

برنامه نویس، روش برای دستور دادن به کامپیوتر است و زبان برنامه نویس زبان است که وسیله آن آمیلار انجام می شود.

زبان C مبینه زبان بازیز (صفرویک) است، اما به دلیل طاقت فرمایودن این زبان ارزان هاست مفسر (مانند C++) با کامپیوتر (مانند پایتون) استفاده می شود. وظیفه کامپیوتر و مفسر تبدیل کردن زبان سطح بالا به سطح پایین به متغیرهای ایجاد است سطح زبان برنامه نویس که نشان دهنده هر دوری یا نزدیک آن ب زبان بشر است، به سطح بالا، متوسط و پایین دسترسی می شود. در شورا در حقیقت زیر ویژگی ها، مزایا، معایب و نوشتهای از هر کدام از سطوح زبان بیان شده است:

**ویژگی ها:** قابلیت انتزاع، نزدیکی بودن به زبان بشر، داشتن قابلیت خوانای بیشتر، انجم مدیریت حافظه  
مدیریت مستقیم بر ازونه ب صورت خودکار

**مزایا:** آسانی در یادگیری، امنیت بیشتر، مناسب برای برای دبیریم

**معایب:** سرعت پایین تر، کنترل سخت تر حافظه و ساخت افزار، نامناسب بودن برای برنامه نویز

**شود:** #، پایتون، جاوا، جاوا اسکریپت، روی

**ویژگی ها:** داشتن قابلیت انتزاع، خواندن شرح توسط ماشین، دور بودن از زبان بشر، نیاز به مدیریت حافظه

**مزایا:** سرعت بالاتر، کنترل حافظه و ساخت افزار، مناسب بودن برای برنامه نویز

**معایب:** ساخت در یادگیری، امنیت پایین تر، نامناسب بودن برای دبیریم

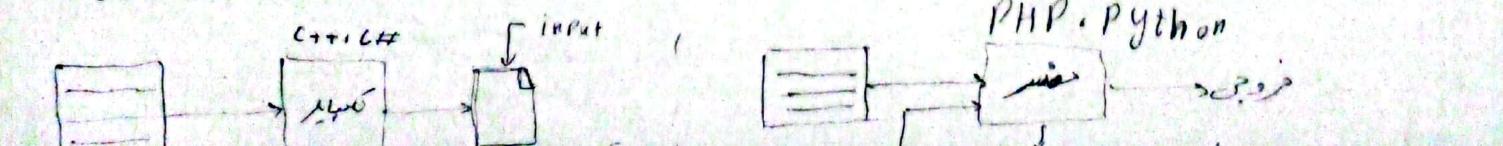
**شود:** بازیز، اسکریپت

**سطح متوسط** [ویژگی ها]: داشتن ویژگی از هر دو سطح دیگر مثل قابلیت انتزاع و مدیریت حافظه

**شود:** C، C++

به طور کلی زبان های برنامه نویس به دو دست چند منظوره (CPL) و خاص منظوره (DCL) دسترسی می شوند.

**کامپیوتر و مفسر:** وظیفه این دو تبدیل یک زبان سطح بالا به زبان بازیز می باشد تا سیستم بزرگ آزار نماید (مانند متوجه) مدل آن کنند



۱- جلوگیری از آنکه درست نمایش خطا ب خط ترجیح نماید و لیکن کامپایلر این را بخواه ترجیح کرد و دیگر مایل تغییل نماید.

۲- وابسته به سیستم محاسبه: زبان کامپایلر وابسته به سیستم محاسبه است.

۳- سرعت دستگاه استفاده از هارددیسک هر دازده: سرعت کامپایلر بینتر و زبان کمتر از RAM CPU را تغییر نماید.

۴- خطای ابر: دو زبان مفسوس راحت تر است.

: هر زبان برنامه نویس حساسیت خاص نسبت به نوع داده دارد. به این حساسیت و رفتار خاص زبان برنامه نویس به داده Type checking می تواند Type checking

۱- استاتیک تایپ (Static Type): با یعنی نوع داده را مشخص نمی نماید.

Java, C, C++, C#

مشخص نماید نوع داده Type checking بر دو دسته تقسیم شود

۲- داینامیک تایپ (Dynamic Type): نیازی به مشخص نمایدن نوع داده نیست.

JavaScript, PHP, Python

۱- قوی (Strong type): این زبان ها روی نوع داده و عملیات آنها

بسیار حساس اند به عنوان مثال اجزه ای جمع یک عدد با یک رشته را نمی دهند.

C++, Java, Python

زبان های برنامه نویس بر دو دسته دیگر نیز تقسیم شوند

۲- ضعیف (Weak type) (لاقیتاً بطلان تایپ قوی) نیست.

JavaScript, PHP, C

با رایم برنامه نویس: درست نمایه به معنای شیوه و نحوه برنامه نویس است.

۱- اعلان، انتبار (declarative Programming): برای انجام عملیات دستور را مشخص نمایند و سیستم

۲- دستوری (Imperative Programming): برای انجام عملیات باید تمام مرحله را مشخص نمایند.

۱- برنامه نویسی بر دستار (رویار) (Procedural Programming)

با رایم دستوری

۲- برنامه نویسی شیگرا (OOP) (object oriented Programming)

۳- پارالل پردازی (Parallel Processing): یک برنامه را طور پنهان کرده بتوان هر زمان

روز چند هر دازده اجرا شود که سرعت ۶۰۰ برود.

