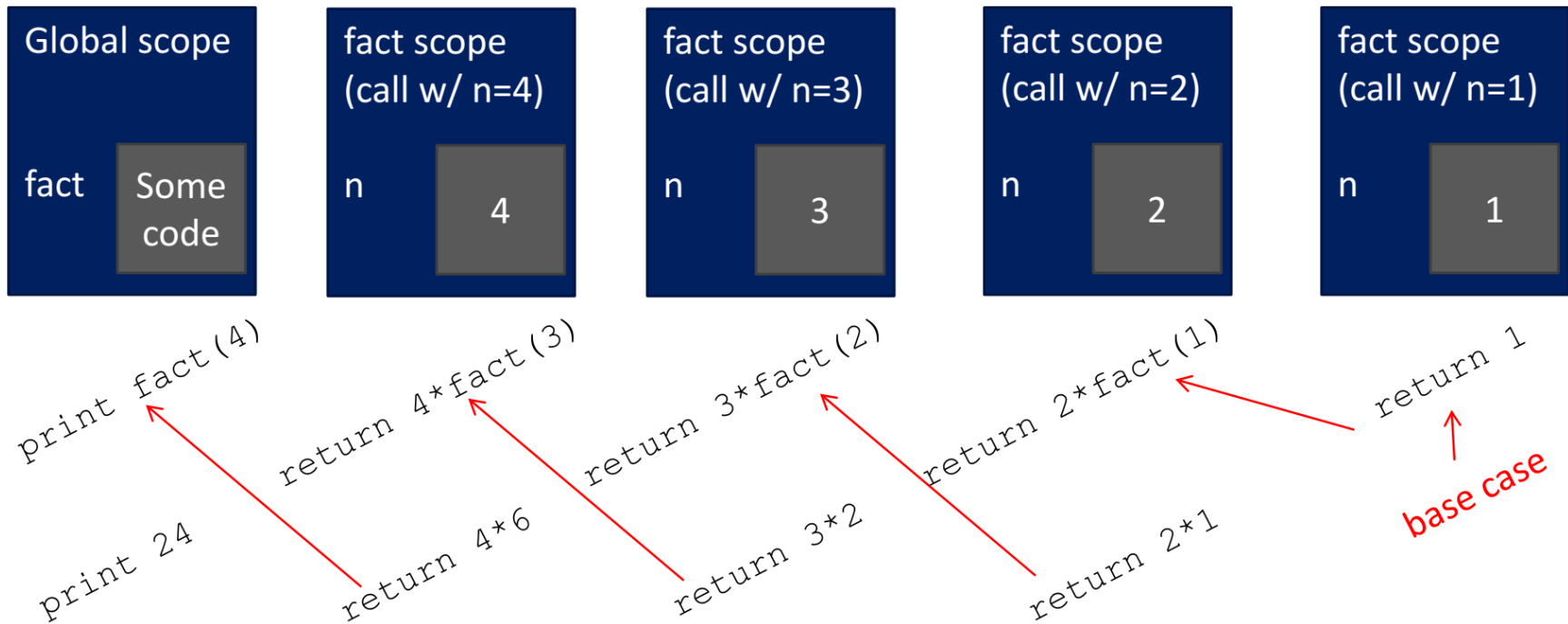
- چگونه مسئله را کاهش دهیم؟ بازنویسی مسئله به فرم ساده‌تری که به جواب حالت پایه برسیم.

```
n*(n-1)!    →    else:
                    return n*factorial(n-1)
```

فاکتوریل به روش بازگشتی

```
def fact(n):  
    if n == 1:  
        return 1  
    else:  
        return n*fact(n-1)
```

```
print(fact(4))
```



فاکتوریل به روش تکراری

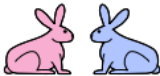
- متغیرهای وضعیت
 - شمارنده i
 - مقدار فعلی محاسبات $prod$

```
def factorial_iter(n):  
    prod = 1  
    for i in range(1, n+1):  
        prod *= i  
    return prod
```

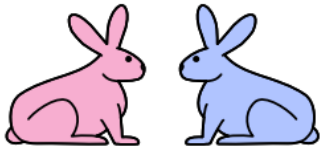
روش بازگشتی با چند حالت پایه

- اعداد فیبوناچی
- لئوناردو فیبوناچی (Leonardo Fibonacci) جالش زیر را مدلسازی کرد :
- یک جفت خرگوش تازه متولد شده (یک مؤنث و یک مذکر) در یک محل قرار داده می‌شوند.
- خرگوش‌ها در سن ۱ ماهگی جفت‌گیری می‌کنند.
- دوره بارداری خرگوش‌ها ۱ ماه می‌باشد.
- فرض این است که خرگوش‌ها هیچوقت نمی‌میرند، بدین معنی که خرگوش مؤنث از ماه دومش، هر ماه یک جفت خرگوش بدنیا می‌آورد (یک مؤنث و یک مذکر).
- تعداد خرگوش‌های مؤنث پس از تعداد ماه مشخص مورد بررسی است.

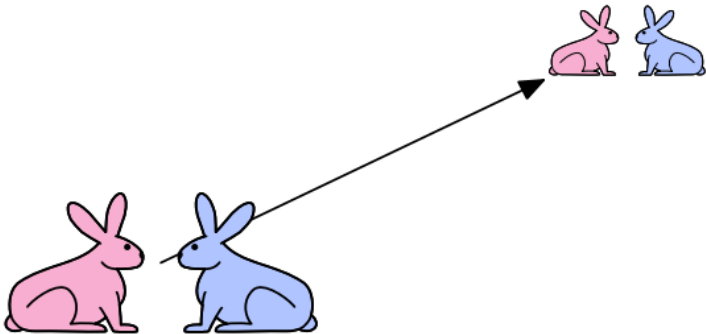
فیوناچی



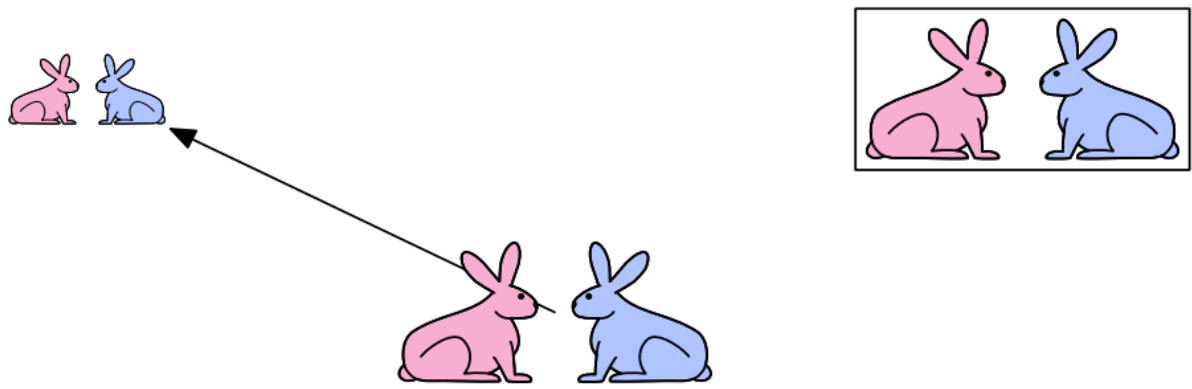
فیوناچی



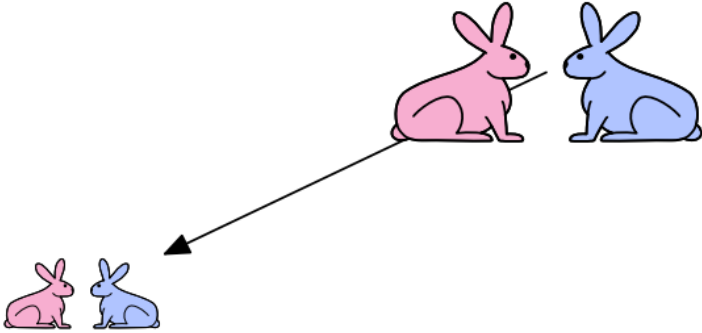
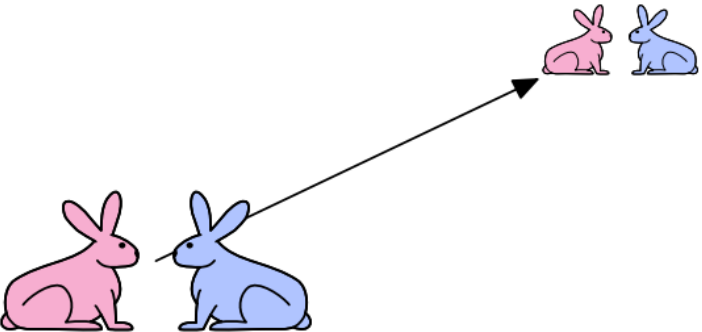
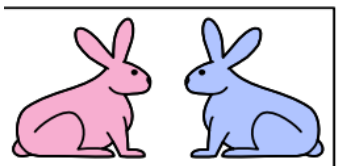
فیوناچی



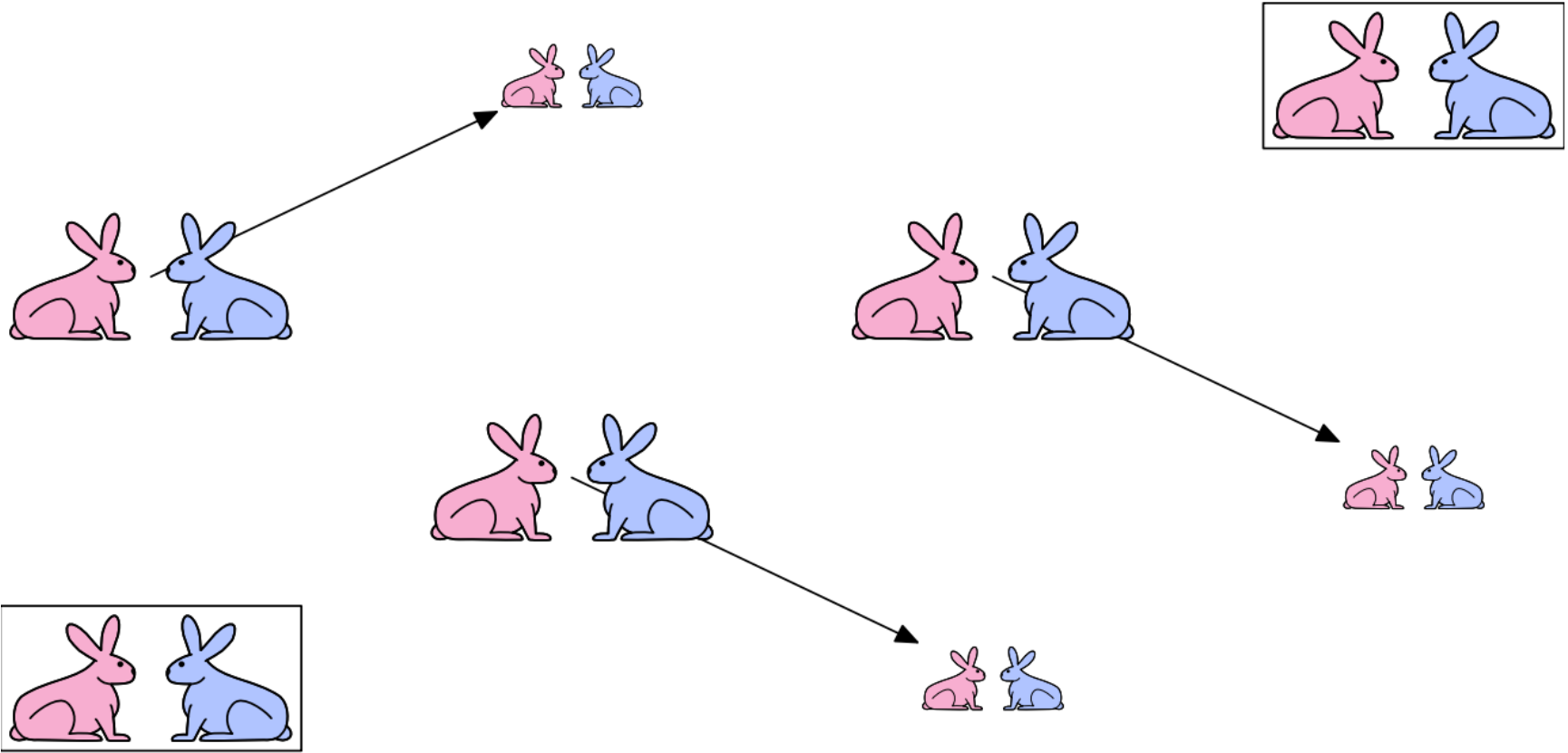
فیوناچی



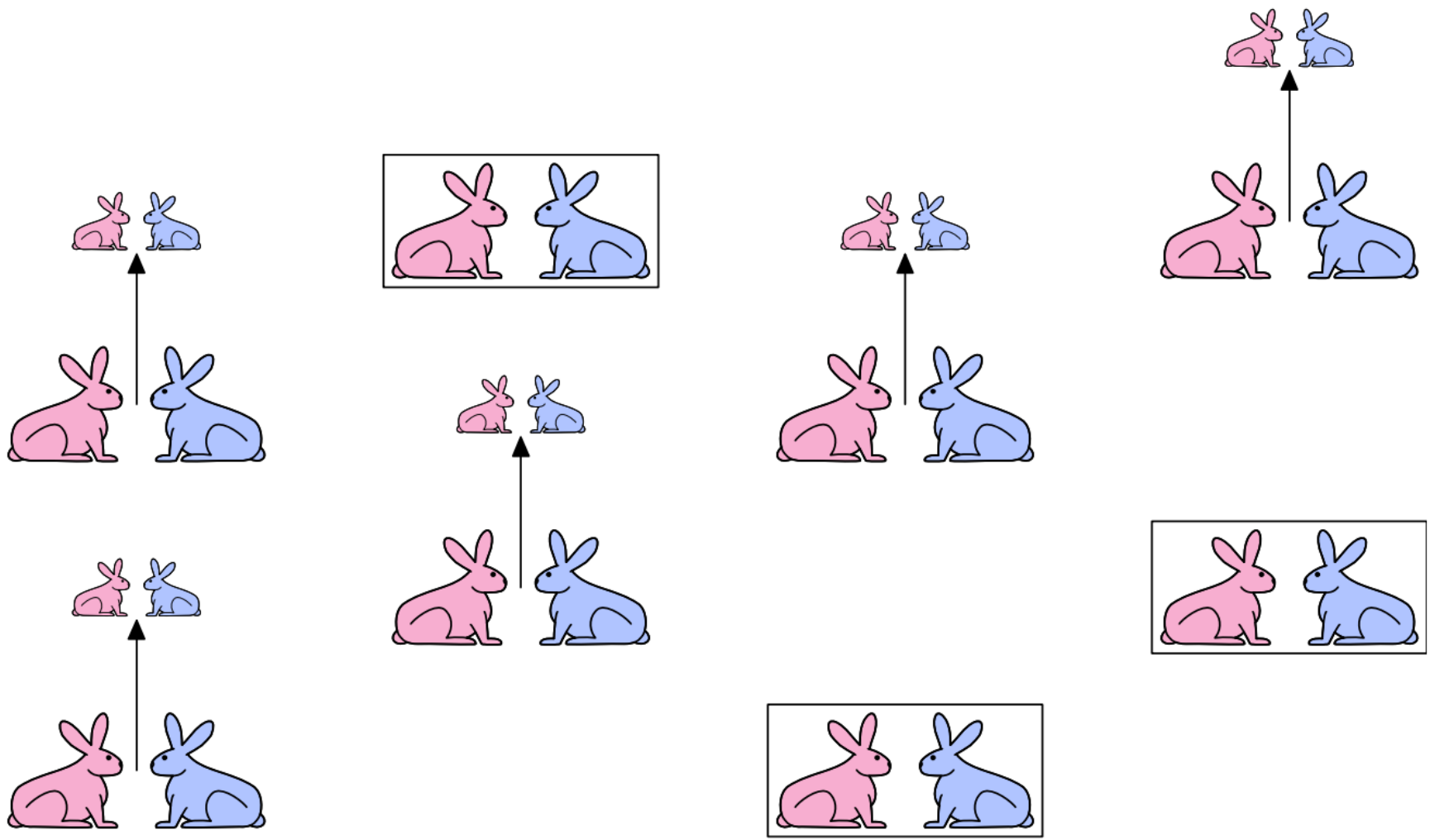
فیوناچی



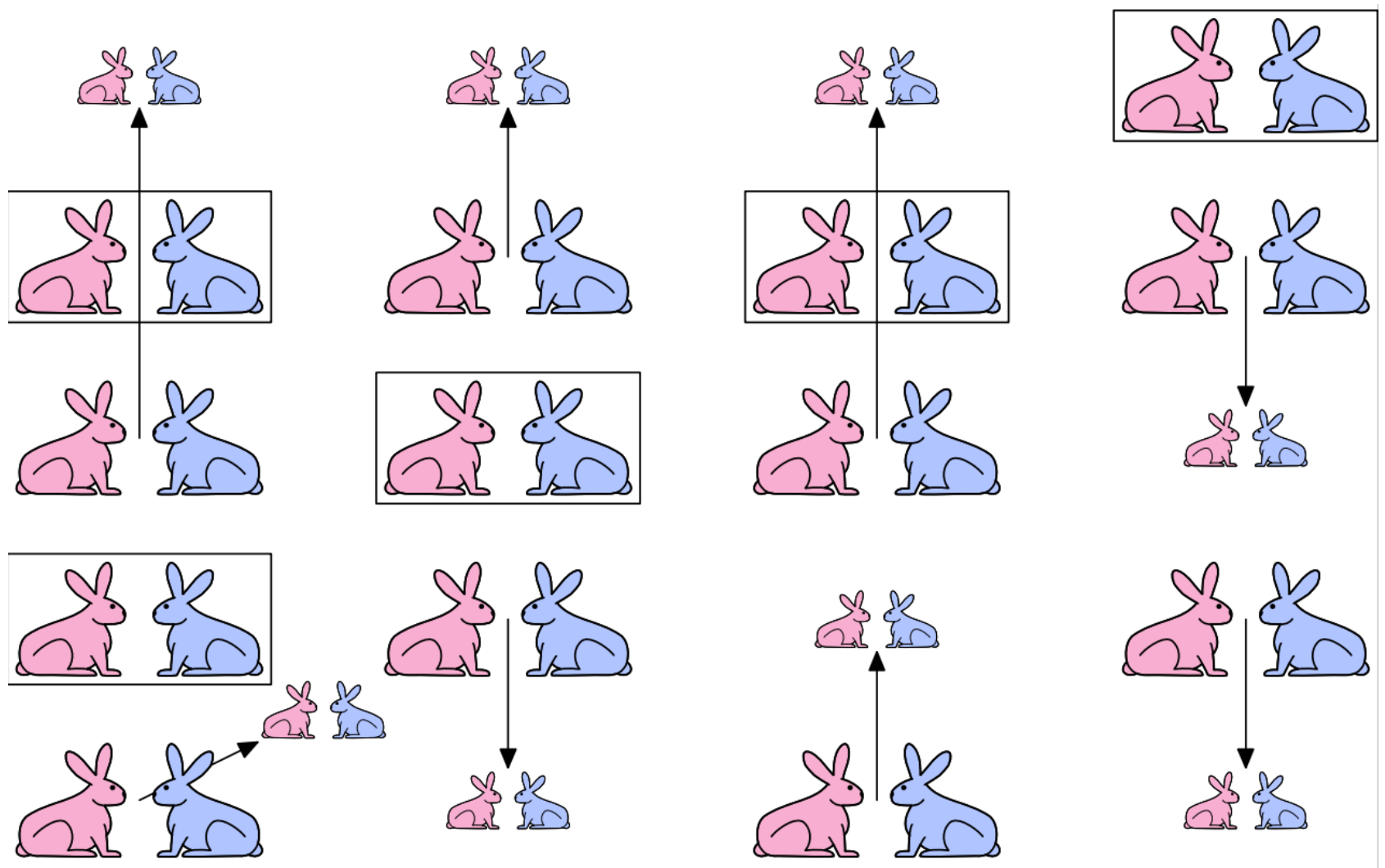
فیوناچی



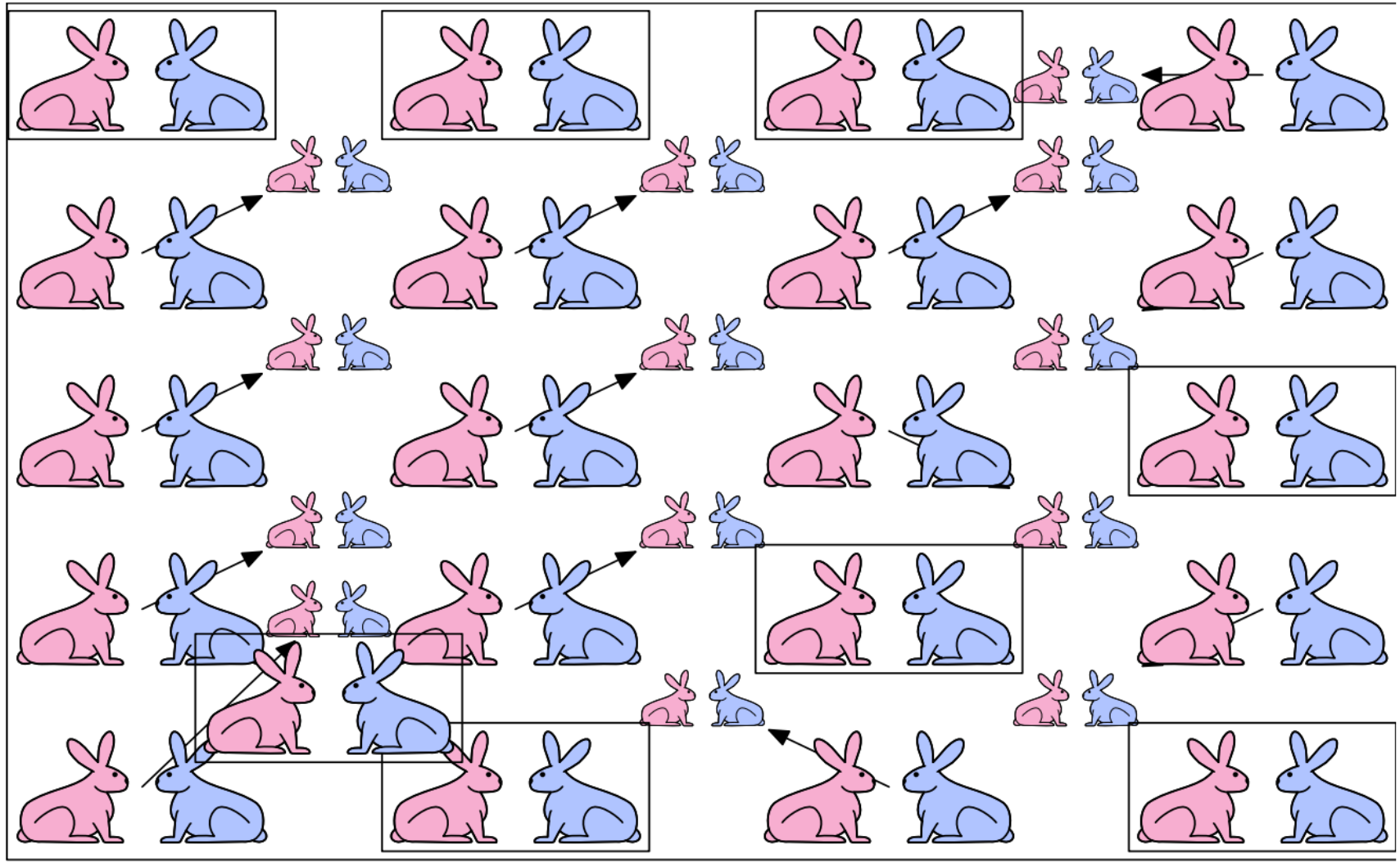
فیوناچی



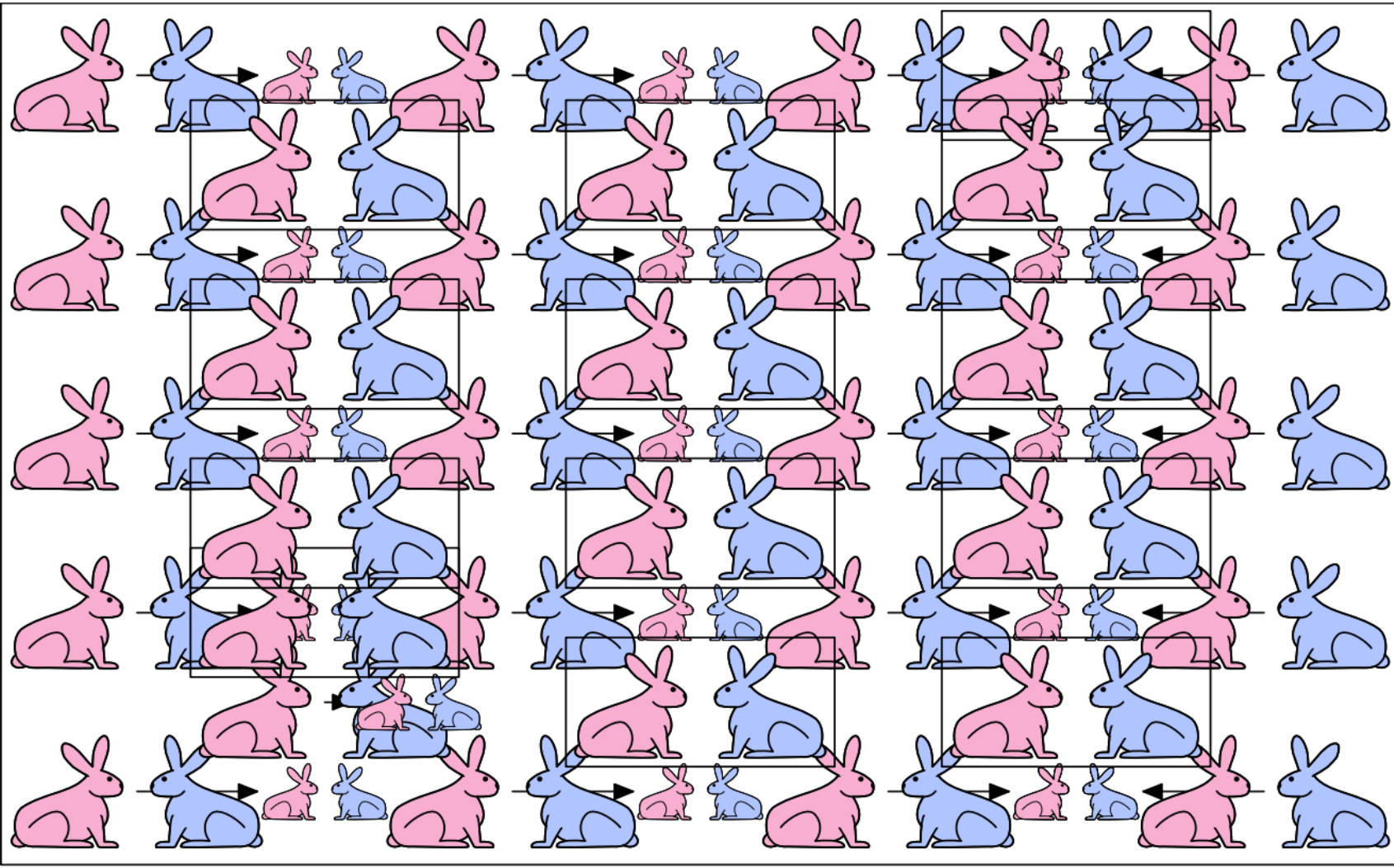
فیوناچی



فیوناچی



فیوناچی



فیوناچی

- بعد از یک ماه (ماه صفر نامگذاری می‌کنیم)
۱ مؤنث داریم.
- ماه یک هنوز ۱ خرگوش مؤنث داریم (که
هم‌اکنون باردار می‌باشد).
- در ماه دو، ۲ خرگوش مؤنث داریم (که یکی از
آنها باردار است).
- به صورت کلی تعداد مؤنث‌ها در ماه n ام
برابر است با :
 - $Females(n) = Females(n-1) + Females(n-2)$

ماه	مؤنث‌ها
0	1
1	1
2	2
3	3
4	5
5	8
6	13

فیوناچی

■ حالات پایه :

- $Females(0) = 1$
- $Females(1) = 1$

■ مرحله بازگشتی :

- $Females(n) = Females(n-1) + Females(n-2)$

```
def fib(x):  
    """assumes x an int >= 0  
        returns Fibonacci of x"""  
    if x == 0 or x == 1:  
        return 1  
    else:  
        return fib(x-1) + fib(x-2)
```


فراخوانی یک تابع از فایل

فرض کنید فایل circle.py شامل کد زیر می‌باشد :

```
pi = 3.14159
```

```
def area(radius):  
    return pi*(radius**2)
```

```
def circumference(radius):  
    return 2*pi*radius
```

فراخوانی یک تابع از فایل

برای فراخوانی و استفاده از آن می توان به صورت زیر عمل کرد

```
In [3]: import circle :
```

```
In [4]: pi = 3
```

```
In [5]: print(pi)  
3
```

```
In [6]: print(circle.pi)  
3.14159
```

```
In [7]: print(circle.area(3))  
28.27431
```

```
In [8]: print(circle.circumference(3))  
18.849539999999998
```

شیوه دیگر فراخوانی

اگر نخواهیم هنگام فراخوانی به طور مداوم از اسم فایل استفاده کنیم می‌توان به صورت زیر عمل کرد که کلیه میدان‌های درون فایل مورد نظر را به داخل حافظه فرا می‌خواند :

```
In [9]: from circle import *
```

```
In [10]: print(pi)  
3.14159
```

```
In [11]: print(area(3))  
28.27431
```



PYTHON

TUPLES, LISTS

TUPLE ها

- ترتیب مشخصی از المان‌ها که توانایی ترکیب انواع المان‌های دیگر را دارد.
- **تغییر ناپذیرند (immutable)** بدین معنی که مقادیر المان‌های آن غیرقابل تغییر می‌باشند.
- با پرانتز نمایش داده می‌شوند.

👉 TUPLE

```
In [12]: te = ()
```

```
In [13]: t = (2, "one", 3)
```

```
In [14]: t[0]
```

```
Out[14]: 2
```

```
In [15]: (2, "one", 3) + (5,6)
```

```
Out[15]: (2, 'one', 3, 5, 6)
```

```
In [16]: t[1:2]
```

```
Out[16]: ('one',)
```

```
In [17]: t[1:3]
```

```
Out[17]: ('one', 3)
```

```
In [18]: t[1]
```

```
Out[18]: 'one'
```

```
In [19]: t[1] = 4
```

```
Traceback (most recent call last):
```

```
File "<ipython-input-19-87b0f225887f>", line 1, in
```

```
<module>
```

```
    t[1] = 4
```

```
TypeError: 'tuple' object does not support item  
assignment
```

جابجا کردن متغیر با TUPLE

- از Tuple ها می‌توان برای جابجایی مقادیر متغیرها استفاده کرد :

```
x = y
```

```
y = x
```



```
temp = x
```

```
x = y
```

```
y = temp
```



```
(x, y) = (y, x)
```



گرفتن چند خروجی از تابع با Tuple

- با استفاده از Tuple ها می‌توان از تابع چند خروجی گرفت.

```
9 def quotient_and_remainder(x, y):  
10     q = x//y  
11     r = x % y  
12     return (q, r)  
13  
14  
15 (quot, rem) = quotient_and_remainder(15.3, 3)  
16
```


List ها

- توالی مرتبی از داده‌ها، که با اندیس قابل دسترسی است.
- لیست‌ها با علامت [] مشخص می‌شوند.
- لیست شامل المان‌هایی به صورت زیر است :
 - معمولا همگن هستند (مثلا همگی از type عدد صحیح)
 - می‌توانند type های متفاوتی را در خود جای دهند (معمول نیست)
- المان‌های لیست قابل تغییر (mutable) می‌باشند.

اندیس‌ها و ترتیب در List

```
In [5]: a_list = []
```

```
In [6]: b_list = [2, 'a', 4, True]
```

```
In [7]: L = [2, 1, 3]
```

```
In [8]: len(L)
```

```
Out[8]: 3
```

```
In [9]: L[0]
```

```
Out[9]: 2
```

```
In [10]: L[2] + 1
```

```
Out[10]: 4
```

```
In [11]: L[3]
```

```
Traceback (most recent call last):
```

```
File "<ipython-input-11-28c5e42e8527>", line 1, in <module>
```

```
L[3]
```

```
IndexError: list index out of range
```

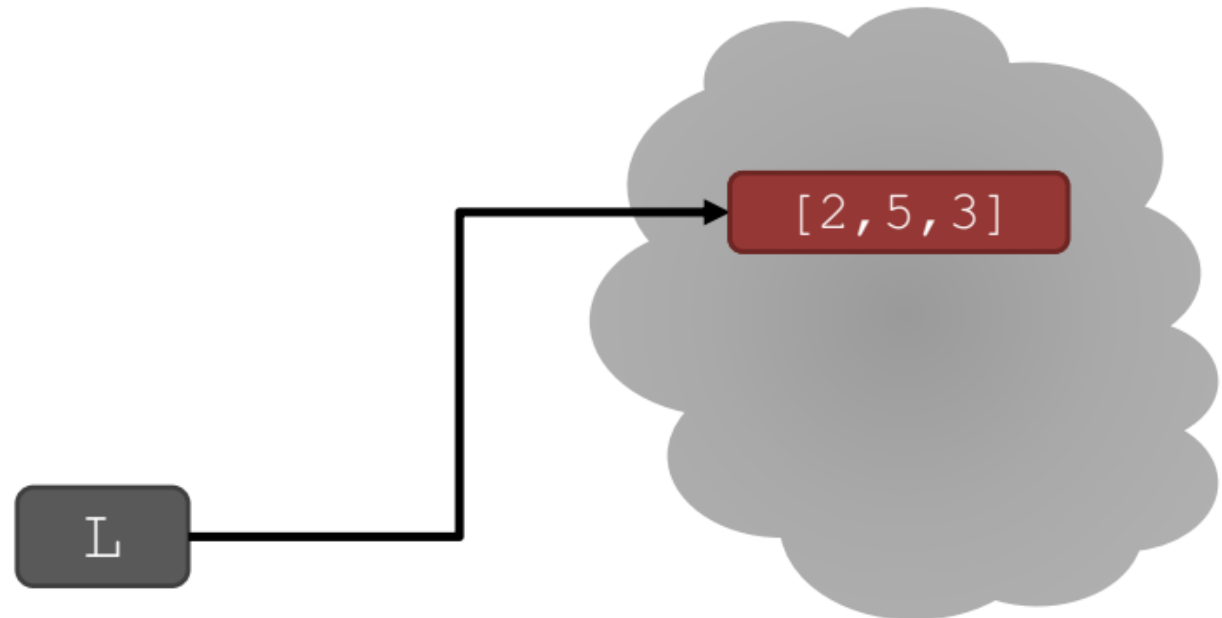
تغییر المان‌های لیست

```
In [12]: L = [2, 1, 3]
```

```
In [13]: L[1] = 5
```

```
In [14]: L
```

```
Out[14]: [2, 5, 3]
```



حلقه بر روی لیست

- به طور مثال اگر هدف بدست آوردن مجموع المان‌های یک لیست باشد می‌توانیم به دو روش زیر اقدام کنیم :

```
8 L = [4, 6, 1, 2]
9
10 total = 0
11 for i in range(len(L)):
12     total += L[i]
13 print(total)
14
```

```
8 L = [4, 6, 1, 2]
9
10 total = 0
11 for i in L:
12     total += i
13 print(total)
14
```

- توجه شود که المان‌های لیست از ۰ تا $\text{len}(L) - 1$ اندیس گذاری می‌شوند.
- $\text{range}(n)$ از ۰ تا $n-1$ ادامه دارد.

عملیات بر روی لیست‌ها

- می‌توانیم با دستور `L.append(element)` به انتهای لیست المانی را اضافه کنیم.
- این دستور لیست را تغییر می‌دهد. (`mutate`)

```
In [17]: L = [2, 1, 3]
```

```
In [18]: L.append(5)
```

```
In [19]: L
```

```
Out[19]: [2, 1, 3, 5]
```

عملیات بر روی لیست‌ها

- برای ترکیب لیست‌ها با هم می‌توانیم از `+` استفاده کنیم.
- می‌توانیم لیست را با دستور `L.extend(some_list)` ترکیب و تغییر دهیم (`mutate`).

```
In [19]: L
```

```
Out[19]: [2, 1, 3, 5]
```

```
In [20]: L1 = [2, 1, 3]
```

```
In [21]: L2 = [4, 5, 6]
```

```
In [22]: L3 = L1 + L2
```

```
In [23]: L1.extend([0, 6])
```

```
In [24]: L1
```

```
Out[24]: [2, 1, 3, 0, 6]
```

عملیات بر روی لیست‌ها

- با دستور `del (L[index])` می‌توانیم المانی از لیست را با اندیس مشخص حذف کنیم.
- با دستور `L.pop()` می‌توانیم المانی را از انتهای لیست حذف کنیم، همچنین خروجی دستور المان حذف شده می‌باشد!
- با دستور `L.remove(element)` می‌توانیم المان خاصی را حذف کنیم:
 - برای آن المان جستجو می‌کند و آن را حذف می‌کند.
 - اگر آن المان چند بار تکرار شده باشد، تنها اولین مورد آن حذف می‌شود.
 - اگر آن المان در لیست نباشد، با خطا مواجه می‌شویم.

عملیات بر روی لیست‌ها

■ دستورات زیر به ترتیب اجرا شوند.

```
In [27]: L  
Out[27]: [1, 3, 6, 3, 7, 0]
```

```
In [28]: L.remove(3)
```

```
In [29]: L  
Out[29]: [1, 6, 3, 7, 0]
```

```
In [30]: del(L[1])
```

```
In [31]: L  
Out[31]: [1, 3, 7, 0]
```

```
In [32]: L.pop()  
Out[32]: 0
```

```
In [33]: L  
Out[33]: [1, 3, 7]
```

■ همچنین شایان ذکر است که تمامی دستورات زیر باعث تغییر لیست می‌شوند.

تبدیل لیست‌ها به string و برعکس

- برای تبدیل string به list از دستور `list(s)` استفاده می‌شود، که خروجی آن یک لیست با تمامی کاراکترهای `s` به صورت جداگانه در لیست می‌باشد.
- از دستور `s.split()` برای تقسیم یک string از محل کاراکتر مشخص شده استفاده کرد. همچنین اگر کاراکتری مشخص نشود، از محل فاصله بین حروف string تقسیم می‌شود.
- از دستور `'.join(L)` برای تبدیل یک لیست از کاراکترها به string استفاده می‌شود.

تبدیل لیست‌ها به string و برعکس

```
In [1]: s = "Alireza, Ghorbi"
```

```
In [2]: list(s)
```

```
Out[2]: ['A', 'l', 'i', 'r', 'e', 'z', 'a', ',', ' ', 'G',  
'h', 'o', 'r', 'b', 'i']
```

```
In [3]: s.split(',')
```

```
Out[3]: ['Alireza', ' Ghorbi']
```

```
In [4]: s.split(' ')
```

```
Out[4]: ['Alireza,', 'Ghorbi']
```

```
In [5]: L = ['A', 'l', 'i']
```

```
In [6]: ''.join(L)
```

```
Out[6]: 'Ali'
```

```
In [7]: '_'.join(L)
```

```
Out[7]: 'A_l_i'
```

مرتب سازی و برعکس کردن لیست

- مرتب سازی با دستور `sorted()` و `sort()`
- برعکس کردن لیست با دستور `reverse()`
- `sort()` و `reverse()` لیست را تغییر (mutate) می‌دهند.

```
In [8]: L = [9, 6, 0, 3]
```

```
In [9]: sorted(L)
```

```
Out[9]: [0, 3, 6, 9]
```

```
In [10]: L.sort()
```

```
In [11]: L
```

```
Out[11]: [0, 3, 6, 9]
```

```
In [12]: L.reverse()
```

```
In [13]: L
```

```
Out[13]: [9, 6, 3, 0]
```

نام‌های مستعار (ALIASES)

```
In [14]: a = 1
```

```
In [15]: b = a
```

```
In [16]: print(a)  
1
```

```
In [17]: print(b)  
1
```

```
In [18]: c1 = ['red', 'yellow', 'blue']
```

```
In [19]: c2 = c1
```

```
In [20]: c2.append('green')
```

```
In [21]: print(c2)  
['red', 'yellow', 'blue', 'green']
```

```
In [22]: print(c1)  
['red', 'yellow', 'blue', 'green']
```

وقتی نام مستعاری برای یک متغیر تعریف شود، با تغییر متغیر اصلی یا مستعار آن، هر دو تغییر می‌کنند.

Clone (کپی کردن) یک لیست

```
In [23]: c1 = ['red', 'yellow', 'blue']
```

```
In [24]: c2 = c1[:]
```

```
In [25]: c2.append('green')
```

```
In [26]: print(c2)  
['red', 'yellow', 'blue', 'green']
```

```
In [27]: print(c1)  
['red', 'yellow', 'blue']
```

لیست در لیست در لیست ...

- می‌توانیم چندین لیست درون یکدیگر داشته باشیم.

```
In [6]: L1 = ['red', 'blue', 'green', 'yellow']
```

```
In [7]: L2 = ['grey', 'brown']
```

```
In [8]: L1.append(L2)
```

```
In [9]: L1
```

```
Out[9]: ['red', 'blue', 'green', 'yellow', ['grey', 'brown']]
```

```
In [10]: len(L1)
```

```
Out[10]: 5
```

مشکلات تغییر پذیر بودن لیست در حلقه

```
9 def remove_duplicate(L1, L2):
10     for el in L1:
11         if el in L2:
12             L1.remove(el)
13     return L1
14
15
16 def remove_duplicate_new(L1, L2):
17     L1_copy = L1[:]
18     for el in L1_copy:
19         if el in L2:
20             L1.remove(el)
21     return L1
22
23
24 L1 = [1, 2, 3, 4]
25 L2 = [1, 2, 5, 6]
26
27 print(remove_duplicate(L1, L2))
28 print(remove_duplicate_new(L1, L2))
29
```

- درون یک حلقه و هنگام تغییر یک لیست حواسمان به مشکل زیر باشد :

- خروجی تابع اول [2,3,4] و خروجی تابع دوم [3,4] می باشد.

- با بررسی گام به گام حلقه دلیل را متوجه شوید.