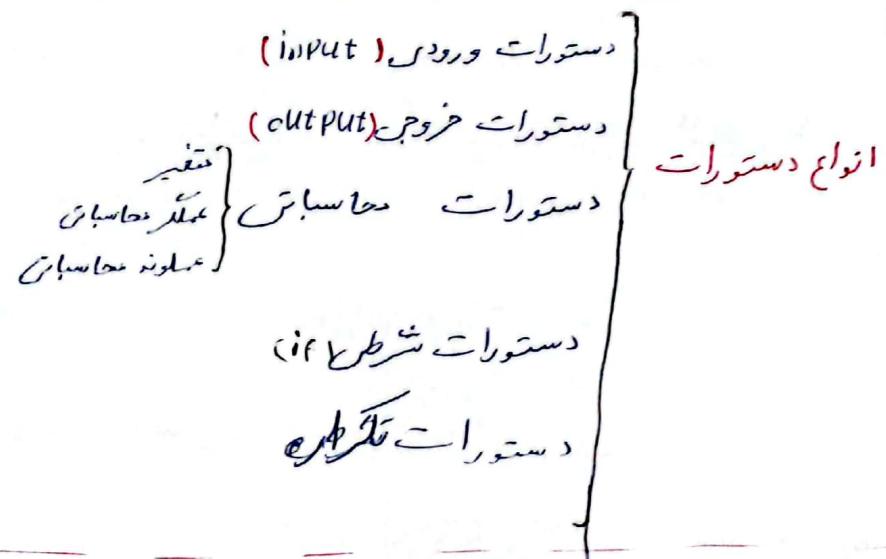
1 - برنامه نویسی منطقی (Logical Programming) : برمایه منطق ریاضی است که راست و از یکسری فرضیاتی برخوردار نتیجه می‌رسیم. مثلاً: مادان آدم است. آدم فارسی زن است. پسر سان مخلوق مرد. برای حل مسائل منطقی مانند چالش بیان روش را در می‌گیریم. (یعنی هر چیزی که می‌گوییم باید بتوانیم تذکر است)  $\text{C}^{\text{H}}\text{L}$  : پروژه.

۲ - رایانه نویسی تابعی (Functional Programming) :

هر تابع و تابعی کاری خاص انجام دهد. یعنی

3 - برنامه نویسی گنجینه داده (Database Programming) : داده معتبر می‌باشد به عنوان مثال می‌توانیم: داده‌هایی بافته و تجزیه را انتخاب کنیم.

**آلгорیتم:** تعریف گل میگل اید عملیاتی برای حل مسئله! ترتیب مشخص که متناهی بوده و دارای یک نقطه شروع و یک نقطه پایان است. (آلgoritم ممکن است چندین بار تکرار شود)



ملوچارت: بیان آلgoritم برای تسلیل.

○: ترمیناتور (Terminator) : شروع و پایان.

خط جریان (Flow line) : انتقال مراحل بین گذگاری.

فرایند (Process) :  $\boxed{N_{243}}$  دستورات.

تصمیم (Decision) :

داده (Data) :

مارک سخت (Hard Skill) : مهارت های الگازم

مهارت های نرم (Soft Skill) : مهارت های اختیاری مانند کارکرد های استثنایی

1 - برنامه نویس منطقی (Logical Programming) : برچاپی ریاضی است که بر اساس قواعد معمولی حل می شود. معمولی آن است. آدم خارج شوند. پس سایان خواهد بود. برای حل سایل معمولی می شود.

چنانچه باید رسود. (بینهایم هشت مصنفوں ترتیب است) C# :

برای این اعلان

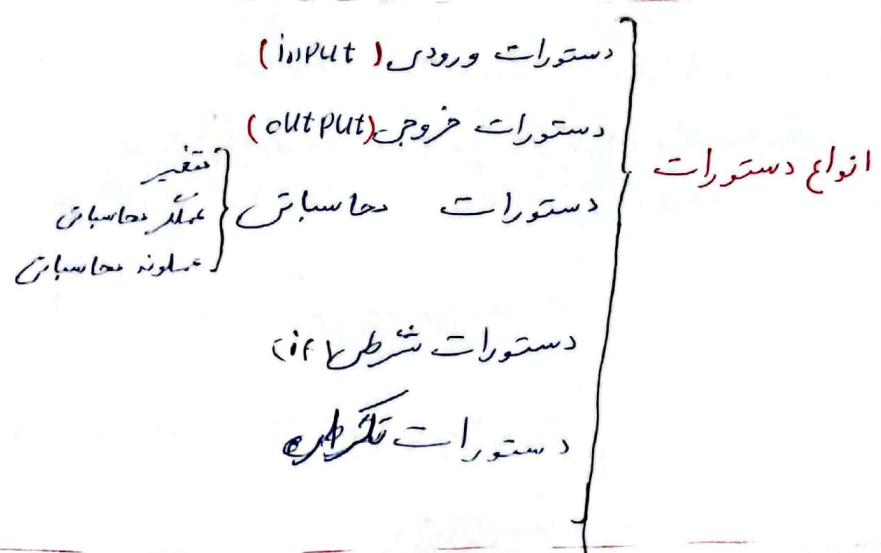
2 - برنامه نویس تابعی (Functional Programming) : برنامه را تابعی می نویسند تا بتوانند تابعی را تابعی نویسند و کشید.

هر کام و تابعی کار را خاص انجام دهد. بینهایم

3 - برنامه نویس پایه داده (Database Programming) : داده مخصوص باشد بینهایم مثال

من توییم: داده هایی با نام و نیز را انتخاب کن.

آلوریتم: تعریف گردن یک عملیات برای حل مسئله با ترتیب مشخص که ممکن است بود و دارای یک نقطه شروع و یک نقطه پایان است. (آلوریتم ممکن است چندین بار تکرار شود)



فلوچارت: بیان آلوریتم به شکل.

Terminator (Terminator) : شروع و پایان.

خط مریان (flow line) : انتقال راهی بین دو دستورات.

فرایند (Process) : دستورات.

تصمیم (Decision) : تغییر شرط.

داده (Data) : داده.

نحوه ایام میم

مهارت سخت (Hard Skill) : مهارت های ارثی

مهارت نرم (Soft Skill) : مهارت های اختیاری مهندسی کارهای باشندگان

مترادهای مارتین  
 سخنان در جمع - فوکل - خلاصت - تطبیق پنیر - خیریت -  
 اربابات - کارتوس - مقاومت - هوش میجان - نوشته قرفاودا  
 بازار طایب - سازماندهی - همکاری - حل مسند - رزوه سازی

1. مفسر و IDE خود پایتهو: هتلام نصب ہائیون پنچب خود و مانند IDE است.
2. خط فرمان و ترمیم: متنظر Power Shell و Cmd خود ویندوز میباشد.
3. Note Pad: کم های این نویسیم، سپس پسون فایل را عوض کنیم.
4. IDE (Integrated Development Environment): تمام ابزارهای مور نیاز برای یک زبان برنامه نویسی را فراهم میکند. (نایتا ۲ یا ۳ بجای ۵)
5. Code editor: از جنده زبان برنامه نویسی پشتیبانی مکنند برای حمله زبان افزونهای سوردندر را دنبه کنیم. (حجم کیترن نسبت به IDE دارد) مانند ویسکوڈ ویسکوڈ
6. Note Book: مانند نوشته کی کتاب گویا است.
7. Web: مفسرها آنلاین درکوچک.