PYTHON

DICTIONARY

دیکشنری‌ها

■ از دو قسمت تشکیل شده و با علامت { } نشان داده می‌شود :

■ کلید

■ مقدار

```
In [13]: D = {}
```

```
In [14]: type(D)
```

```
Out[14]: dict
```

```
In [15]: grades = {'Ali': 'A+', 'Hamid': 'C', 'Sara': 'B', 'Amin': 'A'}
```

```
In [16]: grades['Ali']
```

```
Out[16]: 'A+'
```

```
In [17]: grades['Reza']
```

```
Traceback (most recent call last):
```

```
File "<ipython-input-17-65c69419160e>", line 1, in <module>
    grades['Reza']
```

```
KeyError: 'Reza'
```

عملیات بر روی دیکشنری‌ها

- اضافه کردن یک داده به دیکشنری :

```
In [18]: grades['Reza'] = 'D'
```

```
In [19]: grades
```

```
Out[19]: {'Ali': 'A+', 'Amin': 'A', 'Hamid': 'C', 'Reza': 'D', 'Sara': 'B'}
```

- تست برای بودن یک مقدار در دیکشنری :

```
In [20]: 'Sara' in grades
```

```
Out[20]: True
```

```
In [21]: 'Nima' in grades
```

```
Out[21]: False
```

- پاک کردن یک مقدار از دیکشنری :

```
In [22]: del(grades['Reza'])
```

```
In [23]: grades
```

```
Out[23]: {'Ali': 'A+', 'Amin': 'A', 'Hamid': 'C', 'Sara': 'B'}
```

عملیات بر روی دیکشنری‌ها

- لیست کلیدهای یک دیکشنری (بدون ترتیب خاص):

```
In [24]: grades.keys()
```

```
Out[24]: dict_keys(['Ali', 'Hamid', 'Sara', 'Amin'])
```

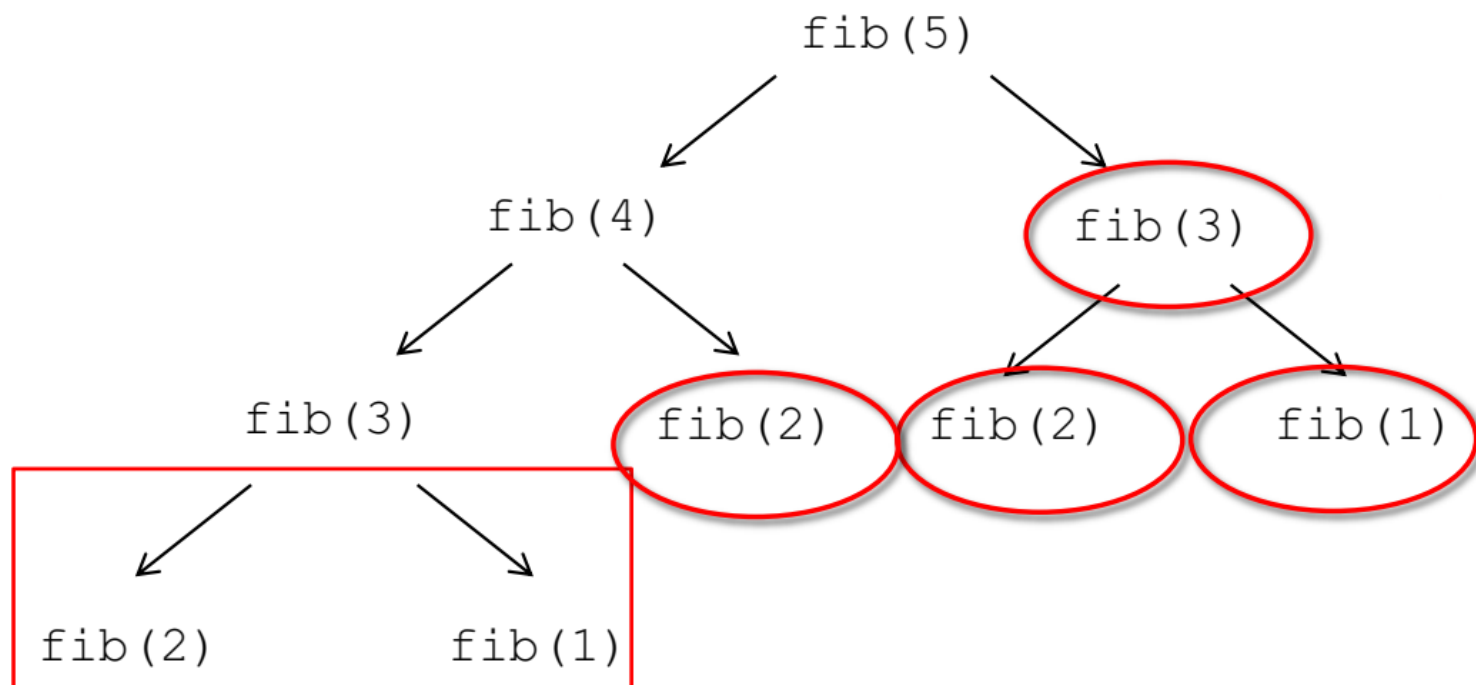
- لیست کلیدهای مقادیر در دیکشنری (بدون ترتیب خاص):

```
In [25]: grades.values()
```

```
Out[25]: dict_values(['A+', 'C', 'B', 'A'])
```

الگوریتم غیر بهینه فیوناچی

- مشکل : محاسبه مجدد مقادیری که از قبل چندین بار محاسبه شده‌اند.
- راه حل : ذخیره محاسبات انجام شده و فراخوانی آن در صورت نیاز



base cases

الگوریتم فیبوناچی بدون دیکشنری و محاسبه تعداد مراتب فراخوانی تابع

- در این حالت اگر $f = 35$ وارد شود، تعداد مراتب فراخوانی تابع fib برابر است با 18454929 !!

```
9 f = 35
10 numFibCall = 0
11
12
13 def fib(n):
14     global numFibCall
15     numFibCall += 1
16     if n == 1 or n == 2:
17         return 1
18     else:
19         return fib(n-1) + fib(n-2)
20
21
22 print(fib(f))
23 print(numFibCall)
```

الگوریتم فیبوناچی با دیکشنری و محاسبه تعداد مراتب فراخوانی تابع

- گشتن دنبال مقادیر محاسبه شده قبلی در دیکشنری
- اضافه کردن مقادیر محاسبه شده جدید به دیکشنری
- در این حالت وقتی $f=35$ باشد، تعداد مراتب فراخوانی تابع برابر است با ۷۶، که بسیار سریع تر از حالت بدون دیکشنری است.

```
26 numFibCall = 0
27
28
29 def fib_eff(n, d):
30     global numFibCall
31     numFibCall += 1
32     if n in d:
33         return d[n]
34     else:
35         res = fib_eff(n-1, d) + fib_eff(n-2, d)
36         d[n] = res
37         return res
38
39
40 d = {1: 1, 2: 1}
41 print(fib_eff(f, d))
42 print(numFibCall)
43
```

متغیر global

- زمانی که نیاز داشته باشیم که یک متغیر از بیرون و درون تابع قابل دسترسی باشد می‌توانیم آن را همانند کد مطرح شده در اسلاید قبل به صورت global تعریف کرد و مقدار آن در میدان تابع و هم میدان کلی قابل دسترس خواهد شد.
- `global numFibCall`

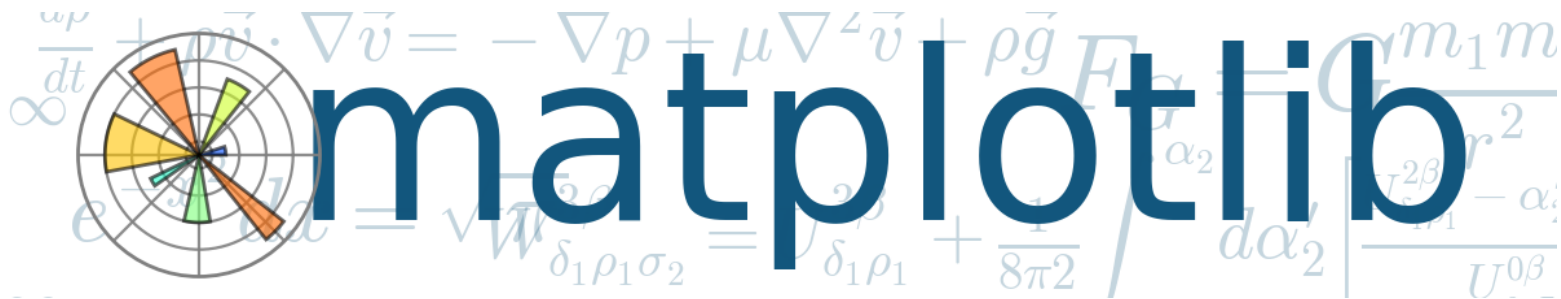
PYTHON

پلات و نمایش داده
ها با PYLAB

فراخوانی PYLAB

- برای فراخوانی مخزن PYLAB به محیط پایتون می‌توانیم به صورت زیر عمل کنیم :

```
import pylab as plt
```



مثال ساده

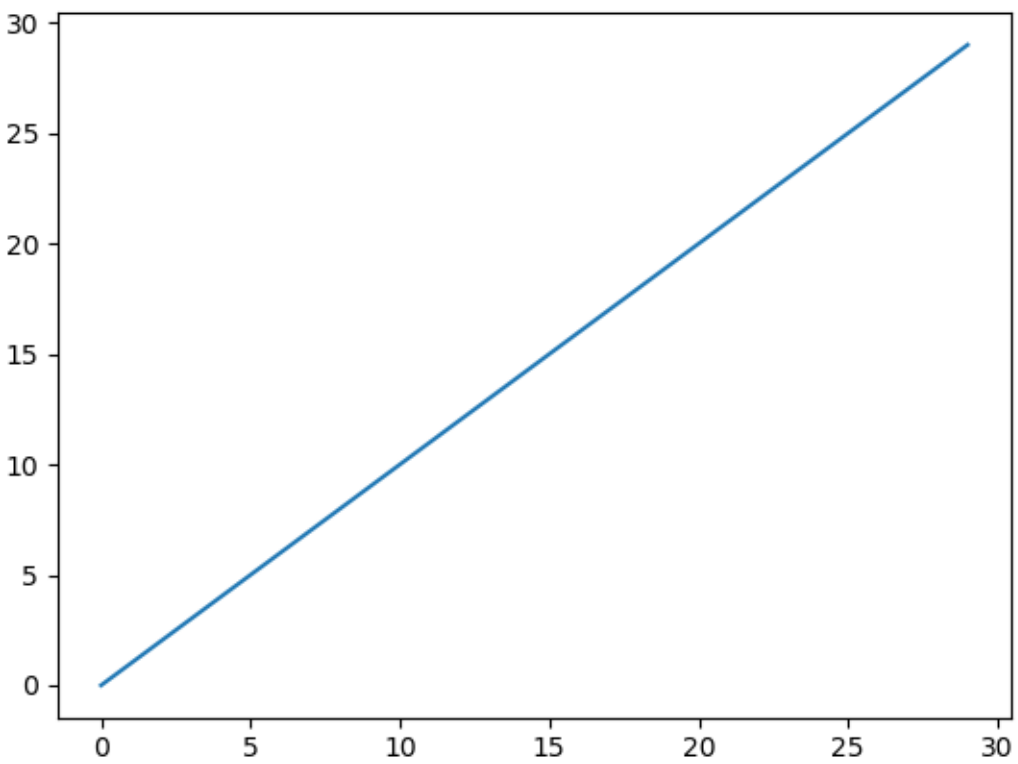
- ترسیم و پلات دو لیست به عنوان x و y
- ابتدا یک سری داده تولید می‌کنیم :

```
10 samples = []
11 slinear = []
12 squadratic = []
13 scubic = []
14 sexpo = []
15
16 for i in range(0, 30):
17     samples.append(i)
18     slinear.append(i)
19     squadratic.append(i**2)
20     scubic.append(i**3)
21     sexpo.append(2**i)
22
```

مثال ساده

- برای ترسیم پلات به عنوان مثال داریم :
- `plt.plot(samples, slinear)`
- در حال حاضر داده‌ها به صورت لیست می‌باشند.
- لیست‌ها باید هم طول باشند.

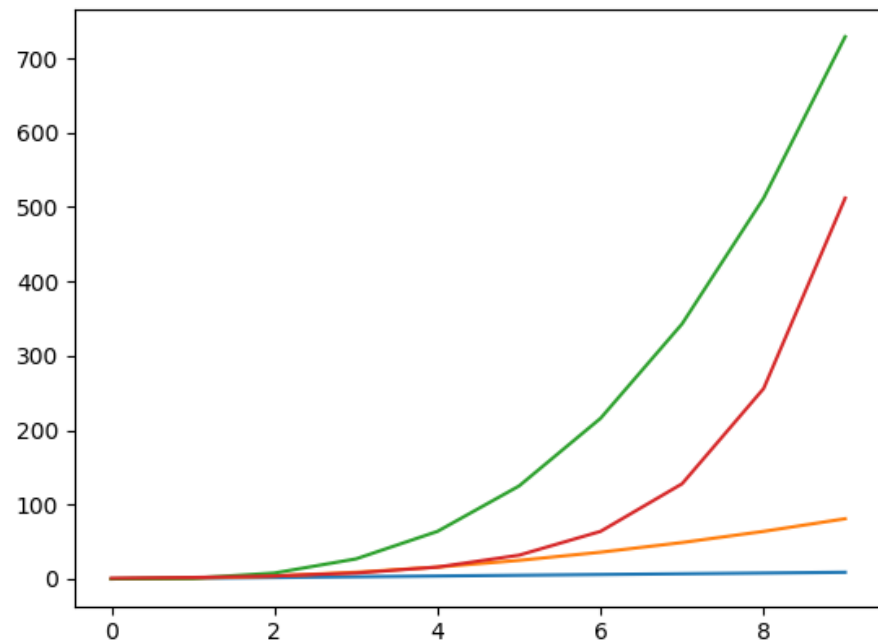
نمونه پلات اسلايد قبل



نمایش چند نمودار در یک شکل

برای این کار به طور پی در پی
از دستورات پلات استفاده
می‌کنیم :

```
8 import pylab as plt
9
10 samples = []
11 slinear = []
12 squadratic = []
13 scubic = []
14 sexpo = []
15
16 for i in range(0, 10):
17     samples.append(i)
18     slinear.append(i)
19     squadratic.append(i**2)
20     scubic.append(i**3)
21     sexpo.append(2**i)
22
23
24 plt.figure('All')
25 plt.plot(samples, slinear)
26 plt.plot(samples, squadratic)
27 plt.plot(samples, scubic)
28 plt.plot(samples, sexpo)
29
```



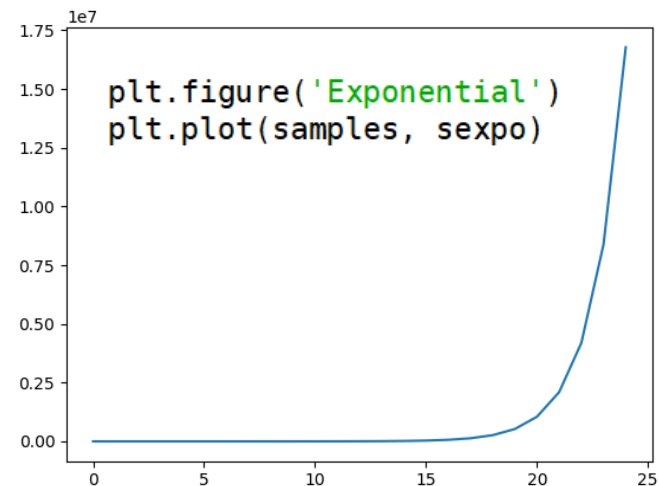
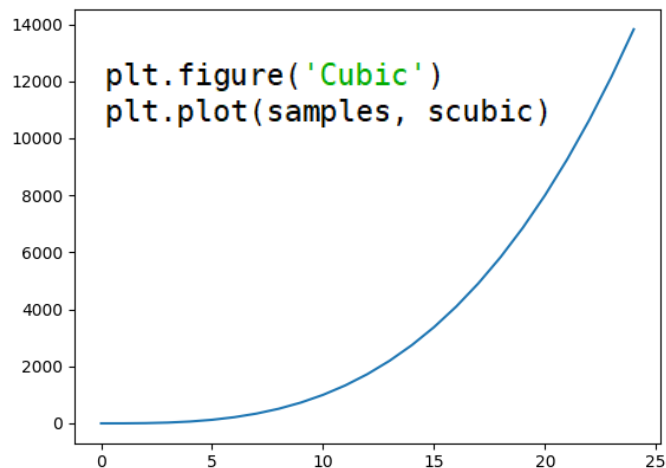
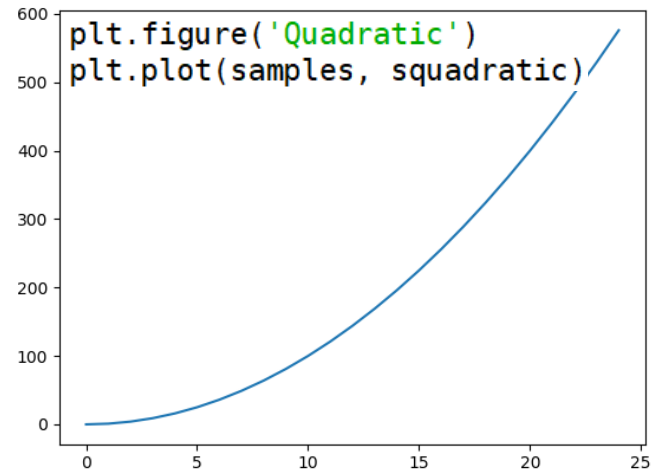
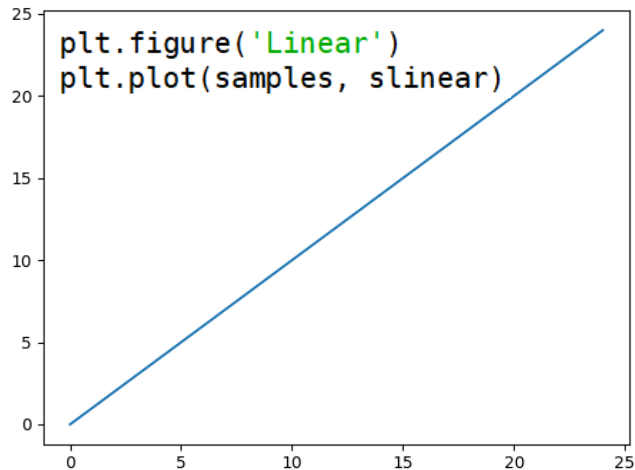
نمایش چند نمودار در شکل‌های جداگانه

- همانطور که در اسلاید قبل دیدیم، به دلیل مقیاس‌های متفاوت منحنی خطی و مرتبه ۲ به خوبی مشخص نیستند و هر چه تعداد داده‌ها بیشتر و بزرگتر شود این مشکل بیشتر به چشم می‌آید.
- در نتیجه نیازمند آن هستیم که نمودارها را در شکل‌های جداگانه ترسیم کنیم.
- با استفاده از دستور `plt.figure(<arg>)` می‌توانیم شکل جدیدی را تولید کنیم و پلات جدید را در آن ترسیم کنیم.

نمایش چند نمودار در شکل‌های جداگانه

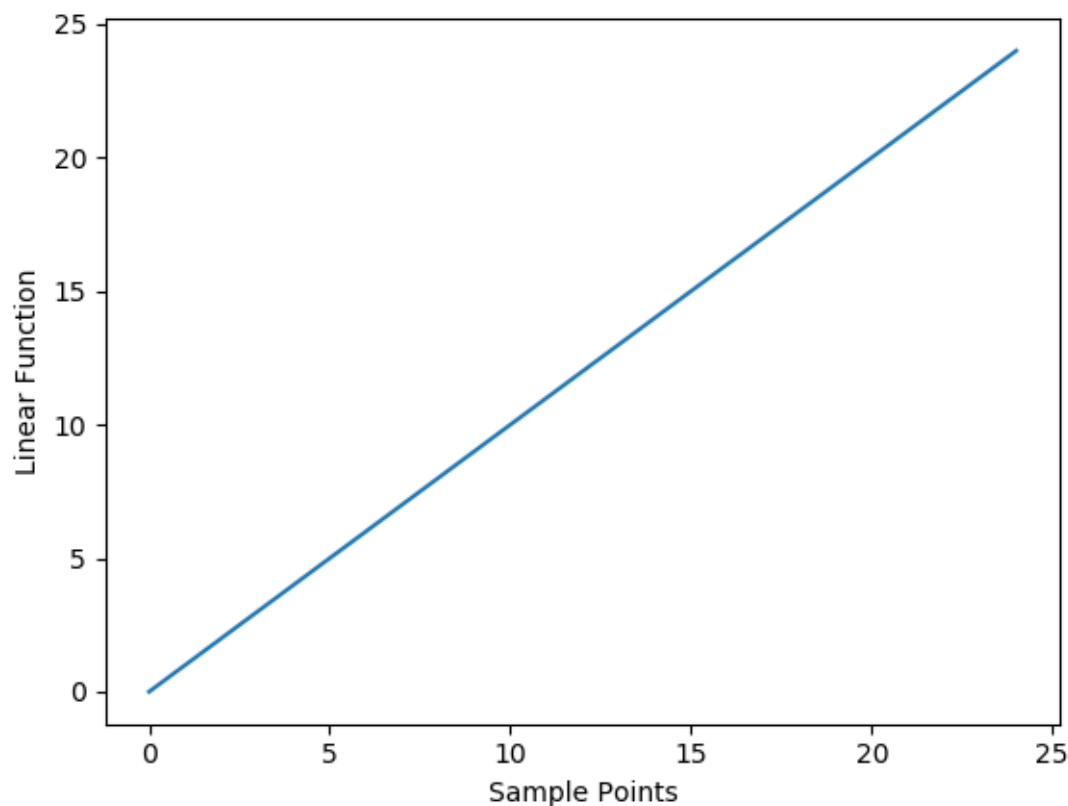
```
8 import pylab as plt
9
10 samples = []
11 slinear = []
12 squadratic = []
13 scubic = []
14 sexpo = []
15
16 for i in range(0, 25):
17     samples.append(i)
18     slinear.append(i)
19     squadratic.append(i**2)
20     scubic.append(i**3)
21     sexpo.append(2**i)
22
23
24 plt.figure('Linear')
25 plt.plot(samples, slinear)
26 plt.figure('Quadratic')
27 plt.plot(samples, squadratic)
28 plt.figure('Cubic')
29 plt.plot(samples, scubic)
30 plt.figure('Exponential')
31 plt.plot(samples, sexpo)
32
```

نمونه پلات کد اسلاید قبل



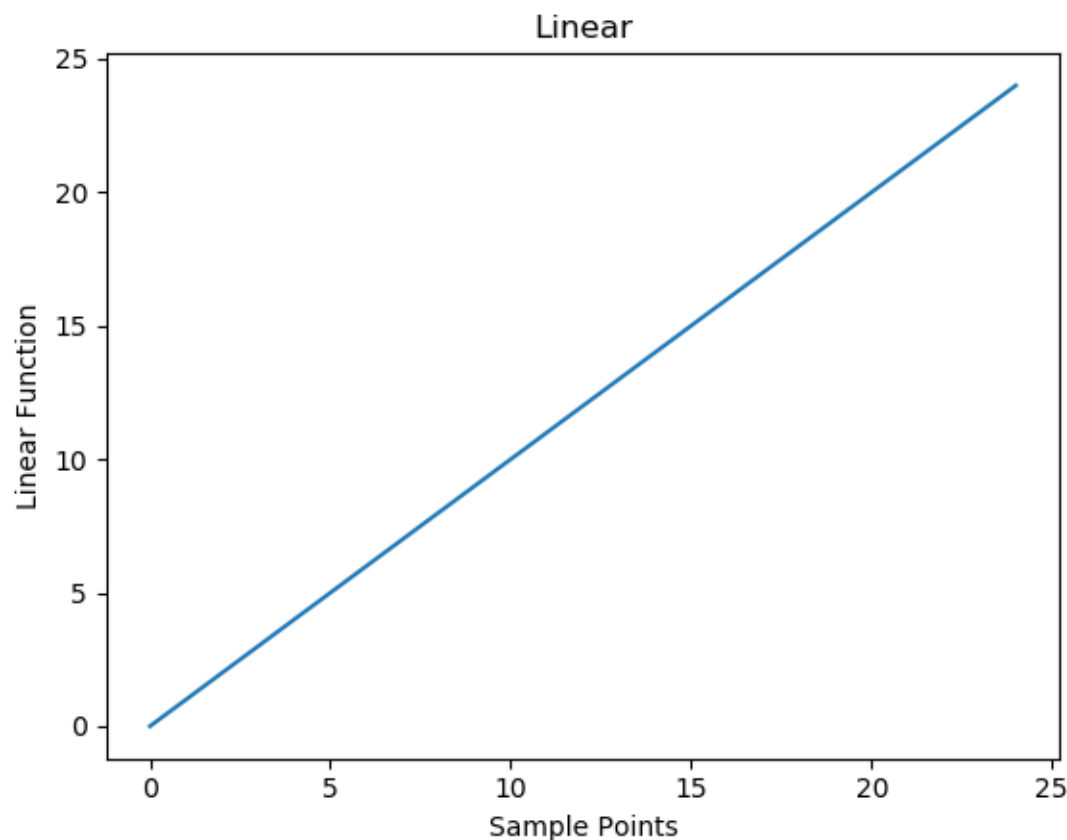
تعریف عنوان برای محورها (xlabel, ylabel)

```
plt.figure('Linear')  
plt.xlabel('Sample Points')  
plt.ylabel('Linear Function')  
plt.plot(samples, slinear)
```



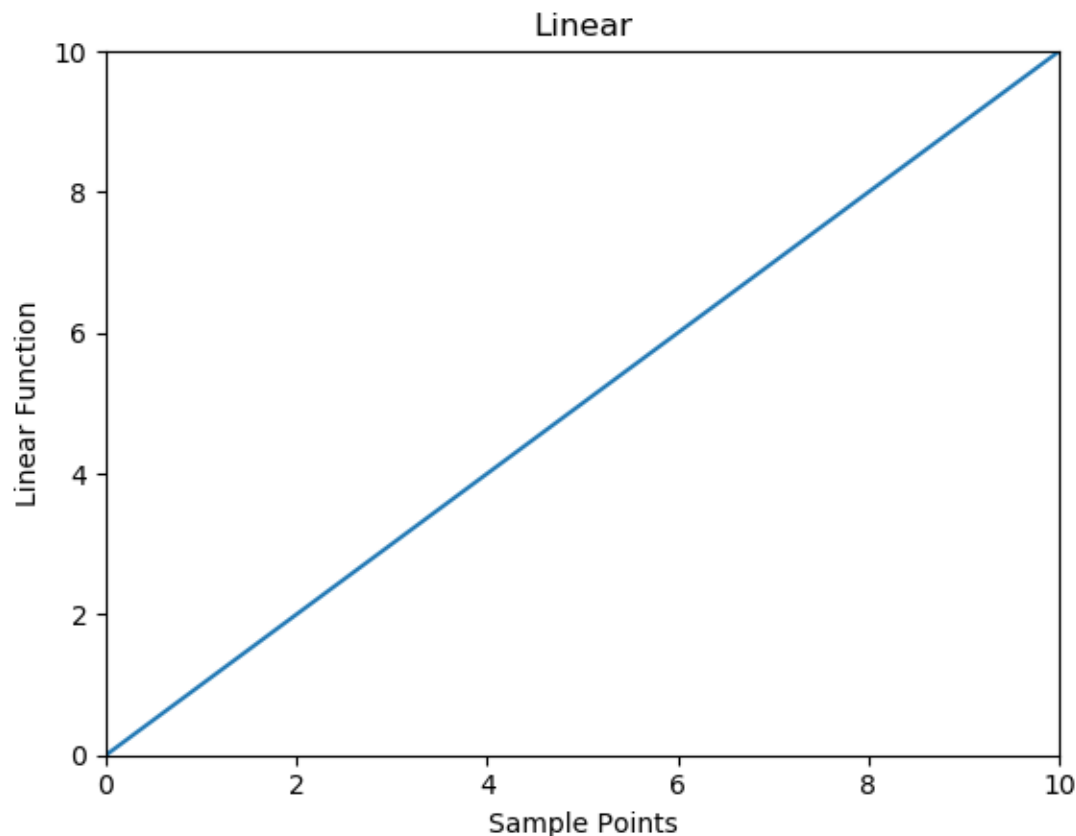
تعریف عنوان برای نمودار (title)

```
plt.figure('Linear')  
plt.title('Linear')  
plt.xlabel('Sample Points')  
plt.ylabel('Linear Function')  
plt.plot(samples, slinear)
```



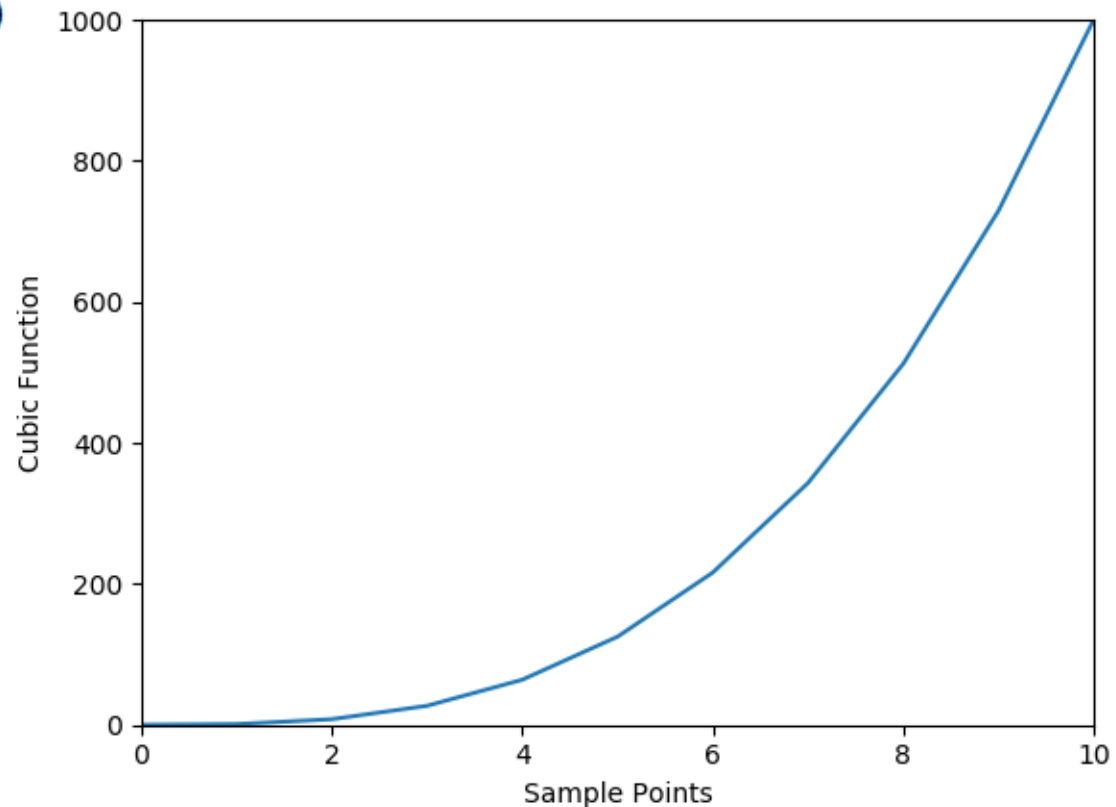
محدود کردن محورها (xlim, ylim)

```
plt.figure('Linear')  
plt.title('Linear')  
plt.xlabel('Sample Points')  
plt.ylabel('Linear Function')  
plt.xlim(0, 10)  
plt.ylim(0, 10)  
plt.plot(samples, slinear)
```



محدود کردن محورها (xlim, ylim)

```
plt.figure('Cubic')  
plt.xlabel('Sample Points')  
plt.ylabel('Cubic Function')  
plt.xlim(0, 10)  
plt.ylim(0, 1000)  
plt.plot(samples, scubic)
```



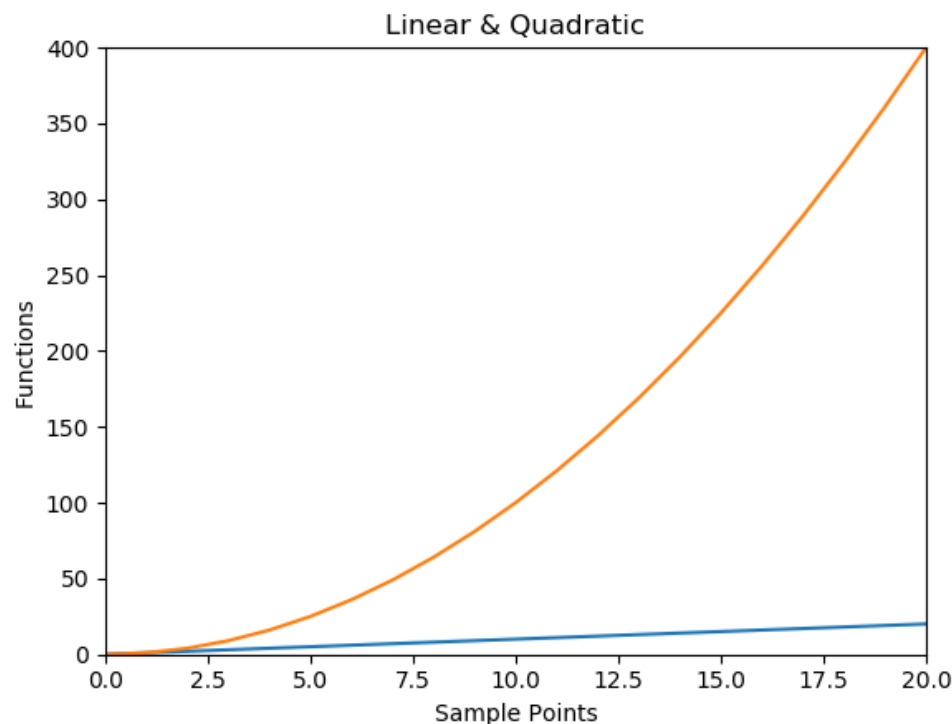
جدا کردن پلات‌ها و ترکیب آن‌ها

■ برای جدا کردن پلات‌ها و ترسیم شکل جدید می‌توانیم از دستور `plt.figure(<arg>)` استفاده کنیم.

```

8 import pylab as plt
9
10 samples = []
11 slinear = []
12 squadratic = []
13 scubic = []
14 sexpo = []
15
16 for i in range(0, 100):
17     samples.append(i)
18     slinear.append(i)
19     squadratic.append(i**2)
20     scubic.append(i**3)
21     sexpo.append(2**i)
22
23
24 plt.figure('Linear & Quadratic')
25 plt.title('Linear & Quadratic')
26 plt.xlabel('Sample Points')
27 plt.ylabel('Functions')
28 plt.xlim(0, 20)
29 plt.ylim(0, 400)
30 plt.plot(samples, slinear)
31 plt.plot(samples, squadratic)
32
33
34 plt.figure('Cubic & Exponential')
35 plt.title('Cubic & Exponential')
36 plt.xlabel('Sample Points')
37 plt.ylabel('Function')
38 plt.xlim(0, 20)
39 plt.ylim(0, 10000)
40 plt.plot(samples, scubic)
41 plt.plot(samples, sexpo)
42

```



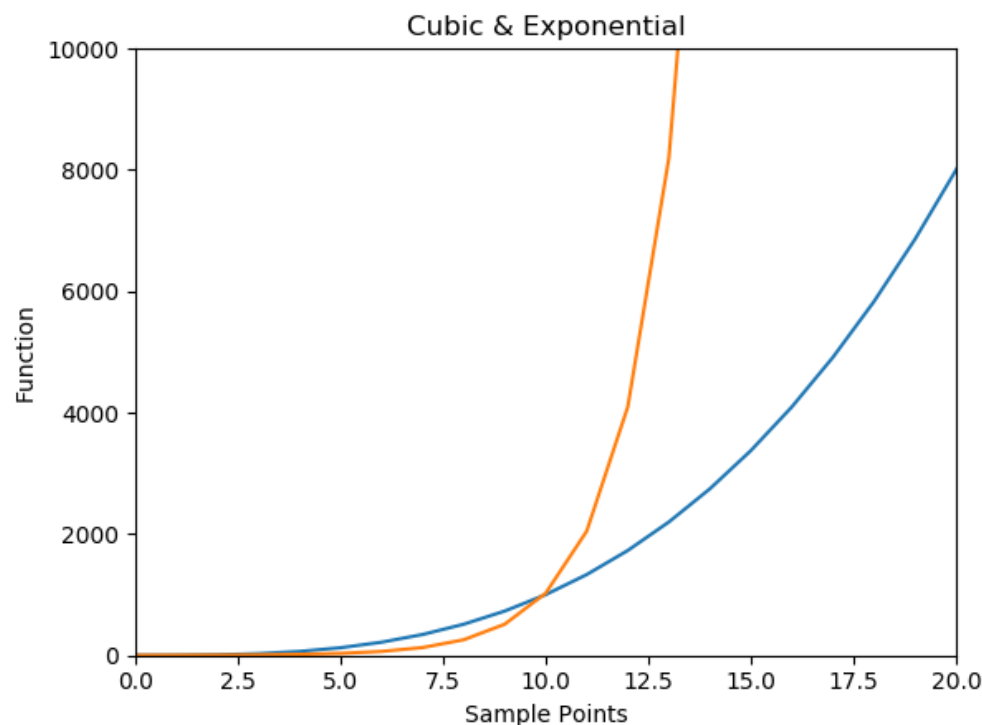
جدا کردن پلات‌ها و ترکیب آن‌ها

■ برای جدا کردن پلات‌ها و ترسیم شکل جدید می‌توانیم از دستور `plt.figure(<arg>)` استفاده کنیم.

```

8 import pylab as plt
9
10 samples = []
11 slinear = []
12 squadratic = []
13 scubic = []
14 sexpo = []
15
16 for i in range(0, 100):
17     samples.append(i)
18     slinear.append(i)
19     squadratic.append(i**2)
20     scubic.append(i**3)
21     sexpo.append(2**i)
22
23
24 plt.figure('Linear & Quadratic')
25 plt.title('Linear & Quadratic')
26 plt.xlabel('Sample Points')
27 plt.ylabel('Functions')
28 plt.xlim(0, 20)
29 plt.ylim(0, 400)
30 plt.plot(samples, slinear)
31 plt.plot(samples, squadratic)
32
33
34 plt.figure('Cubic & Exponential')
35 plt.title('Cubic & Exponential')
36 plt.xlabel('Sample Points')
37 plt.ylabel('Function')
38 plt.xlim(0, 20)
39 plt.ylim(0, 10000)
40 plt.plot(samples, scubic)
41 plt.plot(samples, sexpo)
42

```



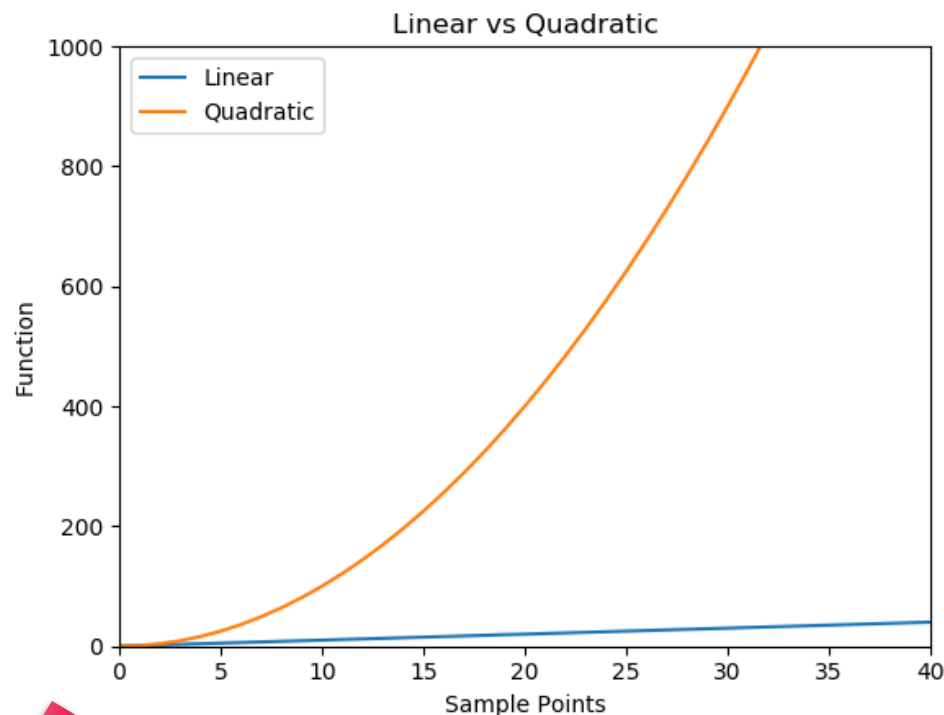
اضافه کردن راهنما به پلات

دستور	محل
'best'	بهترین مکان اتوماتیک
'upper right'	بالا راست
'upper left'	بالا چپ
'lower left'	پایین چپ
'lower right'	پایین راست
'right'	راست
'left'	چپ
'center left'	وسط چپ
'center right'	وسط راست
'lower center'	پایین وسط
'upper center'	بالا وسط
'center'	وسط

- برای اضافه کردن راهنما به پلات و شکل می توانیم از دستور `plt.legend(<arg>)` استفاده کنیم.
- برای اینکار ابتدا لازم است که برای هر پلات یک برچسب (Label) ایجاد کنیم (مطابق کد روبرو)
- مکان راهنما را می توان به صورت جدول روبرو مشخص کرد.

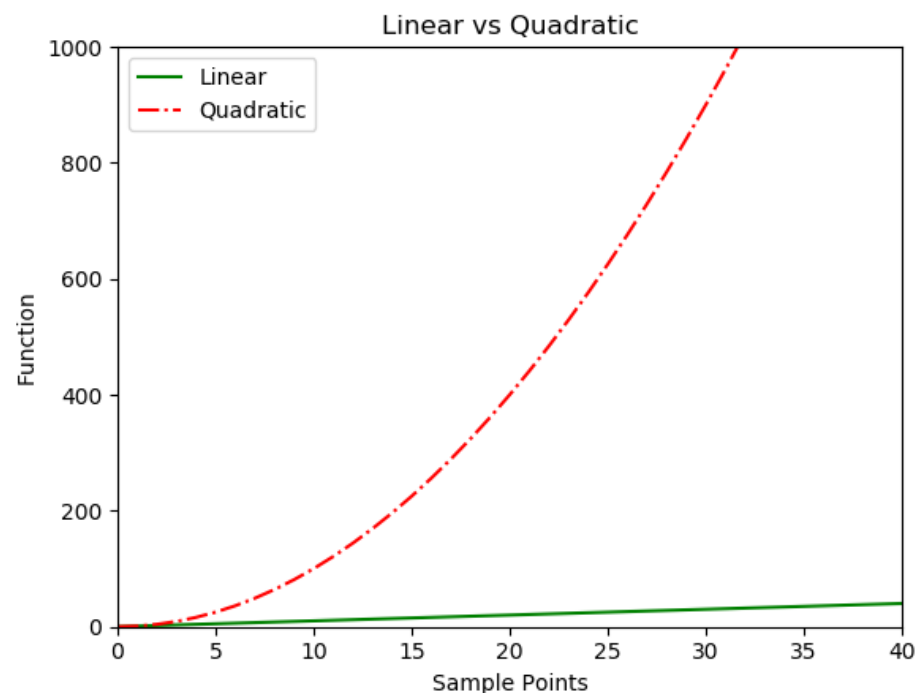
اضافه کردن راهنما به پلات

```
8 import pylab as plt
9
10 samples = []
11 slinear = []
12 squadratic = []
13
14 for i in range(0, 50):
15     samples.append(i)
16     slinear.append(i)
17     squadratic.append(i**2)
18
19
20 plt.figure('Linear vs Quadratic')
21 plt.title('Linear vs Quadratic')
22 plt.xlabel('Sample Points')
23 plt.ylabel('Function')
24 plt.xlim(0, 40)
25 plt.ylim(0, 1000)
26
27 plt.plot(samples, slinear, label='Linear')
28 plt.plot(samples, squadratic, label='Quadratic')
29 plt.legend(loc='upper left')
30
```



تغییر رنگ خطوط و استایل خط

```
8 import pylab as plt
9
10 samples = []
11 slinear = []
12 squadratic = []
13
14 for i in range(0, 50):
15     samples.append(i)
16     slinear.append(i)
17     squadratic.append(i**2)
18
19
20 plt.figure('Linear vs Quadratic')
21 plt.title('Linear vs Quadratic')
22 plt.xlabel('Sample Points')
23 plt.ylabel('Function')
24 plt.xlim(0, 40)
25 plt.ylim(0, 1000)
26
27
28 plt.plot(samples, slinear, 'g-', label='Linear')
29 plt.plot(samples, squadratic, 'r-.', label='Quadratic')
30 plt.legend(loc='upper left')
31
```



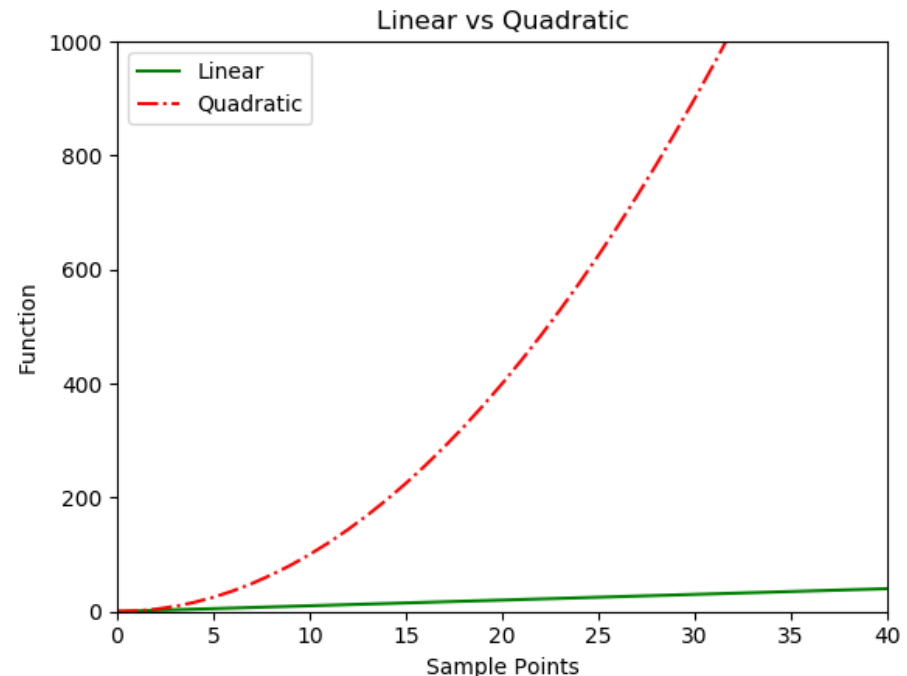
character	description
' _ '	solid line style
' - - '	dashed line style
' - . '	dash-dot line style
' : '	dotted line style
' . '	point marker
' , '	pixel marker
' o '	circle marker
' v '	triangle_down marker
' ^ '	triangle_up marker
' < '	triangle_left marker
' > '	triangle_right marker
' 1 '	tri_down marker
' 2 '	tri_up marker
' 3 '	tri_left marker
' 4 '	tri_right marker
' s '	square marker
' p '	pentagon marker
' * '	star marker
' h '	hexagon1 marker
' H '	hexagon2 marker
' + '	plus marker
' x '	x marker
' D '	diamond marker
' d '	thin_diamond marker
' '	vline marker
' _ '	hline marker