## محیط های کمز

کوح، ارکوش موبایل  
 نیز اپے هایر  
 برائی کمز  
 وجود دارد.

## شروع کمز

Syntax: شواهد خاص در زبان برنامه نویسی.

سطر فیزیکی (Physical): سطرهای کد در هر محیط برنامه نویس، (مشهد شاریکنار شده)

منطقی (Logical): سطرهای کد مفسر آنها را تشخیص می دهد. شکل از زیر معرفت (تعداد سطر را

ewPrintl (Seni colon): برای جو اساسی سطرهای منطقی در انتشار آن قرار میکشد. (";"؛ Print("all"; all; out); all)

\ (Back slash): برای نوشتن اراده دستور در سطرفیزیک بعد. وقت که را را برای آنکواد و مانند اینست نیازی نیست از آن است.

PEP: توصیه های از طرف پایتهو برای بسته شدن کد (اجبری نیست).

معناده های ۷ عدد داشت که تعداد ۷ لکھ های هر سطغ فراخواست: ۷ عدد باشد.

کامنت (Comment): از عالمت شارپ `#` استفاده می‌کند توسط مفسر نادیده گرفته می‌شود.  
فقط جای برخای خوبی ها فایل خواهد بود.

بین ۳ کوتیشن (`'''` - `'''`) نوشته می‌شود، برای تابع دلایل هماور دارد.

استفاده قرار گیرد. برخلاف کامنت توسط مفسر خوانده می‌شود. من طبعاً با تابع Print نشان داده می‌شود. سکانر برای نمایش داده خواهد بود. آن در صورت تابع دلایل ها نوشت نشود تفسیر شود.

باک بند و تردیک نوشت رخ از کدهای خط بعد پیکوئی که از دستگاه اصل جلد را باند.

ورودی خروجی  $\xrightarrow{\text{in}} \boxed{\text{pr}} \xrightarrow{\text{out}}$   
پیکوئی را چه اعمال روی ورودی.  
ورود (input): گرفتن اطلاعات از `input` (استفاده می‌شود).

خروجی (output): برای نشان دادن ورودی پردازش شده کاربر از تابع Print (استفاده می‌شود).

متغیر (Variable): مکانی از حافظه هست که یک نوع داده خاص را در خود ذخیره می‌کند. بعنوان مثال متغیر int مکانی از حافظه بود که اعداد صحیح را در خود ذخیره می‌کند.

ID: تابع برای نشان دادن آدرس متغیر در حافظه است. (Print(id(a))

del: دستوری برای پاک کردن متغیرها است.

متغیر (identifier):

۱. بحروفه بزرگ و کوچک توجه شود (Case Sensitive).
۲. نام با عدد شروع نشود.
۳. از امفورا `all` تا `llyis` استفاده نشود.
۴. تاحد ممکن از `A-Z` (آلفا) و `I-I` (آلفا) استفاده نشود اچو در بخش سیستم ها به صورت `ایران نایاب داده` می شوند.

قوانين نام‌ذاری تساسه

۱- عمل های حسابی:  $\frac{+}{-}$  خانه قسمت صحیح // خانه باقیمانده %

$$\left. \begin{array}{ll} < & = \\ > & \neq \\ \leq & > \end{array} \right\} \text{مطابق معايير} - 2$$

مطابق ای عکسها باشد است  
با غلط

$$e_2 \cdot u_1 = 2 \text{ or } u_1 \cdot u_1 = 2$$

$$\begin{array}{lcl} / = & & \\ * = & + = & \\ ** = & - = & \end{array} \quad \left. \right\} \quad \text{with this - 3}$$

انواع معلم در operator

**and :** بار دست بدیں باید ہر دو طریقہ دست باشد  
**or :** بار دستے بدیں اُئرگیں دست باشد  
**Not :** عالم امیرش رائشن م دھد.

5- عبارت های عضویت با درست است یا نه  
 in : بررس وجود داشتن یک عضو در یک جمیع  
 Not in : بررس وجود نداشتن یک عضو در یک جمیع

**ex**  $\begin{matrix} x=5 \\ y=5 \end{matrix}$   $x$  is  $y \rightarrow$  out:True : is }  
**ex**  $\begin{matrix} x=5 \\ y=5.0 \end{matrix}$   $x$  is not  $y \rightarrow$  out:True : is not } تفاوت ملائمه - 6

: (and) &      }  
: (or) |      }      عبارات مترافق

(%R)  $\wedge$

(Not) ~

1

5

1

WALKS Mrs : 8

بیت (bit) : کوچکترین واحد حافظه که فقط می تواند ۰ و ۱ در خود ذخیره کند.

بايت (byte) : هر ۸ بیت یک بايت است.

عبارت : حداقل یک مقدار تولید می کند و باید ارزیاب شود (ارزیابی نتیجه عبارت است) شناسنامه مارت است

دستور : یک دستور را انجام می دهد و ممکن است دارای خود پاسخ یابشد، اختصار آزاد یک دستور شامل جمله می باشد.

Set .4 مجموعه های با خصوصیات غیر ممکن در گفته شده

String -1

List -2

Tuple -3

(None) Special -6

integer -1  
float -2  
complex -3 } Numeric ( اعداد )

Dictionary ( ترتیبی )

True -1  
False -2 } Boolean .3

انواع داده در پایتون

داده های عددی

int(x)  
float(a)  
complex(a)

این نوع داده ها ۳ نوع دارند که هر کدام می تواند یک تابع builtin باشد.

این توابع یک پیور و معرفت عدد را به نوع خود در می آورند.

توضیح: در پایتون مرتکان (رقم یک عدد را با -) ازهم جدا کرد. مثل ۱-۰۰۰-۰۰۰-۰۰۰

تابع round : عدد را رد مکن. ( ۰ تا ۵ )

Pow ( عدد و عدد ) : قدرت .