جداول رنگ و استایل خطوط

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

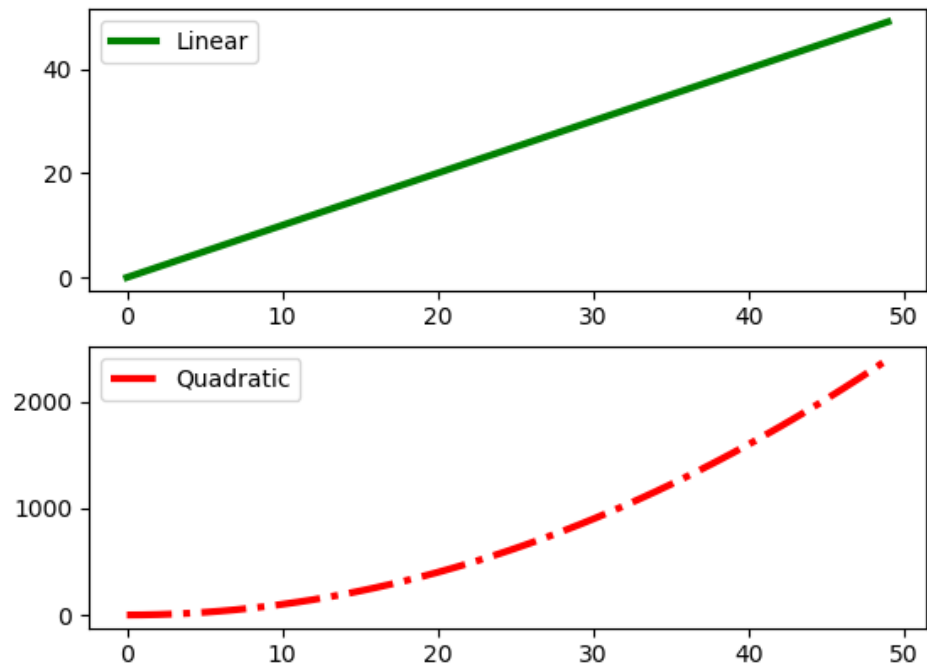
تغيير ضخامت خطوط

```
8 import pylab as plt
9
10 samples = []
11 slinear = []
12 squadratic = []
13
14 for i in range(0, 50):
15     samples.append(i)
16     slinear.append(i)
17     squadratic.append(i**2)
18
19
20 plt.figure('Linear vs Quadratic')
21 plt.title('Linear vs Quadratic')
22 plt.xlabel('Sample Points')
23 plt.ylabel('Function')
24 plt.xlim(0, 40)
25 plt.ylim(0, 1000)
26
27
28 plt.plot(samples, slinear, 'g-', label='Linear', linewidth=3.0)
29 plt.plot(samples, squadratic, 'r-.', label='Quadratic', linewidth=3.0)
30 plt.legend(loc='upper left')
31
```



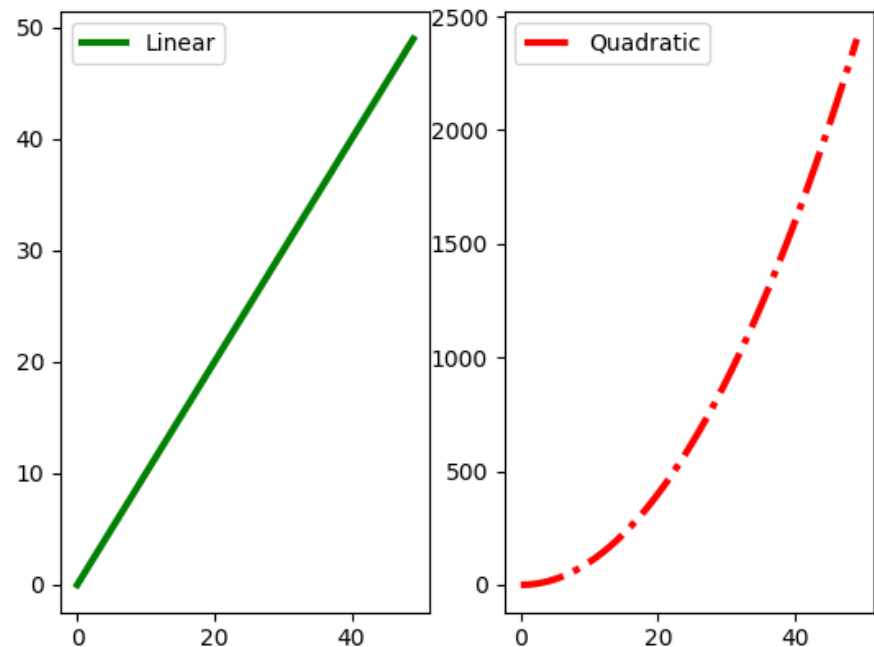
زیر پلات

```
8 import pylab as plt
9
10 samples = []
11 slinear = []
12 squadratic = []
13
14 for i in range(0, 50):
15     samples.append(i)
16     slinear.append(i)
17     squadratic.append(i**2)
18
19
20 plt.figure('Linear vs Quadratic')
21 plt.title('Linear vs Quadratic')
22 plt.xlabel('Sample Points')
23 plt.ylabel('Function')
24 plt.xlim(0, 40)
25 plt.ylim(0, 1000)
26
27 plt.subplot(211)
28 plt.plot(samples, slinear, 'g-', label='Linear', linewidth=3.0)
29 plt.legend()
30 plt.subplot(212)
31 plt.plot(samples, squadratic, 'r-.', label='Quadratic', linewidth=3.0)
32 plt.legend()
33
```



زیر پلات

```
8 import pylab as plt
9
10 samples = []
11 slinear = []
12 squadratic = []
13
14 for i in range(0, 50):
15     samples.append(i)
16     slinear.append(i)
17     squadratic.append(i**2)
18
19
20 plt.figure('Linear vs Quadratic')
21 plt.title('Linear vs Quadratic')
22 plt.xlabel('Sample Points')
23 plt.ylabel('Function')
24 plt.xlim(0, 40)
25 plt.ylim(0, 1000)
26
27 plt.subplot(121)
28 plt.plot(samples, slinear, 'g-', label='Linear', linewidth=3.0)
29 plt.legend()
30 plt.subplot(122)
31 plt.plot(samples, squadratic, 'r-.', label='Quadratic', linewidth=3.0)
32 plt.legend()
33
```





PYTHON

Math Library

فراخوانی کتابخانه math

- برای فراخوانی کتابخانه math به محیط پایتون می‌توانیم به صورت زیر عمل کنیم :

```
import math
```

ثابت‌ها (عدد پی و نیپر)

```
In [2]: import math
```

```
In [3]: math.pi
```

```
Out[3]: 3.141592653589793
```

```
In [4]: math.e
```

```
Out[4]: 2.718281828459045
```


قدر مطلق

```
In [17]: math.fabs(-9)
```

```
Out[17]: 9.0
```

```
In [18]: type(math.fabs(-9))
```

```
Out[18]: float
```

```
In [19]: abs(-9)
```

```
Out[19]: 9
```

```
In [20]: type(abs(-9))
```

```
Out[20]: int
```

```
In [21]: abs(-9.0)
```

```
Out[21]: 9.0
```

```
In [22]: type(abs(-9.0))
```

```
Out[22]: float
```

جذر

```
In [5]: math.sqrt(2)  
Out[5]: 1.4142135623730951
```

```
In [6]: math.sqrt(4)  
Out[6]: 2.0
```

```
In [7]: type(math.sqrt(4))  
Out[7]: float
```

فاكتوريل

```
In [8]: math.factorial(4)
```

```
Out[8]: 24
```

```
In [9]: type(math.factorial(4))
```

```
Out[9]: int
```

توان

```
In [10]: math.pow(2,4)
```

```
Out[10]: 16.0
```

```
In [11]: math.pow(3,3)
```

```
Out[11]: 27.0
```

```
In [12]: type(math.pow(3,3))
```

```
Out[12]: float
```

```
In [13]: 3**3
```

```
Out[13]: 27
```

```
In [14]: type(3**3)
```

```
Out[14]: int
```

```
In [15]: 3.0**3
```

```
Out[15]: 27.0
```

```
In [16]: type(3.0**3)
```

```
Out[16]: float
```

گرد کردن اعداد به سمت عدد صحیح بالاتر

```
In [24]: math.ceil(9.7)
```

```
Out[24]: 10
```

```
In [25]: math.ceil(9.1)
```

```
Out[25]: 10
```

```
In [26]: math.ceil(9.000001)
```

```
Out[26]: 10
```

```
In [27]: math.ceil(8.999999)
```

```
Out[27]: 9
```

```
In [28]: type(math.ceil(9.000001))
```

```
Out[28]: int
```

گرد کردن اعداد به سمت عدد صحیح پایین‌تر

```
In [29]: math.floor(9.7)
```

```
Out[29]: 9
```

```
In [30]: math.floor(9.1)
```

```
Out[30]: 9
```

```
In [31]: math.floor(9.0001)
```

```
Out[31]: 9
```

```
In [32]: math.floor(8.99999)
```

```
Out[32]: 8
```

```
In [33]: type(math.floor(9.00001))
```

```
Out[33]: int
```

تابع نمایی e^x

```
In [34]: math.exp(1)
```

```
Out[34]: 2.718281828459045
```

```
In [35]: math.exp(0)
```

```
Out[35]: 1.0
```

```
In [36]: math.exp(2)
```

```
Out[36]: 7.38905609893065
```

Ln لگاریتم طبیعی (مبنای e)

```
In [37]: math.log(2)
```

```
Out[37]: 0.6931471805599453
```

```
In [38]: math.log(3)
```

```
Out[38]: 1.0986122886681098
```

```
In [39]: math.log(4)
```

```
Out[39]: 1.3862943611198906
```

```
In [40]: math.log(5)
```

```
Out[40]: 1.6094379124341003
```

```
In [41]: math.log(6)
```

```
Out[41]: 1.791759469228055
```

```
In [42]: math.log(7)
```

```
Out[42]: 1.9459101490553132
```

number	approximate natural logarithm
2	0.693 147 180 559 945 309 417 232 121 458
3	1.098 612 288 668 109 691 395 245 236 92
4	1.386 294 361 119 890 618 834 464 242 92
5	1.609 437 912 434 100 374 600 759 333 23
6	1.791 759 469 228 055 000 812 477 358 38
7	1.945 910 149 055 313 305 105 352 743 44

لگاریتم مبنای ۱۰

```
In [43]: math.log10(2)
```

```
Out[43]: 0.3010299956639812
```

```
In [44]: math.log10(3)
```

```
Out[44]: 0.47712125471966244
```

```
In [45]: math.log10(4)
```

```
Out[45]: 0.6020599913279624
```

```
In [46]: math.log10(5)
```

```
Out[46]: 0.6989700043360189
```

```
In [47]: math.log10(6)
```

```
Out[47]: 0.7781512503836436
```

```
In [48]: math.log10(7)
```

```
Out[48]: 0.8450980400142568
```

لگاریتم در مبنای دلخواه

```
In [49]: math.log(2,10)
```

```
Out[49]: 0.30102999566398114
```

```
In [50]: math.log(2,math.e)
```

```
Out[50]: 0.6931471805599453
```

```
In [51]: math.log(2,2)
```

```
Out[51]: 1.0
```

```
In [52]: math.log(3,2)
```

```
Out[52]: 1.5849625007211563
```

```
In [53]: math.log(4,2)
```

```
Out[53]: 2.0
```

تبدیل درجه به رادیان و برعکس

```
In [58]: math.radians(90)
```

```
Out[58]: 1.5707963267948966
```

```
In [59]: math.degrees(math.pi/2)
```

```
Out[59]: 90.0
```

```
In [60]: math.radians(30)
```

```
Out[60]: 0.5235987755982988
```

```
In [61]: math.degrees(math.pi/6)
```

```
Out[61]: 29.999999999999996
```

روابط مثلثاتی

```
In [62]: math.sin(math.pi/6)
Out[62]: 0.49999999999999994
```

```
In [63]: math.sin(math.pi/3)
Out[63]: 0.8660254037844386
```

```
In [64]: math.sin(math.pi/2)
Out[64]: 1.0
```

```
In [65]: math.sin(math.radians(30))
Out[65]: 0.49999999999999994
```

```
In [66]: math.sin(math.radians(60))
Out[66]: 0.8660254037844386
```

```
In [67]: math.sin(math.radians(90))
Out[67]: 1.0
```

```
In [68]: math.cos(math.pi/6)
Out[68]: 0.8660254037844387
```

```
In [69]: math.tan(math.pi/6)
Out[69]: 0.5773502691896257
```

```
In [70]: math.asin(0.5)
Out[70]: 0.5235987755982989
```

```
In [71]: math.degrees(math.asin(0.5))
Out[71]: 30.000000000000004
```

```
In [72]: math.degrees(math.acos(0.5))
Out[72]: 60.000000000000001
```

```
In [73]: math.degrees(math.atan(0.5))
Out[73]: 26.56505117707799
```



PYTHON

SCIPY Library

Numpy

NumPy یکی از اصلی ترین پکیج های محاسبات علمی در پایتون است. این کتابخانه امکان ایجاد آرایه های چند بعدی، ماتریسها و .. را به ما می دهد، یک آرایه NumPy ویژگی هایی همچون محاسبات ریاضی، منطقی، تغییر شکل آرایه ها، مرتب سازی، انتخاب، جبر خطی، محاسبات آماری، شبیه سازی تصادفی و... را نیز در اختیار می گذارد.



Numpy

هسته این کتابخانه از نوع ndarray است، این آرایه چند بعدی که انواع مختلفی از داده را می‌تواند ذخیره نماید، بسیار کارآمد و بهینه طراحی شده و تفاوت‌های اصلی آن به با لیست‌های استاندارد پایتون به شرح زیر می‌باشد:

- آرایه NumPy برخلاف لیست‌های پایتون اندازه ثابتی در هنگام ساخته شدن دارد، و تغییر در اندازه این آرایه منجر به ساخته شدن یک آرایه جدید و پاک شدن آرایه قبلی می‌شود.
- تمامی پارامترهای یک آرایه NumPy در حالت پیش فرض می‌بایست از یک نوع خاص داده باشند که باعث بهبود عملکرد در مدیریت حافظه می‌شود. اگرچه امکان تعریف داده‌های متفاوت نیز در یک آرایه وجود دارد.
- آرایه‌های NumPy قادر به انجام بسیاری از عملیات آماری و ریاضی بر روی داده‌ها زیاد، به صورت کاملاً کارآمد و بهینه با خطوط بسیار کمی از کد هستند.

فراخوانی کتابخانه numpy

- برای فراخوانی کتابخانه numpy به محیط پایتون می‌توانیم به صورت زیر (تعریف اسم مستعار برای کتابخانه) عمل کنیم :

```
import numpy as np
```


تعریف بردار سطری در numpy

```
In [81]: R = np.array([1, 4, 6, 10])
```

```
In [82]: print(R)  
[ 1  4  6 10]
```

```
In [83]: type(R)  
Out[83]: numpy.ndarray
```

تعریف بردار ستونی در numpy

```
In [84]: C = np.array([[1], [4], [6], [10]])
```

```
In [85]: print(C)
```

```
[[ 1]
 [ 4]
 [ 6]
 [10]]
```

```
In [86]: type(C)
```

```
Out[86]: numpy.ndarray
```

تعریف ماتریس در numpy

```
In [87]: M = np.array([[1, 4], [6, 10]])
```

```
In [88]: print(M)
```

```
[[ 1  4]
 [ 6 10]]
```

```
In [89]: type(M)
```

```
Out[89]: numpy.ndarray
```

تغییر شکل ماتریس و بردار

```
In [104]: np.array([1, 4, 6, 10]).reshape(2, 2)
```

```
Out[104]:
```

```
array([[ 1,  4],  
       [ 6, 10]])
```

```
In [105]: np.array([1, 4, 6, 10]).reshape(1, 4)
```

```
Out[105]: array([[ 1,  4,  6, 10]])
```

```
In [106]: np.array([1, 4, 6, 10]).reshape(4, 1)
```

```
Out[106]:
```

```
array([[ 1],  
       [ 4],  
       [ 6],  
       [10]])
```

تغییر شکل ماتریس و بردار

```
In [107]: np.array([1, 5, 10, 12, 3, 6, 9, 0])
```

```
Out[107]: array([ 1,  5, 10, 12,  3,  6,  9,  0])
```

```
In [108]: np.array([1, 5, 10, 12, 3, 6, 9, 0]).reshape(2, 4)
```

```
Out[108]:
```

```
array([[ 1,  5, 10, 12],  
       [ 3,  6,  9,  0]])
```

```
In [109]: np.array([1, 5, 10, 12, 3, 6, 9, 0]).reshape(4, 2)
```

```
Out[109]:
```

```
array([[ 1,  5],  
       [10, 12],  
       [ 3,  6],  
       [ 9,  0]])
```

```
In [110]: np.array([1, 5, 10, 12, 3, 6, 9, 0]).reshape(8, 1)
```

```
Out[110]:
```

```
array([[ 1],  
       [ 5],  
       [10],  
       [12],  
       [ 3],  
       [ 6],  
       [ 9],  
       [ 0]])
```

ماتريس واحد

```
In [90]: np.eye(4)
```

```
Out[90]:
```

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

```
In [91]: np.eye(4, 4)
```

```
Out[91]:
```

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

```
In [92]: np.eye(3, 4)
```

```
Out[92]:
```

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.]])
```

ماتریس صفر

```
In [96]: np.zeros(4)
```

```
Out[96]: array([0., 0., 0., 0.])
```

```
In [97]: np.zeros([4, 4])
```

```
Out[97]:
```

```
array([[0., 0., 0., 0.],  
       [0., 0., 0., 0.],  
       [0., 0., 0., 0.],  
       [0., 0., 0., 0.]])
```

```
In [98]: np.zeros([3, 4])
```

```
Out[98]:
```

```
array([[0., 0., 0., 0.],  
       [0., 0., 0., 0.],  
       [0., 0., 0., 0.]])
```

```
In [99]: np.zeros(8).reshape(2, 4)
```

```
Out[99]:
```

```
array([[0., 0., 0., 0.],  
       [0., 0., 0., 0.]])
```

ماتریس یک

```
In [100]: np.ones(4)
```

```
Out[100]: array([1., 1., 1., 1.])
```

```
In [101]: np.ones([4, 4])
```

```
Out[101]:
```

```
array([[1., 1., 1., 1.],  
       [1., 1., 1., 1.],  
       [1., 1., 1., 1.],  
       [1., 1., 1., 1.]])
```

```
In [102]: np.ones([3, 4])
```

```
Out[102]:
```

```
array([[1., 1., 1., 1.],  
       [1., 1., 1., 1.],  
       [1., 1., 1., 1.]])
```

```
In [103]: np.ones(8).reshape(2, 4)
```

```
Out[103]:
```

```
array([[1., 1., 1., 1.],  
       [1., 1., 1., 1.]])
```


ابعاد ماتریس و بردار

```
In [112]: M = np.array([[ 1,  5], [10, 12], [ 3,  6], [ 9,  0]])
```

```
In [113]: print(M)
```

```
[[ 1  5]
 [10 12]
 [ 3  6]
 [ 9  0]]
```

```
In [114]: np.size(M)
```

```
Out[114]: 8
```

```
In [115]: np.size(M, 0)
```

```
Out[115]: 4
```

```
In [116]: np.size(M, 1)
```

```
Out[116]: 2
```

```
In [117]: np.shape(M)
```

```
Out[117]: (4, 2)
```

تقسیم یک بازه به گام‌های مساوی

```
In [118]: np.arange(0, 10, 2)
```

```
Out[118]: array([0, 2, 4, 6, 8])
```

```
In [119]: np.arange(0, 9, 2)
```

```
Out[119]: array([0, 2, 4, 6, 8])
```

```
In [120]: np.arange(0, 11, 2)
```

```
Out[120]: array([ 0,  2,  4,  6,  8, 10])
```

```
In [121]: np.arange(10, 0, -1)
```

```
Out[121]: array([10,  9,  8,  7,  6,  5,  4,  3,  2,  1])
```

تقسیم یک بازه به n نقطه با فاصله برابر

```
In [122]: np.linspace(0, 10, 6)
```

```
Out[122]: array([ 0.,  2.,  4.,  6.,  8., 10.])
```

```
In [123]: np.linspace(10, 0, 6)
```

```
Out[123]: array([10.,  8.,  6.,  4.,  2.,  0.])
```

```
In [124]: np.linspace(0, 10, 5)
```

```
Out[124]: array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

ترانهاده یک ماتریس

```
In [133]: M = np.array([[ 1,  5], [10, 12], [ 3,  6], [ 9,  0]])
```

```
In [134]: np.shape(M)
```

```
Out[134]: (4, 2)
```

```
In [135]: print(M)
```

```
[[ 1  5]
 [10 12]
 [ 3  6]
 [ 9  0]]
```

```
In [136]: MT = np.transpose(M)
```

```
In [137]: np.shape(MT)
```

```
Out[137]: (2, 4)
```

```
In [138]: print(MT)
```

```
[[ 1 10  3  9]
 [ 5 12  6  0]]
```

```
In [139]: M.T
```

```
Out[139]:
```

```
array([[ 1, 10,  3,  9],
       [ 5, 12,  6,  0]])
```