تابع sep : دو str را ازهم جدا مکن.

سیو کام ار ۷ ربر قوت و برجسته کارکرد ملایم دهد.

که مساحت یک مثلث را محاسبه نمود.

: ۲

۴- جدا کردن از گام پنجم

۵- تبدیل درجه به دایگو

۱- تبدیل Kg به گرم

۲- مساحت مثلث

۳- محاسبه اعمال اصلی روزی

بروکس

: ۲

n = int(input('عدد'))

%n% / ۱۰

Print(u)

n = n // ۱۰

%n% / ۱۰

Print(u\_2)

n = n // ۱۰

%n% / ۱۰

Print(u\_3)

عمل نتیجہ

مشت (String): دنباله ای از شکل کارکرد می‌باشد.

کارکرد: هر حرف یا رقم یک کارکرد مشت می‌باشد.

نحوه حفظ یا این مثلث، رشت با دایلکوئیشن و کارکرد با کوئیشن نیاش داده می‌شود، ولی در باقی هنر مرق ندارد.

برای لغویک استور استفاده می‌شود.

مثال: ۲ بخش رشت تابع دستورات آنچه کارکرد را خیرفهال نمود. اعمال در رشت می‌باشد.

اندیس (index): شماره لیاری کارکردهای کارکرد که از سمت چپ از صفر داشتار است (از ۱ شروع می‌شود).

ex: a[4-۳-۲-۱] a[۰] + a[-۴] = ۱  
۰ ۱ ۲ ۳

اندیس درون [ ] نوشته شود

تعویچ: فناوری خالی بین کارکردهای اندیس اندیس دارد.

e: a['Sam an'] → out: 'ama' → تکینیز (slicing)

آخرین کارکرد از اول  
رشت حساب نمود.

ex: s: (' Saman ') → out: ۵  
Print(Len(s))

تابع Len(): طول رشت را مشاند (۵).

تعویچ: نهایی خالی نیز جزو طول رشت می‌باشد.

تابع ord(): بازگشتی که کارکرد که آسن آنرا تحویل می‌دهد. تعویچ: کارکرد کارکرد را داخل کوئیشن قرار دهید.

متد (UPPER) : حروف کو جک انجام را ب حروف بزرگ تبدیل می کند.

ex: a='Saman'  
a.a.UPPER()  
Print(a)

متد (LOWER) : حروف بزرگ را کو جک می کند.

ex: a='Saman'  
Print(a.endsWith('n'))

متد (.Starts with) : برعکس متد با آ است.

ex: a='Saman'  
Print(a.find('a'))

متد (Find) : اولین اندیشه که در آن ورودی متد انتخاب شده است را جایز می کند.

نکتہ: با افاده کردن از find() میتوان راست شروع یا پایان متد را پیدا کرد.

ex: a='S@man 22'  
Print(a.isalnum())

متد (isalnum) : برعکس می کند که تمامی چیزی که نباشد اصل حرف و عدد اندیشه است.

ex: a='123'  
Print(a.isnumeric())

متد (isnumeric) : برعکس می کند که تمامی چیزی که عدد اندیشه است را پیدا می کند.

ex: S='A-B-C'  
L=['A','B','C']  
Print(S.join(L))

متد (join) : این متد با دریافت یک لیست اعضا رشته اعضا رشته اصل را بهم وصل می کند.

~~join~~

متد (split) : این متد با دریافت یک رشته اعضا رشته اصل را از هم جدا می کند و صورت یک لیست تغییر می کند.

ex: S="Saman - fardin-22-Hi"  
Print(S.split("-"))

ex: S='Saman'  
Print(S.replace('n','2'))

replaced : جایگزینی دو چیز را انجام می دهد (old, new)

→ out: 'Sama2'

متده استrip(): این متد یک کارکتر را از دو طرف رشته اصل حذف مکند. با اضافه کردن + فقط کارکتر مورد نظر را از راست و با اضافه کردن - کارکتر را از چپ حذف مکند.

ex: `s = ' Saman '` } `s.strip() = 'Saman'`

`print(s.strip())`

متده capitalize(): حرف اول رشته را به حرف بزرگ تبدیل مکند.

متده count(): تعداد تکرار کارکتر ورد را در رشته اصل برسی مکند.

### 1. Escape character

۱: جایی که بکس کوئیش در فرود

۲: بخط بعدی مروود  
۳: کارکتر قبل خود را پاک مکند BackSpace

۴: جایی که بکس اسلش

۵: بعد از خود را به خط اول مبرد و خط اول را پاک مکند...

۶: بار نوشته بین کارکتر های مابین

۷: براندازه tab فاصله می دهد.

توجه: آنر قبلاً از رشته از ۲ استفاده کنید بکس اسلش لغور شود.

فرمت دهنده رشته: ۱- عبارت % - ۲- .format

Print("n is: %.1f \ny is: %1%.(3,5))  
نوع داده: `n` `y` `%(3,5)` `.1f` `%1%` `%(3,5)`

۱- استفاده از عبارت %:

نحوه: `%.2f` میان ۸ = `%.2f` عدد صحیح: `%.0f` رشته: `%.5f` کارکتر = `%.c` عدد صحیح: `%.0f` میان ۸ = `%.2f`

نحوه: `%.2f` عدد اعشاری: `%.4f` عدد اعشاری: `%.2f` (۰ کارکتر اعشاری) تا چند رقم اعشاری

سیار از این بس در آنکه در استفاده از

en: d = { 'x': 5, 'y': 8 }

Print("x is {} \n y is {}".format(x, y))

en: print("x is {} \n y is {} \n z is {} ".format("reza"))

en: print("x is {} ".format("reza"))

en: x = 5  
y = 2

f = "x is {} \n y is {} \n z is {} ".format(x, y, z)

en:  
1. تعداد دفعات تکرار شدن یک کارت در یافتن آن کارت در یک جمله

2. آخرین کلمه صدیق و رسیم را بر

3. حرفی دوچندان و رسیم آن را برای وجود راشت کلات جمله دوم در جمله اول

4. حذف کردن تمام عضاهای خالی در پشتی ورسیم

5. پاک کردن صفر از ابتداء شماره تلفن ورسیم

S = input("جمله")

en4:

i = S.replace(" ", "").replace("\t", "")  
Print(i)

S = input("جمله")

en2:

S = S.lstrip()  
i = S.find(" ")  
Print(S[i+1:])

لیست (List) : یک نوع داده از دنیاگار است که انواع داده های راسی دو گو را می ذخیره کند.

index 1 2 3 4  
L = [1, 2.34, "Saman", [P, 11]]

تابع (List) : باگرفتن جنبی و ورود آنها در قالب یک لیست ذخیره کند.

ex: List("reza") → out, ['r', 'e', 'z', 'a']  
index: 1 2 3 4

ex: L[4][0] = 7

توجه: قوانین انواع slicing در لیست دقیقاً مانند شش باشد.

آخر دو لیست را باهم جمع کنیم، لیست دوم به انتشار لیست اول اضافه شود.

ex: L = [1, 2] + L + [3, 4] → out, L = [1, 2, 3, 4]

نکت: لیست mutable است.

L[1] = 1 → out: L = [9, 1]

L[4:2] = ['a', 'b', 'c'] → out: L = ['c', 'b', 'a']

که از لیست.

با استفاده از از لیست گرفته نشود، زیرا انتساب فقط ID خلاصه را دارد.

Copy 1. برای این سیس پرمند

برای که از لیست از لیست ها در روش وجود دارد 2. استفاده از مازول که و که می توان

ex: L1 = [1, 2, 3]

import copy

مشغل روشن اول این است که سطح مرکزی

L2 = L1[:]

L2 = copy.deepcopy(L1)

و آخر لیست تودر توباند از لیست داخل که مرکز

L2 = L1.copy()

مشغل روشن 2

پس استفاده از روشن 2 سطح آشنا باشد.

ex: L = [1, 2, 3]

L.append(8)  
Print(L)

meth append: عضو ورودی را به انتشار لیست اضافه می کند.

عضو ورودی من تواند لیست نیز باشد.

ورودی حق تواند دو متدار باشد.

ex: L = [1, 2, 3, 4, 5, 6, 7]

انتساب چندگانه

a, b, c, d, E, f, g = L unpack

Pack ]

a, b, \*c = L Pack

[, c, [3, 4, 5, 6, 7]]

یک نوع تابع

مانند لیست است با این تفاوت که immutable بوده و میان هر انتزاع نتوشته شود.  $(1, 2, 3, 4, 5)$ ,  $(ALI, 1, 2, 3, 4)$

نکته: تابع در حافظه نسبت به لیست بسیار راست است. توجه: برای تابع استفاده از زیرنویس (جایگزین نمودن).

## (Dictionary)

دیکشنری (Dictionary) دنباله ای از item که شامل یک کلید و یک پارسی ارزش (Value) می‌باشد.

در زبان‌های برنامه‌نویسی شناخته شده که این دو آنکواد نوشتار شود. در زبان‌های کامپیوتری شناخته شده که نام کلید immutatble باشد.

دسترسی عناصر: برای این کار از کلید استفاده می‌کنیم.  
توجه: لغایتی که دیکشنری اندیس ندارد و ترتیب هم نیست

دیکشنری قابل تغییر mutable است.

$d = \{ 'a': 1, 'b': 2 \}$

$d['a'] = 1 \xrightarrow{\text{Print}(d)} \text{out} : d = \{ 'a': 1, 'b': 5 \}$

$d['c'] = 2 \xrightarrow{\text{Print}(d)} \text{out} : d = \{ 'a': 1, 'b': 5, 'c': 2 \}$

توجه: خروجی همچنان دیکشنری است.

لیست نیست که بجز  
برای تبدیل به لیست باید در  
تابع list() قرار بگیرد.

d.keys()  $\Rightarrow [ 'a', 'b' ]$

d.values()  $\Rightarrow [ 1, 5 ]$

d.items()  $\Rightarrow [ ('a', 1), ('b', 5) ]$

حذف عناصر از دیکشنری: به وسیله دستور del این کار قابل انجام است

تفصیل: کلیدی - مطابق با immutatble نباشد. هر کسی تغییر آن را داشت غیر مستقیم وجود دارد. ابتدا مقدار کلید مورد نظر را داخل دیکشنری جدیدی از لذایم. سپس کلید را حذف کرده و دوباره یک کلید جدید تعریف می‌کنیم.

ex:  $u = d['b']$   
 $del d['b']$

$\xrightarrow{\text{Print}} \text{out} : d = \{ 'a': 1, 'c': 3, 'd': 5 \}$

**تابع sorted :** یک فلیست را با ترتیب داشتال آنها را به طور پیشفرض به صورت صعودی مرتب می‌کند. برای مرتب کردن نزولی می‌باشد  $\text{sorted}(\text{list}, \text{reverse}=\text{True})$  یعنی از انتهای لیست تا ابتداء آن را در ترتیب معکوس می‌کند.

**تابع dict :** یک جفت کلید و مقدار را که دو قالب لیست منطقه و به صورت دیکشنری تغییر می‌دهد.

**مثال :** `dict([("a", 1), ("b", 2), ("c", 3), ("d", 4)])`  $\xrightarrow{\text{out}}$  `d = {"a": 1, "b": 2, "c": 3, "d": 4}`

**مجموعه (Set)**

دقتیقاً مانند مجموعه های دیگر نیستند. دقتیقاً مجموعه های تکرار نمی‌شوند.

**Update :** استفاده از `add()`.

**اضافه :** `add('reza')`

**Mutable :** میتوان عنوانهایی را آنها اضافه یا از آنها حذف کرد.

**حذف :** `remove('reza')`

**مجموعه های معاصر تکرار ندارند.**

**توجه :** در مجموعه های اندیس وجود ندارد. ~~چون~~ ترتیب هم نیست. همچنین در مجموعه های توان عناصر قبل تغییر مانند لیست استفاده نمی‌شود.

**تابع dict :** در ورود های در قالب مجموعه ذخیره می‌کند.

**متدهای add :** برای اضافه کردن یک عضو به Set بکار می‌رود، در این متدها تنها از لیست استفاده شود. همچنین آنکه ورود یک تابع یا رشته باشد این ورود را یک عضو در نظر نمی‌گیرد.