فراخوانی درایه‌ها ماتریس و بردار

```
In [21]: A
```

```
Out[21]:
```

```
array([[ -6,   4, -18,  11],  
       [ 10,  -5, -18,  14],  
       [ -3,   6, -18,  14],  
       [ 14, -17,  -1, -16],  
       [ -4,   9,  17,  14]])
```

```
In [22]: A[0,1]
```

```
Out[22]: 4
```

```
In [23]: A[1,3]
```

```
Out[23]: 14
```

```
In [24]: A[2,2]
```

```
Out[24]: -18
```

```
In [25]: A[4,2]
```

```
Out[25]: 17
```

فراخوانی سطرها و ستون‌های ماتریس

```
In [30]: A[:,0]
Out[30]: array([-6, 15,  9, -9, -4])
```

```
In [31]: A[:,2]
Out[31]: array([ 15,  2, 12,  6, -12])
```

```
In [32]: A[0,:]
Out[32]: array([-6,  8, 15, -3])
```

```
In [33]: A[2,:]
Out[33]: array([ 9, -3, 12, 15])
```

```
In [34]: A[[0,2],:]
Out[34]:
array([[ -6,  8, 15, -3],
       [  9, -3, 12, 15]])
```

```
In [35]: A[:,[0,2]]
Out[35]:
array([[ -6, 15],
       [ 15,  2],
       [  9, 12],
       [-9,  6],
       [-4, -12]])
```

```
In [36]: A[:,:]
Out[36]:
array([[ -6,  8, 15, -3],
       [ 15, -6,  2, -1],
       [  9, -3, 12, 15],
       [-9, -12,  6,  9],
       [-4, -13, -12, -19]])
```

```
In [38]: A[np.arange(3),:]
Out[38]:
array([[ -6,  8, 15, -3],
       [ 15, -6,  2, -1],
       [  9, -3, 12, 15]])
```

جمع ماتریس

```
8 import numpy as np
9 A = np.array([4, -6, 1, -6, 5, 7, 1, 7, 8]).reshape(3, 3)
10 B = np.array([3, 5, 0, 5, 7, -2, 0, -2, 9]).reshape(3, 3)
11 C = A + B
12 D = B + A
13
14 print('Matrix A =\n', A)
15 print('Matrix B =\n', B)
16
17 print('AB =\n', C)
18 print('Size of matrix C is =', C.shape)
19 print('AB =\n', D)
20 print('Size of matrix C is =', D.shape)
```

تفریق ماتریس

```
8 import numpy as np
9 A = np.array([4, -6, 1, -6, 5, 7, 1, 7, 8]).reshape(3, 3)
10 B = np.array([3, 5, 0, 5, 7, -2, 0, -2, 9]).reshape(3, 3)
11 C = A - B
12 D = B - A
13
14 print('Matrix A =\n', A)
15 print('Matrix B =\n', B)
16
17 print('AB =\n', C)
18 print('Size of matrix C is =', C.shape)
19 print('AB =\n', D)
20 print('Size of matrix C is =', D.shape)
```


ضرب ماتریس و بردار

Determine the products $\mathbf{C} = \mathbf{AB}$ and $\mathbf{D} = \mathbf{BA}$ if

$$\mathbf{A} = \begin{bmatrix} 4 & -6 & 2 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 3 \\ 1 \\ -5 \end{bmatrix}$$

```
9 import numpy as np
10 A = np.array([4, -6, 2]).reshape(1, 3)
11 B = np.array([3, 1, -5]).reshape(3, 1)
12
13 C = A @ B
14 D = B @ A
15
16 print('Vector A =\n', A)
17 print('Vector B =\n', B)
18
19 print('AB =\n', C)
20 print(type(C))
21 print('Size of matrix C is =', C.shape)
22
23 print('BA =\n', D)
24 print('Size of matrix D is =', D.shape)
25
26 F = D.tolist()
27 print(type(F))
28 print(F)
```

ضرب ماتریس و بردار (روش دوم)

Determine the products $\mathbf{C} = \mathbf{AB}$ and $\mathbf{D} = \mathbf{BA}$ if

$$\mathbf{A} = \begin{bmatrix} 4 & -6 & 2 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 3 \\ 1 \\ -5 \end{bmatrix}$$

```
8 import numpy as np
9 A = np.matrix([4, -6, 2]).reshape(1, 3)
10 B = np.matrix([3, 1, -5]).reshape(3, 1)
11
12 C = A * B
13 D = B * A
14
15 print('Vector A =\n', A)
16 print('Vector B =\n', B)
17
18 print('AB =\n', C)
19 print(type(C))
20 print('Size of matrix C is =', C.shape)
21
22 print('BA =\n', D)
23 print('Size of matrix D is =', D.shape)
```

ضرب ماتریس و بردار

Determine the products $\mathbf{C} = \mathbf{AB}$ and $\mathbf{D} = \mathbf{BA}$ if

$$\mathbf{A} = \begin{bmatrix} 4 & -6 & 1 \\ -6 & 5 & 7 \\ 1 & 7 & 8 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 3 & 5 & 0 \\ 5 & 7 & -2 \\ 0 & -2 & 9 \end{bmatrix}$$

```
8 import numpy as np
9 A = np.array([4, -6, 1, -6, 5, 7, 1, 7, 8]).reshape(3, 3)
10 B = np.array([3, 5, 0, 5, 7, -2, 0, -2, 9]).reshape(3, 3)
11 C = A @ B
12 D = B @ A
13
14 print('Matrix A =\n', A)
15 print('Matrix B =\n', B)
16
17 print('AB =\n', C)
18 print('Size of matrix C is =', C.shape)
19 print('BA =\n', D)
20 print('Size of matrix C is =', D.shape)
```

ضرب ماتریس و بردار

Determine the matrix triple product $\mathbf{C} = \mathbf{B}^T \mathbf{A} \mathbf{B}$ if

$$\mathbf{A} = \begin{bmatrix} 300 & -100 \\ -100 & 200 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0.6 & 0.8 & -0.6 & -0.8 \\ -0.8 & 0.6 & 0.8 & -0.6 \end{bmatrix}$$

```
8 import numpy as np
9 A = np.array([300, -100, -100, 200]).reshape(2, 2)
10 B = np.array([0.6, 0.8, -0.6, -0.8, -0.8, 0.6, 0.8, -0.6]).reshape(2, 4)
11 C = B.transpose() @ A @ B
12
13 print('Matrix A =\n', A)
14 print('Matrix B =\n', B)
15
16 print('transpose(B)*A*B =\n', C)
17 print('Size of matrix C is =', C.shape)
```

حل دستگاه خطی به روش ماتریسی

مطلوبست حل دستگاه معادلات زیر به روش ماتریسی :

$$2x_1 - 3x_2 + x_3 = -18$$

$$-9x_1 + 5x_2 + 3x_3 = 18$$

$$4x_1 + 7x_2 - 8x_3 = 53$$

```
8 import numpy as np
9 A = np.array([2, -3, 1, -9, 5, 3, 4, 7, -8]).reshape(3, 3)
10 B = np.array([-18, 18, 53]).reshape(3, 1)
11 X = np.linalg.solve(A, B)
12
13 print('Matrix A =\n', A)
14 print('Vector B =\n', B)
15
16 print('X =\n', X)
```

حل دستگاه خطی به روش ماتریسی

مطلوبست حل دستگاه معادلات زیر به روش ماتریسی :

$$2x_1 - 5x_2 + 8x_3 + 11x_4 = 39$$

$$10x_1 + 7x_2 + 4x_3 - x_4 = 127$$

$$-3x_1 + 9x_2 + 5x_3 - 6x_4 = 58$$

$$x_1 - 4x_2 - 2x_3 + 9x_4 = -14$$

```
8 import numpy as np
9 A = np.array(
10     [2, -5, 8, 11, 10, 7, 4, -1, -3, 9, 5, -6, 1, -4, -2, 9]).reshape(4, 4)
11 B = np.array([39, 127, 58, -14]).reshape(4, 1)
12 X = np.linalg.solve(A, B)
13
14 print('Matrix A =\n', A)
15 print('Vector B =\n', B)
16
17 print('X =\n', X)
```

معکوس ماتریس

مطلوبست تعیین معکوس ماتریس زیر :

$$\mathbf{A} = \begin{bmatrix} 5 & 3 & -4 \\ 3 & 8 & -2 \\ -4 & -2 & 7 \end{bmatrix}$$

```
8 import numpy as np
9 np.set_printoptions(precision=7)
10
11 A = np.array(
12     [5, 3, -4, 3, 8, -2, -4, -2, 7]).reshape(3, 3)
13 AI = np.linalg.inv(A)
14
15 print('Matrix A =\n', A)
16 print('Inverse of matrix A =\n', AI)
```

معكوس ماتريس

مطلوبست تعيين معكوس ماتريس زير :

$$\mathbf{A} = \begin{bmatrix} 7 & -6 & 3 & -2 \\ -6 & 4 & -1 & 5 \\ 3 & -1 & 8 & 9 \\ -2 & 5 & 9 & 2 \end{bmatrix}$$

```
19 B = np.array(  
20     [7, -6, 3, -2, -6, 4, -1, 5, 3, -1, 8, 9, -2, 5, 9, 2]).reshape(4, 4)  
21 BI = np.linalg.inv(B)  
22  
23 print('Matrix B =\n', B)  
24 print('Inverse of matrix B =\n', BI)
```


دترمینان ماتریس

مطلوبست تعیین رنک ماتریس زیر :

$$\begin{vmatrix} 9 & 1 & 9 & 9 & 9 \\ 9 & 0 & 9 & 9 & 2 \\ 4 & 0 & 0 & 5 & 0 \\ 9 & 0 & 3 & 9 & 0 \\ 6 & 0 & 0 & 7 & 0 \end{vmatrix}$$

```
8 import numpy as np
9 np.set_printoptions(precision=7)
10
11 A = np.array(
12     [9, 1, 9, 9, 9, 9, 0, 9, 9, 2, 4, 0, 0, 5, 0, 9, 0, 3, 9, 0, 6, 0, 0, 7, 0]).reshape(5, 5)
13 AD = np.linalg.det(A)
14
15 print('Matrix A =\n', A)
16 print('Determinant of matrix A =\n', AD)
```

رنگ ماتریس

مطلوبست تعیین رنگ ماتریس زیر :

Determine the rank of the matrix

$$A = \begin{bmatrix} 2 & 5 & -3 & -4 & 8 \\ 4 & 7 & -4 & -3 & 9 \\ 6 & 9 & -5 & 2 & 4 \\ 0 & -9 & 6 & 5 & -6 \end{bmatrix}$$

```
8 import numpy as np
9 np.set_printoptions(precision=7)
10
11 A = np.array(
12     [2, 5, -3, -4, 8, 4, 7, -4, -3, 9, 6, 9, -5, 2, 4, 0, -9, 6, 5, -6]).reshape(4, 5)
13 AR = np.linalg.matrix_rank(A)
14
15 print('Matrix A =\n', A)
16 print('Rank of matrix A =\n', AR)
17
18 B = A[1:4, 1:4]
19 print('Matrix B =\n', B, '\n')
20 print('Determinant of matrix B =\n', np.linalg.det(B), '\n')
21 print('Inverse of matrix B =\n', np.linalg.inv(B))
```

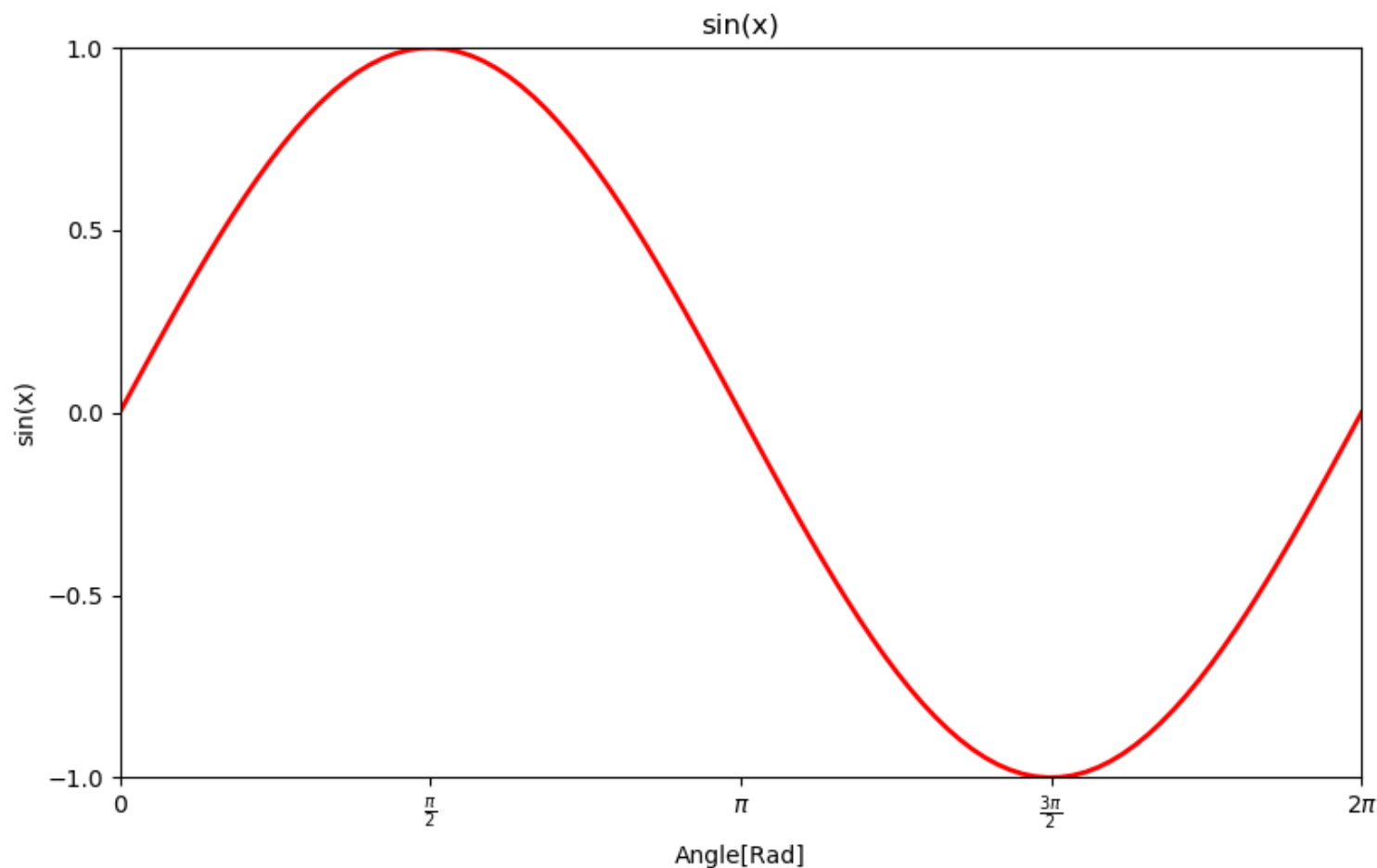
قطر اصلی، جمع درایه‌ها، ماکزیمم و مینیمم ماتریس

```
8 import numpy as np
9
10 A = np.random.randint(-50, 50, (5, 5))
11 print('Random integer matrix A is =\n', A, '\n')
12
13 A_D = np.diag(A)
14 print('Diagonal elements of matrix A is =\n', A_D, '\n')
15
16 A_Sum_C = np.sum(A, 0)
17 A_Sum_R = np.sum(A, 1)
18 print('Sum of matrix A by columns =\n', A_Sum_C)
19 print('Sum of matrix A by rows =\n', A_Sum_R)
20
21 A_Sum_D = np.sum(A_D)
22 print('Sum of diagonal elements =\n', A_Sum_D)
23
24 A_max = np.max(A)
25 print('Max of matrix A =\n', A_max)
26 A_max_C = np.max(A, 0)
27 print('Max of matrix A by column =\n', A_max_C)
28 A_max_R = np.max(A, 1)
29 print('Max of matrix A by row =\n', A_max_R)
```

ترسیم تابع $\sin(x)$ در بازه صفر تا 2π

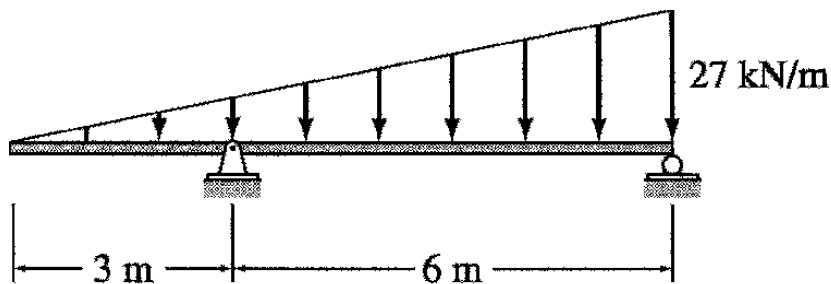
```
8 import numpy as np
9 import pylab as plt
10
11 x = np.linspace(0, 2*np.pi, 100)
12 y = np.sin(x)
13 plt.figure('Plot of Sin(x)')
14 plt.plot(x, y, 'r-', linewidth=2.0)
15 plt.title('sin(x)')
16 plt.xlabel('Angle[Rad]')
17 plt.ylabel('sin(x)')
18 plt.xlim(0, 2*np.pi)
19 plt.ylim(-1, 1)
20 plt.xticks(np.linspace(0, 2*np.pi, 5),
21            ('0', r'$\frac{\pi}{2}$', r'$\pi$',
22             r'$\frac{3}{2} \pi$', r'$2 \pi$'))
23 plt.yticks(np.linspace(-1, 1, 5))
24 plt.rc('font', size=18)
25
```

ترسیم تابع $\sin(x)$ در بازه صفر تا 2π

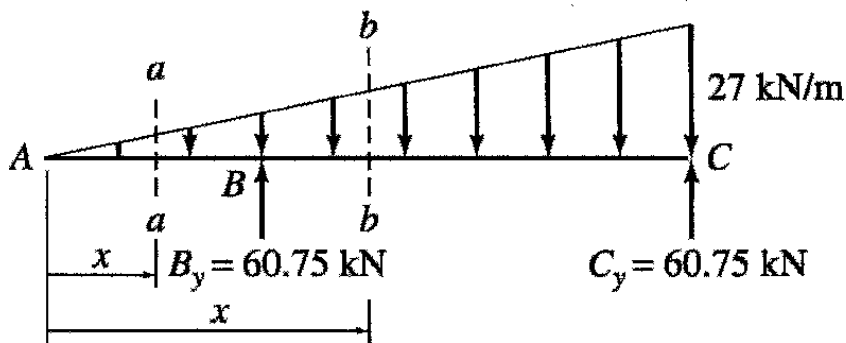


رسم دیاگرام برش و خمش با پایتون

مطلوبست رسم نمودار تغییرات نیروی برشی و لنگر خمشی
سازه زیر به روش مقطع زدن :

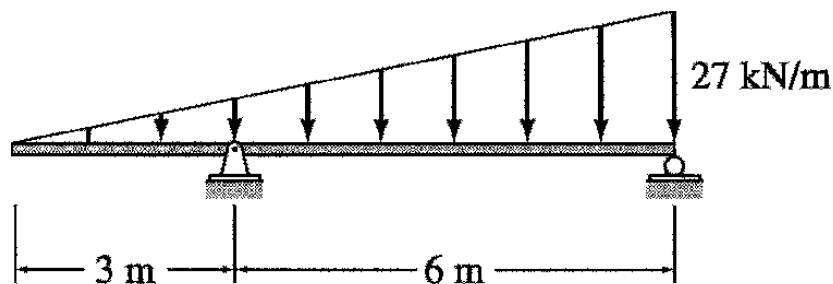


رسم دیاگرام آزاد :

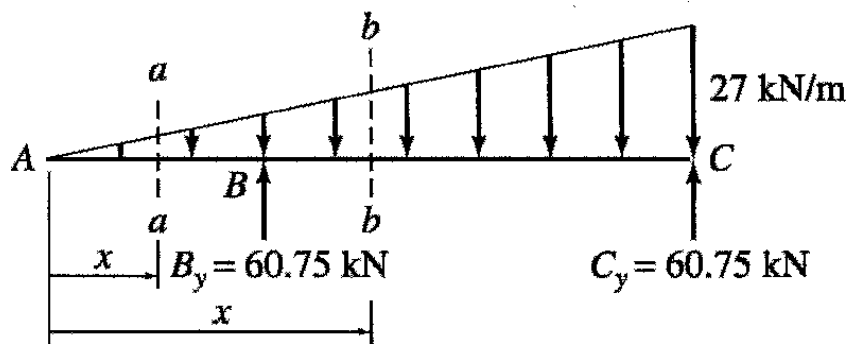


رسم دیاگرام برش و خمش با پایتون

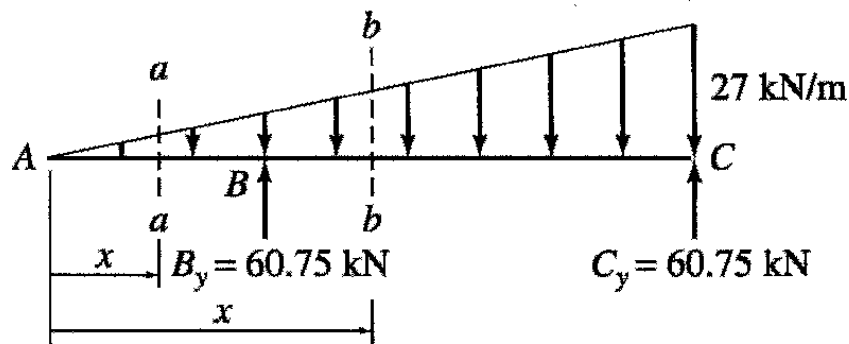
مطلوبست رسم نمودار تغییرات نیروی برشی و لنگر خمشی
سازه زیر به روش مقطع زدن :



رسم دیاگرام آزاد :



رسم دیاگرام برش و خمش با پایتون



محاسبه عکس العمل های تکیه گاهی :

$$+ \rightarrow \sum F_x = 0 \quad B_x = 0$$

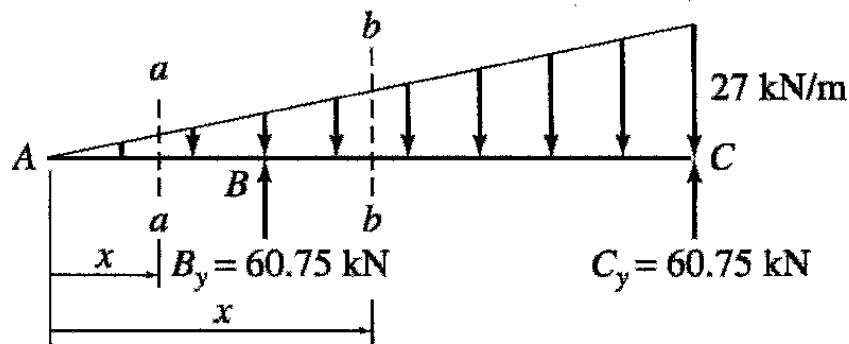
$$+ \zeta \sum M_c = 0$$

$$\left(\frac{1}{2}\right)(9)(27)\left(\frac{9}{3}\right) - B_y(6) = 0 \quad B_y = 60.75 \text{ kN } \uparrow$$

$$+ \uparrow \sum F_y = 0$$

$$-\left(\frac{1}{2}\right)(9)(27) + 60.75 + C_y = 0 \quad C_y = 60.75 \text{ kN } \uparrow$$

رسم دیاگرام برش و خمش با پایتون



محاسبه عکس العمل های تکیه گاهی :

$$+ \rightarrow \sum F_x = 0 \quad B_x = 0$$

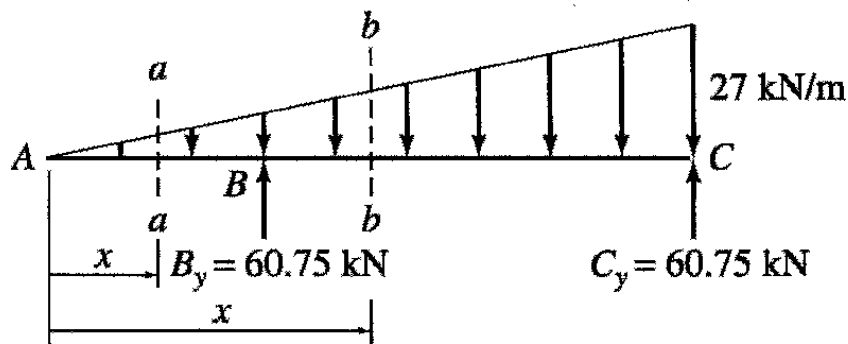
$$+ \curvearrowright \sum M_c = 0$$

$$\left(\frac{1}{2}\right)(9)(27)\left(\frac{9}{3}\right) - B_y(6) = 0 \quad B_y = 60.75 \text{ kN } \uparrow$$

$$+ \uparrow \sum F_y = 0$$

$$-\left(\frac{1}{2}\right)(9)(27) + 60.75 + C_y = 0 \quad C_y = 60.75 \text{ kN } \uparrow$$

رسم دیاگرام برش و خمش با پایتون



محاسبه نیروهای برشی داخلی :

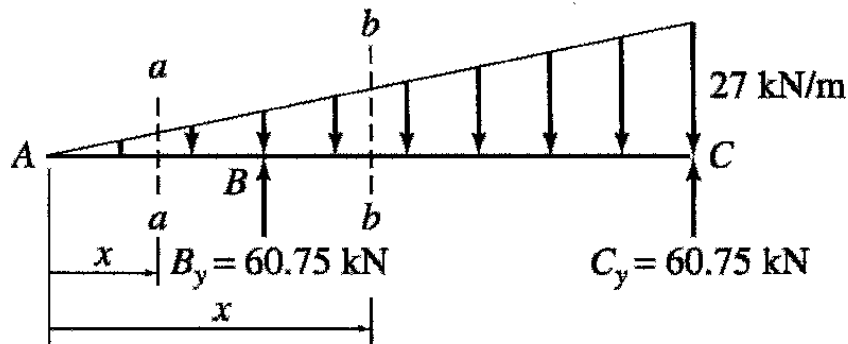
$$w(x) = \left(\frac{27}{9}\right)x = 3x \text{ kN/m}$$

با رسم مقاطع aa و bb در قطعات AB و BC خواهیم داشت :

$$V = -\left(\frac{1}{2}\right)(x)(3x) = -\frac{3x^2}{2} \quad \text{for } 0 \leq x < 3 \text{ m}$$

$$V = -\left(\frac{3x^2}{2}\right) + 60.75 \quad \text{for } 3 \text{ m} < x < 9 \text{ m}$$

رسم دیاگرام برش و خمش با پایتون



محاسبه لنگرهای خمشی داخلی :

$$w(x) = \left(\frac{27}{9}\right)x = 3x \text{ kN/m}$$

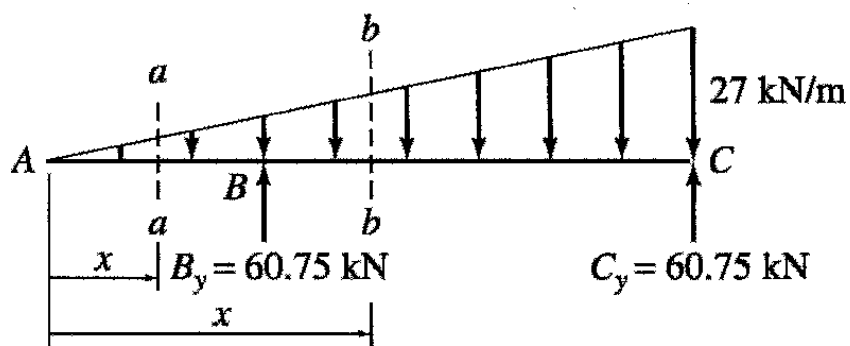
با رسم مقاطع aa و bb در قطعات AB و BC خواهیم داشت :

$$M = -\left(\frac{1}{2}\right)(x)(3x)\left(\frac{x}{3}\right) = -\frac{x^3}{2} \quad \text{for } 0 \leq x \leq 3 \text{ m}$$

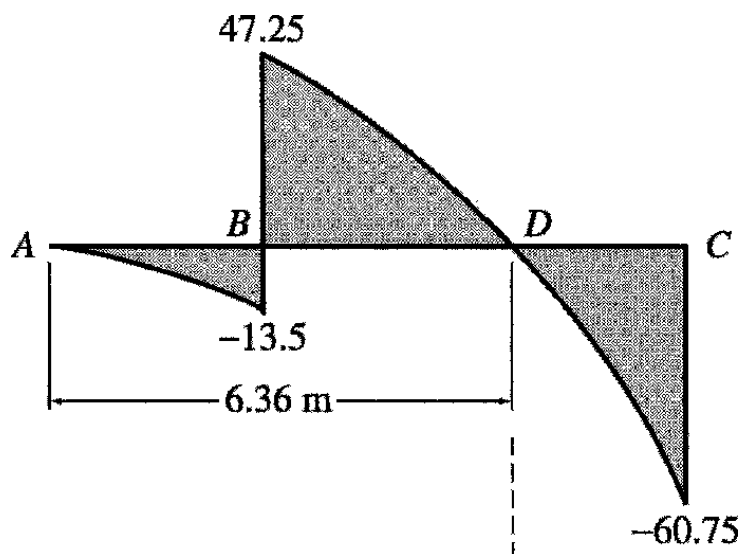
$$M = -\left(\frac{x^3}{2}\right) + 60.75(x - 3) \quad \text{for } 3 \text{ m} \leq x \leq 9 \text{ m}$$

رسم دیاگرام برش و خمش با پایتون

دیاگرام برش :



برای پیدا کردن محل محل نقطه D در قطعه BC داریم :

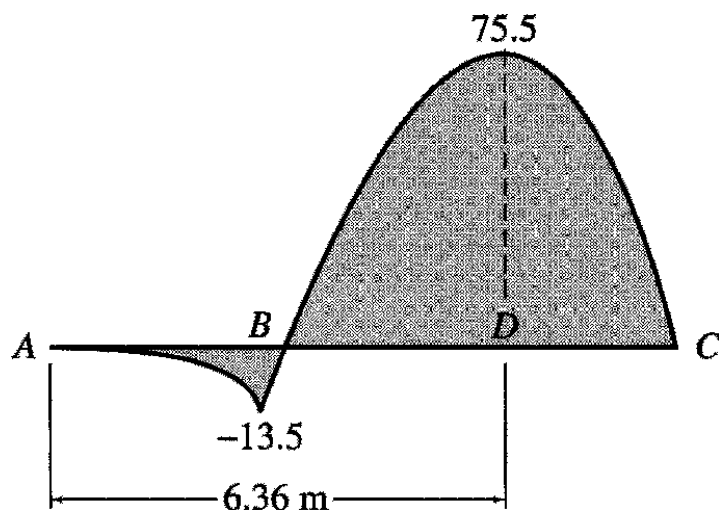
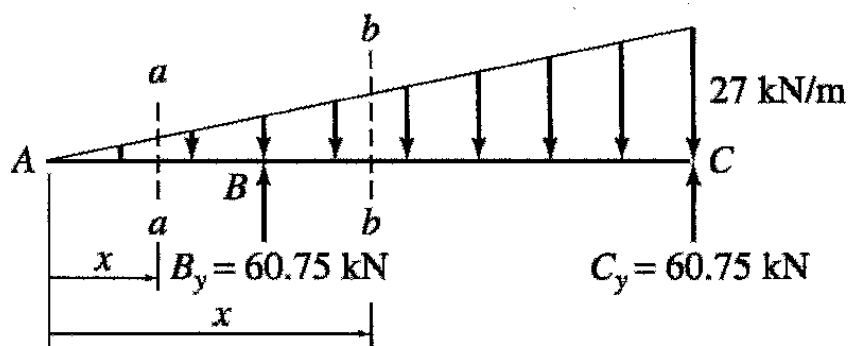


$$V = -\left(\frac{3x^2}{2}\right) + 60.75 = 0$$

$$x = 6.36 \text{ m}$$

رسم دیاگرام برش و خمش با پایتون

دیاگرام خمش :

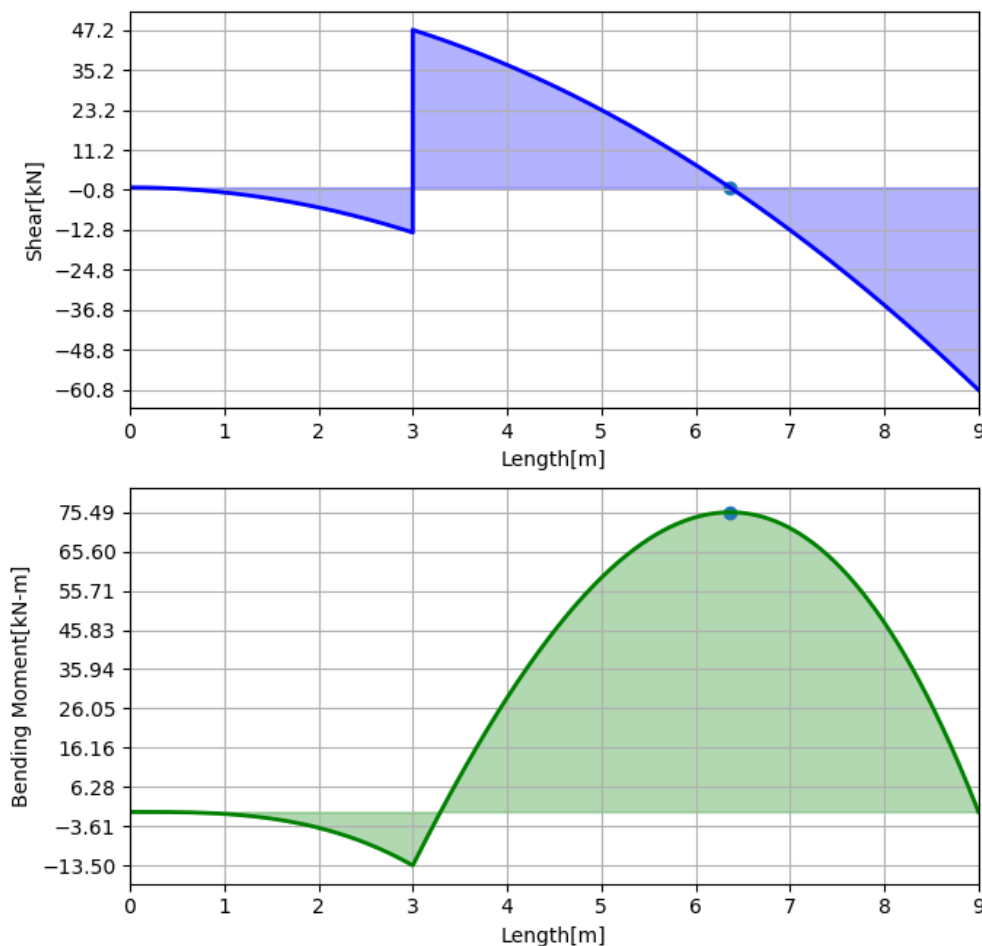


برای پیدا کردن محل نقطه D ماکزیمم ممان
از معادله ممان در قطعه BC مشتق می گیریم :

$$\frac{dM}{dx} = \left(-\frac{3x^2}{2} \right) + 60.75 = 0 \quad \rightarrow \quad x = 6.36 \text{ m}$$

رسم دیاگرام برش و خمش با پایتون

Shear & Bending Moment Diagram

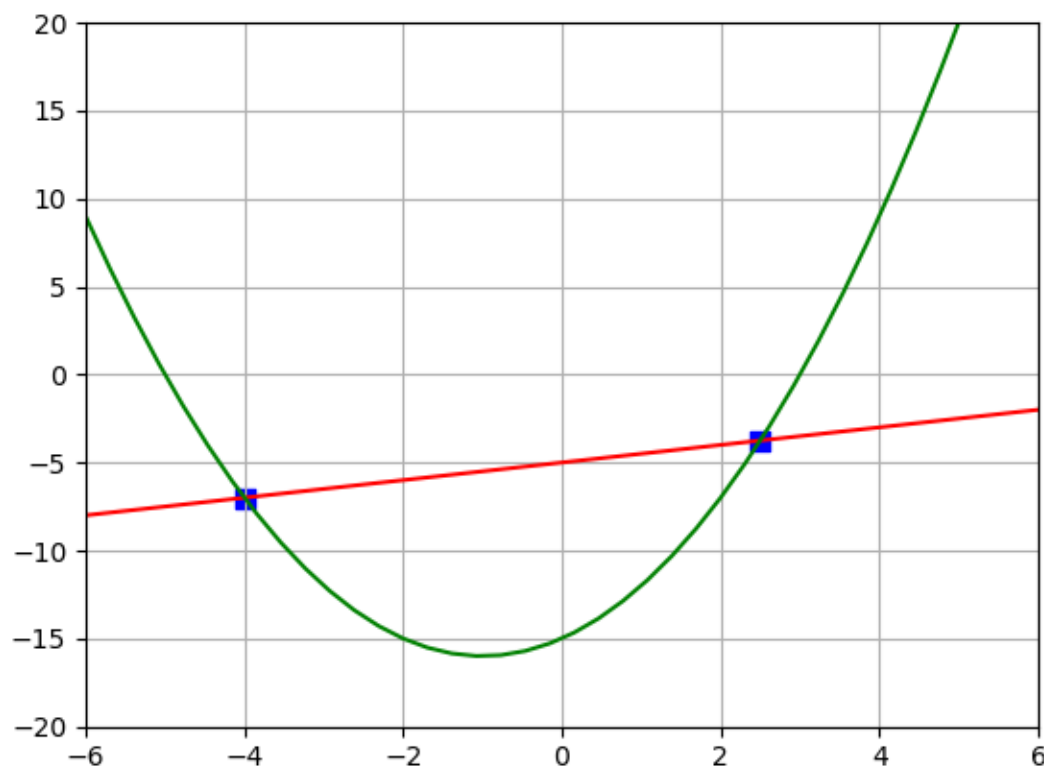


با استفاده از کد
diagramSM.py در فولدر
C10 داریم :

ریشه‌های دستگاه معادله غیر خطی و ترسیم محل تلاقی دو منحنی

مطلوبست تعیین ریشه‌های دستگاه زیر :

$$\begin{cases} y = 0.5x - 5 \\ y = x^2 + 2x - 15 \end{cases}$$



ریشه‌های دستگاه معادله غیر خطی و ترسیم محل تلاقی دو منحنی

کد مربوط به حل دستگاه و
رسم آن :

```
8 import sympy as sp
9 import numpy as np
10 import pylab as plt
11
12 (x, y) = sp.symbols(('x', 'y'))
13
14 eq1 = y - 0.5*x + 5
15 eq2 = y - x**2 + -2*x + 15
16 ans = sp.solve([eq1, eq2], [x, y])
17 print('The roots are (x, y) =\n', ans)
18
19 x = np.linspace(-6, 6, 40)
20 y1 = 0.5*x - 5
21 y2 = x**2 + 2*x - 15
22 plt.plot(x, y1, 'r')
23 plt.plot(x, y2, 'g')
24 plt.grid(True)
25 plt.xlim(-6, 6)
26 plt.ylim(-20, 20)
27
28 ans1 = ans[0]
29 ans2 = ans[1]
30 rootsX = [ans1[0], ans2[0]]
31 rootsY = [ans1[1], ans2[1]]
32 plt.scatter(rootsX, rootsY, s=50, c='b', marker='s')
33
```


انتگرال (عددی) معین از روابط پیچیده با Scipy

$$\begin{aligned}\int_0^{\pi/2} \frac{2 \sin x \cos x}{(1 + \sin^2 x)^3} dx &= \int_1^2 \frac{1}{u^3} du \\ &= -\frac{1}{2u^2} \Big|_1^2 \\ &= -\frac{1}{8} - \left(-\frac{1}{2}\right) = \frac{3}{8}\end{aligned}$$

Let $u = 1 + \sin^2 x$, $du = 2 \sin x \cos x dx$.

When $x = 0$, $u = 1$.

When $x = \pi/2$, $u = 2$.

```
8 import scipy
9 import math
10
11
12 def y(x):
13     return (2*math.sin(x)*math.cos(x)) / (1 + (math.sin(x))**2)**3
14
15
16 # y = lambda x: (2*math.sin(x)*math.cos(x)) / (1 + (math.sin(x))**2)**3
17 iy = scipy.integrate.quad(y, 0, math.pi/2)
18 print(iy)
19
```

انتگرال (عددی) معین از روابط پیچیده با Scipy

$$\begin{aligned}\int_{\pi/4}^{\pi/2} \cot \theta \csc^2 \theta d\theta &= \int_1^0 u \cdot (-du) \\ &= -\int_1^0 u du \\ &= -\left[\frac{u^2}{2}\right]_1^0 \\ &= -\left[\frac{(0)^2}{2} - \frac{(1)^2}{2}\right] = \frac{1}{2}\end{aligned}$$

Let $u = \cot \theta$, $du = -\csc^2 \theta d\theta$,

$-du = \csc^2 \theta d\theta$.

When $\theta = \pi/4$, $u = \cot(\pi/4) = 1$.

When $\theta = \pi/2$, $u = \cot(\pi/2) = 0$.

```
8 import scipy.integrate
9 import math
10
11
12 def g(theta):
13     return (1 / math.tan(theta))*(1 / math.sin(theta))**2
14
15
16 ig = scipy.integrate.quad(g, math.pi/4, math.pi/2)
17 print(ig)
18
```

انتگرال نامعین با Sympy

$$\int \sqrt{2x+1} dx.$$

```

8 import sympy as sp
9
10 sp.init_printing()
11 x = sp.symbols('x')
12
13 y = sp.sqrt(2*x + 1)
14 iy = y.integrate(x)
15 sp.pprint(iy)

```

$$\int x^2 \cos x^3 dx = \int \cos x^3 \cdot x^2 dx$$

$$= \int \cos u \cdot \frac{1}{3} du$$

Let $u = x^3$, $du = 3x^2 dx$,
(1/3) $du = x^2 dx$.

$$= \frac{1}{3} \int \cos u du$$

$$= \frac{1}{3} \sin u + C$$

Integrate with respect to u .

$$= \frac{1}{3} \sin x^3 + C$$

Replace u by x^3 .

```

8 import sympy as sp
9
10 sp.init_printing()
11 x = sp.symbols('x')
12
13 k = x**2 * sp.cos(x**3)
14 ik = k.integrate(x)
15 sp.pprint(ik)
16

```

انتگرال معین با Sympy

$$\int_{-1}^1 3x^2 \sqrt{x^3 + 1} dx$$

Let $u = x^3 + 1$, $du = 3x^2 dx$.

When $x = -1$, $u = (-1)^3 + 1 = 0$.

When $x = 1$, $u = (1)^3 + 1 = 2$.

$$= \int_0^2 \sqrt{u} du$$

$$= \left. \frac{2}{3} u^{3/2} \right|_0^2$$

Evaluate the new definite integral.

$$= \frac{2}{3} [2^{3/2} - 0^{3/2}] = \frac{2}{3} [2\sqrt{2}] = \frac{4\sqrt{2}}{3}$$

```

8 import sympy as sp
9 sp.init_printing()
10
11 x = sp.symbols('x')
12
13 p = (3*x**2) * sp.sqrt(x**3 + 1)
14 ip = sp.integrate(p, (x, -1, 1))
15 sp.pprint(ip)
16

```

انتگرال معین با Sympy

$$\begin{aligned}\int_0^{\pi/2} \frac{2 \sin x \cos x}{(1 + \sin^2 x)^3} dx &= \int_1^2 \frac{1}{u^3} du \\ &= \left[-\frac{1}{2u^2} \right]_1^2 \\ &= -\frac{1}{8} - \left(-\frac{1}{2} \right) = \frac{3}{8}\end{aligned}$$

Let $u = 1 + \sin^2 x$, $du = 2 \sin x \cos x dx$.

When $x = 0$, $u = 1$.

When $x = \pi/2$, $u = 2$.