**مثال :** `s = {1, 2}`

**اضافه :** `s.add('reza')`

**نتیجه :** `s = {1, 2, 'reza'}`

**حذف :** `s.add((1, 2))`

**نتیجه :** `s = {1, 2}`

**متدهای update :** برای اضافه کردن چند عضو به Set بکار می‌رود. ورود عددی عددی باشد در قالب لیست باشد و همچنین آنکه ورود یک تابع باشد. عنوانی که تابع را به صورت مجدد از لیست استفاده می‌کند.

**مثال :** `s = {1, 2}`

`s.update([3, 4])`  $\xrightarrow{\text{print}}$  `out: s = {1, 2, 3, 4}`

`s.update(['a', 'b'])`  $\xrightarrow{\text{print}}$  `out: s = {1, 2, 'a', 'b'}`

`s.update((3, 4))`  $\xrightarrow{\text{print}}$  `out: s = {1, 2, 3, 4}`

برای حذف یک عضو `Set` وجود داشت `remove()` متد.  
ex: `S = {1, 2, 3}` } Print, `out: S = {2, 3}`  
`S.remove(1)`

برای حذف یک عضو `Set` وجود دارد، آن ورودی در `Set` وجود نداشت باشد، خود `Set` را نشان می‌دهد.  
ex: `S = {1, 2, 3}` } Print, `S = {1, 3}`  
`S.discard(2)`

(intersection) & : اشتراک

(symmetric difference) ^ : تفاصل متقابل

هم اعضاً دو مجموعه  
با هم اشتراک

- : تفاصل (difference) `Set`

| : اجتماع (union)

(super set)

: زیرمجموعه (subset)

۱- آر芬 نام ~~و~~ یک داشجو در قالب یک لیست و محاسبه می‌نماید نزالت.

۲- آرفن کامل پاسخ به تلفن آذکاربر و ذخیره کردن ارقام در قالب دیلشتر شد.

۳- آرفن دو لیست آذکاربر و جایه جاگرد خانه‌ها آخر همه لیست اول با لیست دوم.

۴- آرفن دو نام داشجو و دو نزدیک آنها و ذخیره کردن شان در قالب دیلشتر.

۵- آرفن یک بین از کاربر و محاسبه تعداد کاربرها استفاده شده غیرنگاری و نایاب آنها.

داده‌ی نوع `None` آنرا از نوع داده‌های دیگر نبینه، از این نوع من باشد

تابع (`bool`) ساخت از کند که ورودی درست تفسیر داده‌ی بولین (`boolean`) آنها دو مقدار دارد `True` و `False`.

ترجمه تابع داده‌ای دنباله‌ای خارجی `False`، `nonetype` معنی مشروطه، غیر از اینها باقی `True` معنی مشروطه.

ترفند: ترفتن چند ورودی در یک خط

%: input('Enter a List:').split()

با عبارت از

ترشید و اینها را می‌توان با دادن چنین نوع داشت که باشد درست ۳۱. بجای اینجا (۱) می‌توان از import typing:Union[] استفاده کرد.

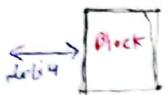
ex: from typing import Union

%: Union[int, float] = 5, 6 or, u: int | float = 5, 6

{ شرط (Conditional) } دستورات کنترل { تکرار

دستورات شرط

Syntax: if شرط :



else : شرط  
↓  
Block

یک دستور یا بلک دستورات زمان اجرام شرکت که به شرط برقرار باشد.

elif شرط

else : شرط  
↓  
Block

توجه: اگر if درست باشد، else، elif، else همها را از هم جدا نمایند.

Syntax:  
if شرط:  
↓  
Block  
elif:  
↓  
Block

من تمام در دستورات شرطی از شرط های تو در تو استفاده کرد.

- ۱- تشخیص زوج یا فرد بودن عدد ورودی کاربر  
۲- ترفتن عدد ورودی از کاربر و مشخص کردن عدد کوچکتر



تابع (min): تابعی که صورت پشت سرمه (چال) آن را کوچکترین آنها را تابعی می‌دهد. ورودی من تواند به صورت لیست یا تابل باشد.

آخر ورودی بصورت لیست و تابل خواهد بود، تراهن آزار کوچک استفاده کرد.

ex: min(1,2,3) print → out1      min([ ] و default=2) print → out2

تابع (max): بزرگترین ورودی را تابعی می‌دهد.

تابع (sum): یک ورودی بصورت لیست، تابل یا مجموعه ترفته و مجموع انسیا را انشان می‌دهد. آخرین اختیاری Start = نیز وجود دارد که جمیع ورودیها را مقادیر Start که جمع کند.

## دستورات مدار

### حلقه (Loop)

ساختن Loop:  
Loop  
بازگشتی

**while**: کامپیوچر که شرط برقرار است کارشون  
دو نوع ساختار برای حلقه وجود دارد  
**for**: در این ناچیز مشخص کارشون  
دانه و بیانیان

- 1- اعداد زوج  $\{ \dots , 10, 8, 6, 4, 2 \}$  چاپ اعداد که هر ۲ عدد پیش بیانی باشد
- 2- چاپ یک مثلاًت با دریافت تعداد سطر و گزینه دخواه آرایه
- 3- چاپ کردن مقسم علیه های عدد ورودی
- 4- معلوم کردن آیینه عدد ورودی عدد کامل است یا غیر
- 5- چاپ کردن برش جملت سرس فیلمها

- |   |  |   |
|---|--|---|
| <b>break</b> : با اجراء این دستور از حلقة خارج می شویم (اصطلاحاً حلقة شکسته می شود) | <b>continue</b> : با اجراء این دستور پس دور بعدن حلقة می شویم و ادامه دستورات اجراء نمی شود. | <b>else</b> : در پایه دو حلقاتی می خواه این نشست هنوز اتفاق نباشد |
| <b>درستور های</b>   |  |   |

### حلقه های تودر تو

برنامه نویس بسته به نیاز از تواند درون به ذی یک حلقه حلقاتی دیده تعریف کند. توجه لطفاً که حلقات داخلی باید تمام شود تا  
خط بعد برود.

مرجع: پانوچن حلقه دری دیکشنری، متغیر دخواه به  $\text{الیف}(\alpha)$  اشاره دارد.

### رینج (Range)

برنامه از اعداد که حلقات **for** دری داشته باشد، این بازه ای می تاییم که از عدد داخل بازی باشد.  $2, 4, 6, \dots, 10$   
این بازه اعداد میں بین تواند باشد، معمولی تامها را **range** خوب نمی بینیم است.

: enumerate() کتابخانه عددی درودس

1- فایل تریل عددی درودس

2- مقدار کسری برای عدد

2- چاپ مدلش از اعداد

مس

**Syntax:** `for i, j in zip(L1, L2):` zip کرد . for i, j in zip(L1, L2):  
Print(i, j)

برای مدل کردن `for i in reversed():`, reversed - 1  
ترجیح از دسته مردانه در حل دسته استفاده کرد  
`for i in sorted():`, sorted - 2

شب  
رنوم و اعداد تصادفی

درباره سواره سانده پیشنهاد و رزبکار را ایجاد کند، همچنین وغیره نیازمندیم که اعداد تصادفی تولید کنیم.

**Random:** یک عدد رندوم (بین صفر و یک) تولید می‌کند. این مدل زمان سیستم را به عنوان ورودی آن ترفته و یک عدد تصادفی تولید می‌کند.

برای عرض کردن بازه‌ی عدد از متده Uniform استفاده می‌کنیم. این متد دو ورودی ترفته که محدوده بازه‌ی آنند. تباذه حر

**ex:** `import random`

`random.random()` **print** → out: یک عدد تصادفی بین ۰ و ۱

`random.uniform(0, 5)` **print** → out: یک عدد تصادفی بین ۰ و ۵

**randint( ):** یک عدد صحیح بین بصورت رندوم انتخاب می‌کند، از تواند یک بازه‌ی دلخواه اند و عنوان ورودی بگیرد.

**ex:** `import random`

`random.randint(0, 5)` **print** → out: ۰ ۱ ۲ ۳ ۴

**randrange( ):** با استفاده از این متد تواند یک حکمت راندۀ مشخص کرد.

`random.randrange(0, 5, 2)` **print** → out: ۰ ۲ ۴

**choice( ):** یک عدد تصادفی از داخل یک لیست دلخواه انتخاب می‌کند.

**ex:** `import random`

`random.choice(a)` **print** → out: ۱ ۲ ۳

ساده Sample : در این مازول (random) دو ورودی دارد که اصل و دو نتیجه است. سپس به تعداد عدد ورودی داده شده (n) از لیست اصل که تکلیف دهد.

ex:  $\text{out} = \text{random.sample}([1, 2, 3, 4], 2)$

import random

random.sample( $n, 2$ ) Print, out,  $[1, 2, 3]$  یک زیر لیست از اصل

حدت shuffle : یک لیست را به صورت تصادفی برمزنده یعنی ترتیب یک لیست را بهم مزنده.

ex:  $\text{out} = \text{random.shuffle}([1, 2, 3, 4])$

import random

random.shuffle( $n$ ) Print, out,  $[2, 4, 3, 1]$  انتشار

1 - در یک بازدید چندبار تغییر آید و چندبار خطا

2 - با ازایختن یک تاس ۶ مرتبه چندبار آید

```
import random
random.randint
tas = [1, 2, 3, 4, 5, 6]
for i in range(100):
    tas[randint(1, 6)] += 1
print(tas)
```

## تابع (Function)

تعریف : دستاورد است که ورودی رفته و با انجام پردازش روآن یک خروجی تحویل دهد. INPUT → Process → OUTPUT

```
def name(x):  
    def name():  
        return
```

: Syntax

صادرات (call) : یک تابع زیان کاربرد آن را از اینجا برئیم، برای این کار، نام تابع را نوشته و ورودی تابع را در چندین شخص می‌کنیم.

ex:  $\text{name}(\text{ورودی})$  (ورودی)

return: متغیر محاسبه شده را برگرداند و بجهت دستور صادر کنند تا قرار دهد. برای آنکه مقادیر محاسبه شده ذخیره شود، باید صادر کنند تابع

1 - در این یک متغیر ذخیره شوند. (ورودی) = name(ورودی)

لذت: آنکه تابع جیزیز را برگرداند (return) نداشت باشد، به صورت خودکار none را برگرداند.

در رایج بد از لدی لیسی return کامپ موقوف شود.

۲۰ وردهن مهار تابع هتلام تعریف تابع پارامتر و به هتلام صادرین تابع آرگومان گویند.

بهره امت که در تابع بیشتر از 20 خط که نه آشنا ناشد و همچنین برای هر فرضیه تابع جدا تعریف کنند.

۱- تا بعل که در مدد آگرفته اگر می چند راتن یعنی تکرار قصه، شیوه ای می تکرار قصه چندبار در چند راتن تکرار شده است.

2- تابع که حاصل عبارت مقادیر محسوس است.

۳- تابع  $\theta$  چند عدد رفت و بزرگترین آنها را بیابد.

```

def fact(n):
    f = 1
    for i in range(1, n+1):
        f *= i
    return f

def sum_fact(x):
    sum1 = 0
    for i in range(1, x+1):
        sum1 += fact(i)
    return sum1

```

: ex<sub>2</sub> | : ex<sub>1</sub>

```

def repeate(number, digit):
    count = 0
    while number > 0:
        if number % 10 == digit:
            count += 1
        number //= 10
    return count

number = int(input('أدخل رقمك: '))
digit = int(input('أدخل رقمك: '))

```

ex:  $\det n(a, b)$ : هر آرکیوان - ترتیب یک پارامتر مرتبط باشد: Normal

en, def n(a,b): بطور کامپیوچر جان ارسن ارگونومند تراویر لیرد.  
[ ]

$n(a=u, b=y)$

٣- ترتيب ٢ وا: ابتدأ حالت Normal و سبع نام متقارن، متقدار دهني في لعنم.