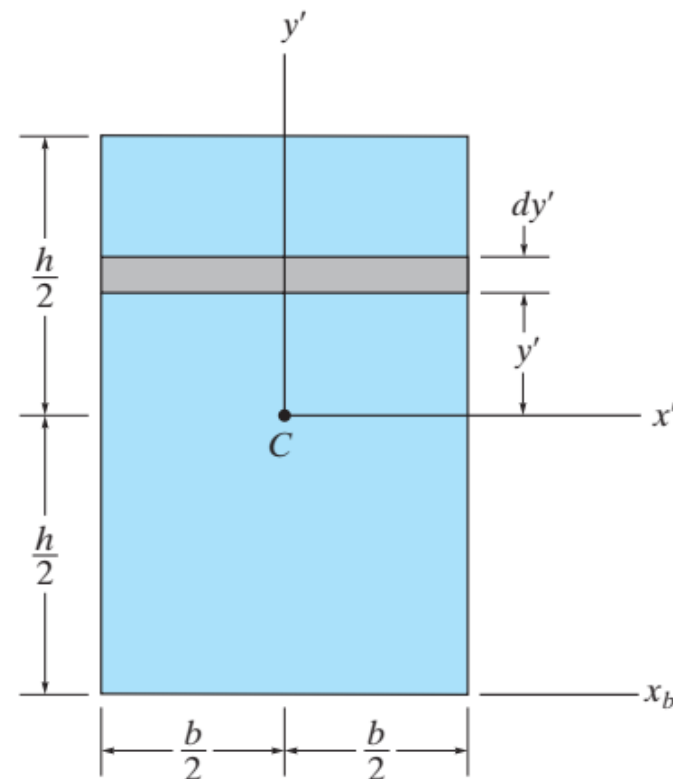
```
8 import sympy as sp
9 sp.init_printing()
10
11 x = sp.symbols('x')
12
13 h = (2*sp.sin(x)*sp.cos(x)) / (1 + (sp.sin(x))**2)**3
14 ih = sp.integrate(h, (x, 0, sp.pi/2))
15 sp.pprint(ih)
16
```

محاسبه پارامتری ممان اینرسی یک مقطع مستطیلی حول مرکز هندسی

$$\bar{I}_{x'} = \int_A y'^2 dA = \int_{-h/2}^{h/2} y'^2 (b dy') = b \int_{-h/2}^{h/2} y'^2 dy'$$

$$\bar{I}_{x'} = \frac{1}{12} b h^3$$

$$\bar{I}_{y'} = \frac{1}{12} h b^3$$



```

8 import sympy as sp
9 sp.init_printing()
10
11 (x, y, b, h) = sp.symbols(('x', 'y', 'b', 'h'))
12
13 Ix = sp.integrate(y**2*b, (y, -h/2, h/2))
14 sp.pprint(Ix)
15 Iy = sp.integrate(x**2*h, (x, -b/2, b/2))
16 sp.pprint(Iy)
17
18 print(float(Ix.subs({b: 5, h: 10})))
19 print(float(Iy.subs({b: 5, h: 10})))
20

```

MATLAB

معرفی و محیط نرم افزار

MATLAB

❑ MATLAB = MATrix laboratory



❑ متلب یک سیستم جامع نرم افزاری برای محاسبات ریاضی و محاسبات تکنیکی می باشد.

❑ نرم افزار متلب از زبان برنامه نویسی سطح بالا استفاده می کند که شامل صدها دستور برای محاسبات ریاضی می باشد.

MATLAB

❑ شرکت توسعه دهنده نرم افزار متلب، شرکت MathWorks می باشد.

❑ آدرس وب سایت نرم افزار:

<https://www.mathworks.com/>

تاریخچه نسخه‌های اخیر MATLAB

R2018b – Not Released	R2018a (Version 9.4) - Mar 2018
R2017b (Version 9.3) - Sep 2017	R2017a (Version 9.2) - Mar 2017
R2016b (Version 9.1) - Sep 2016	R2016a (Version 9.0) - Mar 2016
R2015b (Version 8.6) - Sep 2015	R2015a (Version 8.5) - Mar 2015
R2014b (Version 8.4) - Oct 2014	R2014a (Version 8.3) - Mar 2014
R2013b (Version 8.2) - Sep 2013	R2013a (Version 8.1) - Mar 2013
R2012b (Version 8.0) - Sep 2012	R2012a (Version 7.14) - Mar 2012
R2010b (Version 7.11) - Sep 2010	R2011a (Version 7.12) - Apr 2011
R2011b (Version 7.13) - Sep 2011	R2010a (Version 7.10) - Mar 2010
R2009b (Version 7.9) - Sep 2009	R2009a (Version 7.8) - Mar 2009
R2008b (Version 7.7) - Oct 2008	R2008a (Version 7.6) - Mar 2008
R2007b (Version 7.5) - Sep 2007	R2007a (Version 7.4) - Mar 2007

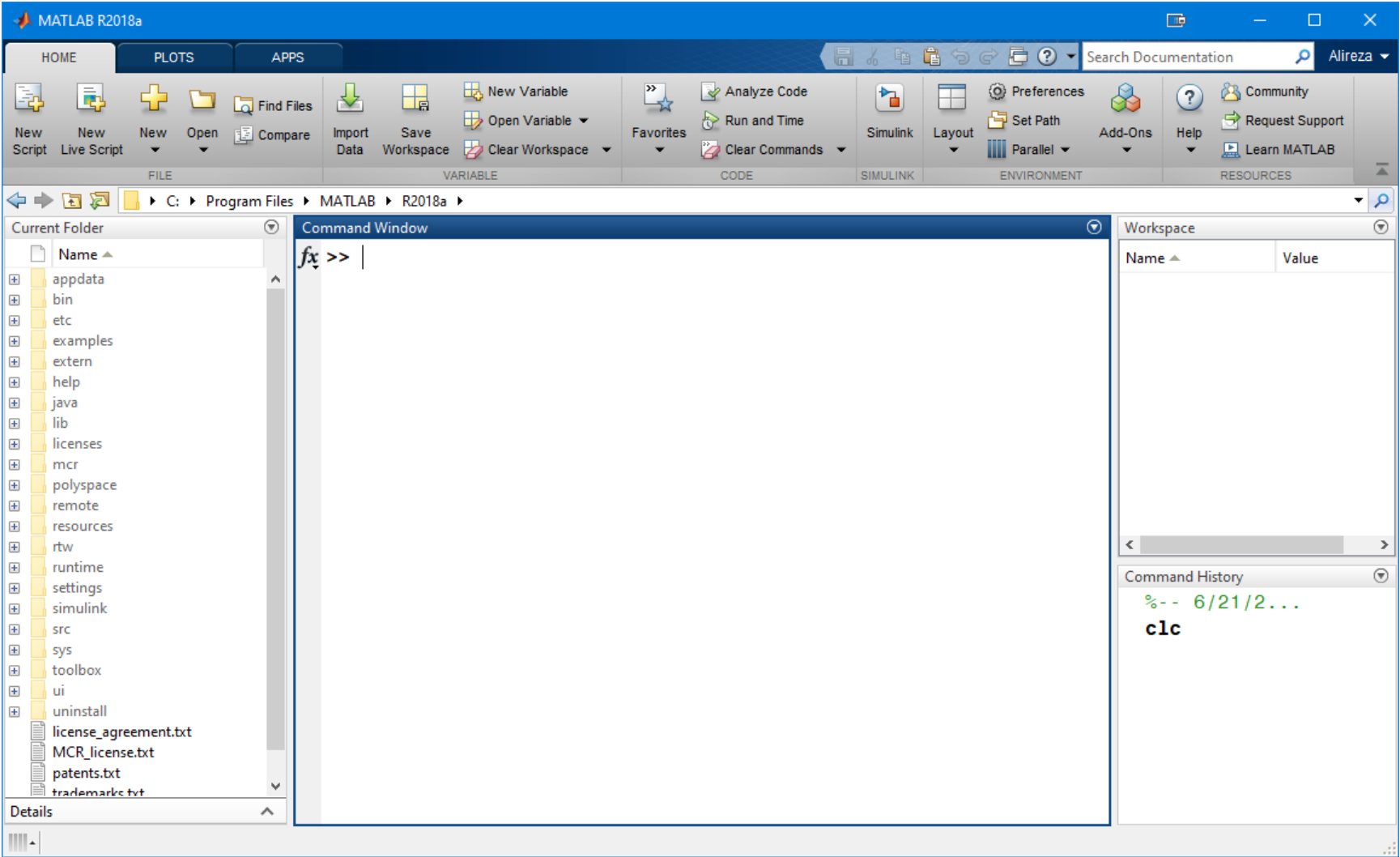
دستور ver

```
>> ver
```

```
-----
MATLAB Version: 9.4.0.857798 (R2018a) Update 2
MATLAB License Number: 968398
Operating System: Microsoft Windows 10 Pro Version 10.0 (Build 16299)
Java Version: Java 1.8.0_144-b01 with Oracle Corporation Java HotSpot(TM) 64-Bit Server VM mixed mode
-----
```

MATLAB	Version 9.4	(R2018a)
Simulink	Version 9.1	(R2018a)
Aerospace Blockset	Version 3.21	(R2018a)
Aerospace Toolbox	Version 2.21	(R2018a)
Antenna Toolbox	Version 3.1	(R2018a)
Audio System Toolbox	Version 1.4	(R2018a)
Automated Driving System Toolbox	Version 1.2	(R2018a)
Bioinformatics Toolbox	Version 4.10	(R2018a)
Communications System Toolbox	Version 6.6	(R2018a)
Computer Vision System Toolbox	Version 8.1	(R2018a)
Control System Toolbox	Version 10.4	(R2018a)
Curve Fitting Toolbox	Version 3.5.7	(R2018a)
DO Qualification Kit	Version 3.5	(R2018a)
DSP System Toolbox	Version 9.6	(R2018a)
Data Acquisition Toolbox	Version 3.13	(R2018a)
Database Toolbox	Version 8.1	(R2018a)
Datafeed Toolbox	Version 5.7	(R2018a)
Econometrics Toolbox	Version 5.0	(R2018a)
Embedded Coder	Version 7.0	(R2018a)
Filter Design HDL Coder	Version 3.1.3	(R2018a)
Financial Instruments Toolbox	Version 2.7	(R2018a)
Financial Toolbox	Version 5.11	(R2018a)
Fixed-Point Designer	Version 6.1	(R2018a)
Fuzzy Logic Toolbox	Version 2.3.1	(R2018a)
GPU Coder	Version 1.1	(R2018a)
Global Optimization Toolbox	Version 3.4.4	(R2018a)
HDL Coder	Version 3.12	(R2018a)

MATLAB محیط



Command Window

Command Window

```
>> A = (25*3)+11
```

```
A =
```

```
86
```

```
>> B = sqrt(2)
```

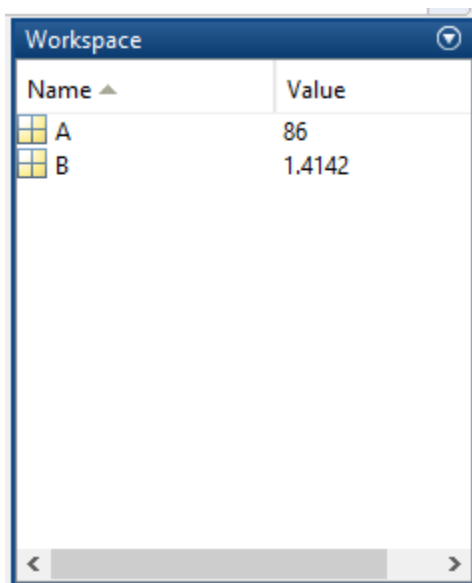
```
B =
```

```
1.4142
```

```
fx >> |
```

❑ در پنجره Command می‌توانیم دستورات خود را نوشته و سپس با فشار دادن کلید Enter از کیبورد، نتایج اجرای دستورات را در همین پنجره مشاهده کنیم.

Workspace

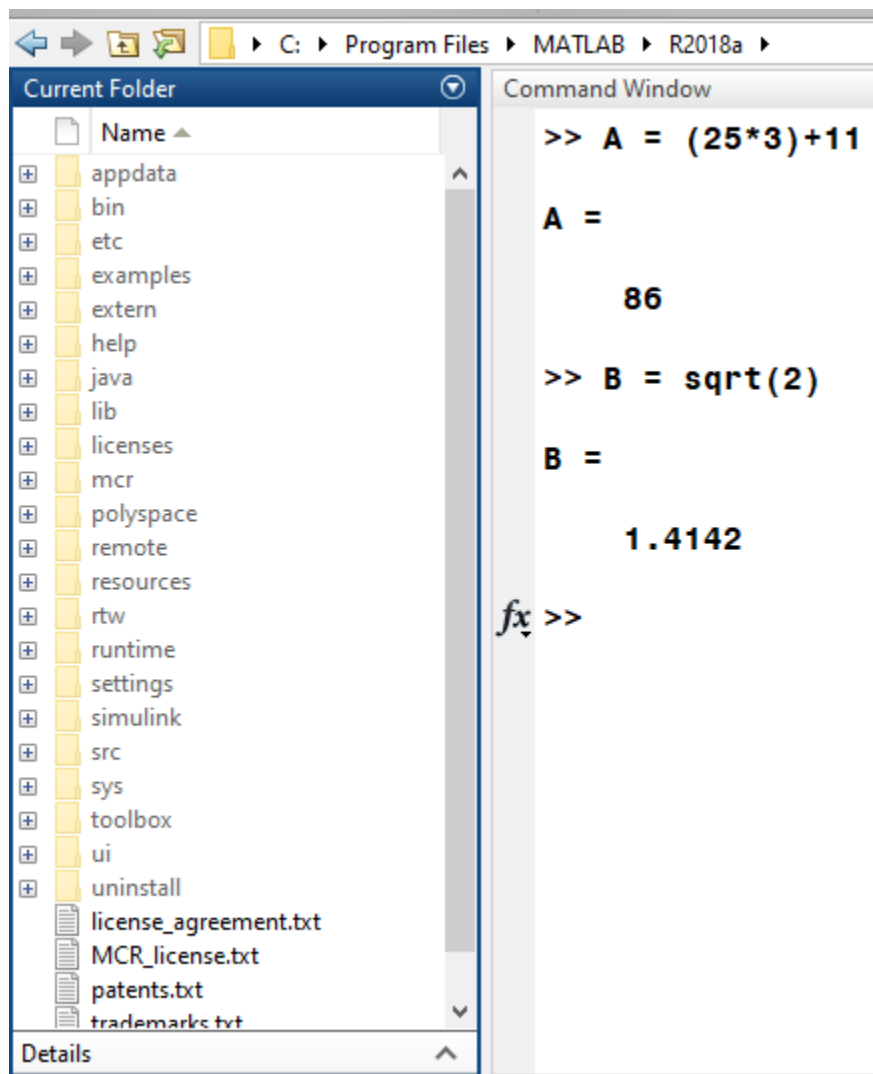


The screenshot shows the MATLAB Workspace window. It has a title bar 'Workspace' with a dropdown arrow. Below the title bar is a table with two columns: 'Name' and 'Value'. There are two rows of data: 'A' with value '86' and 'B' with value '1.4142'. Each row has a small icon to the left of the name. At the bottom of the window is a scrollbar.

Name ▲	Value
A	86
B	1.4142

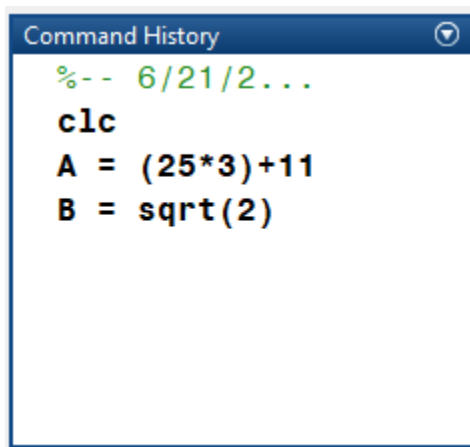
❑ در پنجره Workspace لیستی از متغیرهایی که به وسیله دستورات در متلب تعریف شده است، نمایش داده می شود.

Current Folder



در پنجره Current Folder ☐
می توانیم پوشه ای که در
آن فایل های متلب مورد
نظرمان وجود دارد را
مشاهده کنیم و به آسانی
پوشه و یا فایل های مورد
نظرمان را بیابیم.

Command History

A screenshot of the MATLAB Command History window. The window has a blue title bar with the text "Command History" and a small downward arrow icon on the right. The main area is white and contains a list of commands. The first command is "%-- 6/21/2..." in green. The second command is "clc" in black. The third command is "A = (25*3)+11" in black. The fourth command is "B = sqrt(2)" in black.

```
Command History
%-- 6/21/2...
clc
A = (25*3)+11
B = sqrt(2)
```

□ در پنجره Command History لیستی از دستوراتی که در متلب اجرا کرده ایم، نمایش داده می شود.

MATLAB

انواع داده‌ها

داده‌های عددی در MATLAB

انواع داده های عددی به شرح زیر می باشند :

Numeric Types	Description
double	Convert to double precision
single	Convert to single precision
int8	Convert to 8-bit signed integer
int16	Convert to 16-bit signed integer
int32	Convert to 32-bit signed integer
int64	Convert to 64-bit signed integer
uint8	Convert to 8-bit unsigned integer
uint16	Convert to 16-bit unsigned integer
uint32	Convert to 32-bit unsigned integer
uint64	Convert to 64-bit unsigned integer

داده‌های عددی در MATLAB

مقایسه داده های عددی مختلف با یکدیگر :

Class	Max value	Min Value	Bytes
logical	1	0	1
int8	127	-128	1
int16	32767	-32768	2
int32	2.14E+09	-2.14E+09	4
int64	9.22E+18	-9.22E+18	8
uint8	255	0	1
uint16	65535	0	2
uint32	4.29E+09	0	4
uint64	1.84E+19	0	8
single	3.4E+38	-3.4E+38	4
double	1.79e+308	-1.79e+308	8

داده‌های عددی در MATLAB

Command Window

```
>> x = int8(32.6)
```

```
x =
```

```
    33
```

```
>> y = int8(3000)
```

```
y =
```

```
   127
```

```
>> whos x
```

Name	Size	Bytes	Class	Attributes
x	1x1	1	int8	

داده‌های عددی در MATLAB

Command Window

```
>> intmax('int8')
```

```
ans =
```

```
127
```

```
>> intmin('int8')
```

```
ans =
```

```
-128
```

```
>> z = single(54)
```

```
z =
```

```
54
```

```
>> whos z
```

Name	Size	Bytes	Class	Attributes
z	1x1	4	single	

داده‌های عددی در MATLAB

```
>> realmax('double')
```

```
ans =
```

```
1.7977e+308
```

```
>> realmax('single')
```

```
ans =
```

```
3.4028e+38
```

```
fx >> |
```

ثابت های MATLAB :

pi : عدد پی را با دقت بسیار بالا ارائه می دهد.

Command Window

```
>> pi
```

```
ans =
```

```
3.1416
```

```
>> whos ans
```

Name	Size	Bytes	Class	Attributes
ans	1x1	8	double	

```
>> whos pi
```

```
fx >> |
```

ثابت های MATLAB :

i : ثابت موهومی یا $\sqrt{-1}$ است که در اعداد مختلط ثابت موهومی را تشکیل می دهد. عدد مختلط را می توان بدین گونه تعریف کرد $x + iy$ که x قسمت حقیقی، y قسمت موهومی و i همان ثابت موهومی است.

Command Window

```
>> i
```

```
ans =
```

```
0 + 1i
```

```
>> whos ans
```

Name	Size	Bytes	Class	Attributes
ans	1x1	16	double	complex

ثابت های MATLAB :

```
Command Window
>> sqrt(-1)

ans =

           0 +          1i

>> whos ans
  Name      Size      Bytes  Class  Attributes
  ----      -
  ans       1x1         16  double  complex

>> i^2

ans =

    -1

>> whos ans
  Name      Size      Bytes  Class  Attributes
  ----      -
  ans       1x1          8  double
```

ثابت های MATLAB :

inf : بی نهایت، بیشتر در جواب های متلب دیده می شود به عبارتی جواب از محدوده شناخته شده متلب خارج است

```
Command Window
>> 250^250

ans =

    Inf

>> whos ans
      Name      Size      Bytes  Class  Attributes
      ans      1x1           8  double

>> whos inf
>> 1/0

ans =

    Inf
```

ثابت های MATLAB :

NaN : مبهم، Not a Number

در صورت وقوع $\frac{0}{0}$ مشاهده می شود.

Command Window

```
>> 0/0
```

```
ans =
```

```
NaN
```

```
fx >> |
```

فرمت های رایج MATLAB :

□ برای مقایسه کامل در قسمت Help متلب واژه format را جستجو کنید.

Style	Result	Example
short (default)	Short fixed decimal format, with 4 digits after the decimal point. If you are displaying a matrix with a wide range of values, consider using shortG. See Display Large Data Range in short and shortg Formats	3.1416
long	Long fixed decimal format, with 15 digits after the decimal point for double values, and 7 digits after the decimal point for single values.	3.141592653589793
shortE	Short scientific notation, with 4 digits after the decimal point. Integer-valued floating-point numbers with a maximum of 9 digits do not display in scientific notation.	3.1416e+00
longE	Long scientific notation, with 15 digits after the decimal point for double values, and 7 digits after the decimal point for single values. Integer-valued floating-point numbers with a maximum of 9 digits do not display in scientific notation.	3.141592653589793e+00
shortG	The more compact of short fixed decimal or scientific notation, with 5 digits.	3.1416
longG	The more compact of long fixed decimal or scientific notation, with 15 digits for double values, and 7 digits for single values.	3.14159265358979
shortEng	Short engineering notation, with 4 digits after the decimal point, and an exponent that is a multiple of 3.	3.1416e+000
longEng	Long engineering notation, with 15 significant digits, and an exponent that is a multiple of 3.	3.14159265358979e+000

فرمت های رایج MATLAB :

□ برای مقایسه کامل در قسمت Help متلب واژه format را جستجو کنید.

Style	Result	Example
+	Positive/Negative format, with +, -, and blank characters displayed for positive, negative, and zero elements.	+
bank	Currency format, with 2 digits after the decimal point.	3.14
hex	Hexadecimal representation of a binary double-precision number.	400921fb54442d18
rat	Ratio of small integers.	355/113

DispType	Result	Example
compact	Suppresses excess line feeds to show more output in a single screen. Contrast with loose.	theta = pi/2 theta = 1.5708
loose	Adds linefeeds to make output more readable. Contrast with compact.	theta = pi/2 theta = 1.5708

فرمت های رایج MATLAB :

Command Window

```
>> format shorte
```

```
>> 10^10
```

```
ans =
```

```
1.0000e+10
```

```
>> 6^42
```

```
ans =
```

```
4.8123e+32
```

فرمت های رایج MATLAB :

Command Window

```
>> format bank
```

```
>> pi
```

```
ans =
```

```
3.14
```

```
>> format rat
```

```
>> pi
```

```
ans =
```

```
355/113
```

فرمت های رایج MATLAB :