- ex: def N(a, b, c):  
 [ ]  
 N(\*[x, y, z])
- \*iterable - 4  
 این دنباله UnPack شده و مرضی بیک پارامتر هم ترتیب ارسی اختصاص نماید. مثایل این حالت شامل دیلشنی نیستند.
- ex: def n(a, b):  
 [ ]  
 n(\*\*{'a': x, 'b': y})
- \*\*dictionary - 5  
 ارزش ها آرگمن ارسالی می شوند. در این حالت ترتیب سمت نیست.
- دک استرینگ (doc string)  
 تعریف نوشته ای که بین سه کو تیشن یا دایلکتیشن **در اولین خط بدن تابع تعریف شود** گاربر تابع را مشخص می کند.
- متد docstring : توضیحات در مورد تابع (راز مردم) :-
- ex: def name(type: int) -> tuple:  
 Block
- خروجی تابع : خروجی تابع را می توان لیست کرد.
- موجودیت (ستاد هنگامی) است که همه کارهای آن انجام داد. مثل آن عنوان ارجمنان به تابع ارسال شد. First class
1. ب صورت ~~پوچ~~ (function) می توان آنرا بگیرد ساخت. (عنان آنکه یک تابع بینانه باشد باشندگان داخل به زیر پایه آنها تعین شوند)
- ویژگی های تابع 2. می توان آنرا یک متغیر اختصاص داده first class
3. کامبینیت فرستادن یک تابع به عنوان ارجمنان را دلار است (پارامتر تابع یک تابع دیگر باشد)
4. می توان عنوان نتیجه آنرا از تابع برداشت (return) یک تابع به عنوان نتیجه یک تابع دیگر return شود
5. دارای متد هایی می باشد.
- نات: در یک متد توابع از نوع `first class` است.

فضای نام (Name Space): برای دسترسی بهتر و جلوگیری از تداخل نامها از این ساختار استفاده می‌کنند.

حوزه نام (Scope): شامل چند فضای نام می‌باشد.

- |              |   |
|--------------|---|
| 1. Local:    | به محدوده کاربرید تابع نویسید که مقصرها فقط داخل تابع شناخته شوند.      |
| 2. Enclosed: | در تابع تدریجی، حاوی توابع داخلی نام است.                               |
| 3. Global:   | تبار محدودی نیست برای همه است. مقصر در تابع خارجی مانند را شامل می‌شود. |
| 4. built-in: | تبار سیستم پایتون شامل هر تابع خارجی را دراست.                          |
- از اعضا فضای نام  
"حوزه نام"

ارسال با مقدار (Pass by Value): یک کپی از آرگومان ارسال به تابع ایجاد می‌شود و بعد مقدار فعلی تغییرات نمی‌کند.

ارسال با ارجاع (Pass by Reference): خود آرگومان اصل به تابع ارسال می‌شود و تغییرات روی آرگومان اصلی اخراج شود.

ترجمه: در پایتون تبار آرگومان‌ها به صورت ارجاعی ارسال می‌شوند.

نکته: آرگومان شرط immutatble باشد، ارسال به تابع مقدار است (یعنی یک آرگومان مثبت).

## Lambda توابع

توابع کوچک و کوتاه که سریعاً نیشت منشون و بلطفاً قابله بکار رفتن روند، این توابع معمولاً یک بار معرفت آن و اسنون شارند.

**Syntax:** `def Lambda [arguments]: expression`

`a = Lambda x: x ** 2`

`print(a(5))`

این تابع را من توانم یک مقصر اختصاص داد.

**MAP:** دو ورودی مبینه کرد اولیه تابع و دویی یک لیست است، این تابع عنصر لیست را بدلیل می‌کند به تابع خروجی داده و می‌آزد.

**Example:** `L1 = [1, 2]`

`def func(n),  
 return n**2`

`list(map(func, L1))`

اصلاح کردن عنصر آنها را جایگزین می‌کند.

درست

`L1 = [1, 2]`

لذت: در ورودی هاں تابع map بہ جان سے آنکھ کیک تابع بیرون تعریف کر دے وسیع نہ آزاد ورودیں نہست، **غایب** کیک تابع تعریف کرو.

ex: map(Lambda n: n\*2, li)

تابع filter: ورودی هاں آن میں map رپا شد. عناصر لیست را تابع می فرستہ، اگر تابع True را برداہ عنصر باقی رہاندہ، وار

ex: def func(n): برلست عنصر حذف مریشو.

```
if n > 5:  
    return True  
else:  
    return False
```

Li = [1, 2, 3, 4, 5, 6]

new = list(filter(func, Li))

Print(new)

→ out: new = [9, 11]

توجہ: مانند مثال تابع map بہ جان تابع خارج

می خواہ از تابع استفادہ کرو.

سند Reduce: مانند دو تابع قبلی ورودی مکرر بالین تفاوت کے بارے داریں تابع 2 بارہیں باشد. پہلے زیر میں ملکہ:

1	2	3
---	---	---

$1+2=3$   
 $3+3=6$

from functools import reduce  
n=reduce(lambda x,y: x+y, [1,2,3])  
Print(n)

→ out: 6

تابع sorted: این تابع می خواہیک اگرچاں دم مکرر و طبق آگرچاں دم کر کیک تابع اسے لیست را مرتب کئے.  
لذت دو مرد sorted: این تابع می خواہیک اگرچاں دم مکرر و طبق آگرچاں دم کر کیک تابع اسے لیست را مرتب کئے.  
توجہ لئیے کہ می توانم این تابع دخواہ را خود میں تعریف کئیم.

(Iterator)

اتار کردن (iterate):  
عمل تکراہ تو سطح While و for انجام می شود.

قابل اتار (iterable): بتماس اشیائیں کہ قابلیت اتار داریں و می توانند ایتیور تبدیل شوند. مانند لیست عا و تمار اسی طرح

اتار کر (iterator): اشیائیں اند کہ آخری حالت خود را حفظ فر لئند و من ٹکان روس آنھا منتظر، اجرا کر کر دیں۔

تہیں لئے، پڑھو

ex: i = iter([1, 2, 3])  
Print(next(i))

→ out: 1

دکوراتور (Decorator) : در فنست ب معنی تغییر لسته هر باشد. در واقع تابع دیگر که بر این ورودی تابع

ex: def deco(func):  
 def inner(\*args, \*\*kwargs):  
 if y == 0:  
 print('warning!')  
 else:  
 func(\*args, \*\*kwargs)  
 return inner

@deco  
def dec f(x,y):  
 print(x/y)  
f(5,6)

دکوراتور

تکیه تابع م باشد تغییر اس اعمال نکند.

خوب ب جای دکوراتور @ م توان تابع را صادر کرد و تابع را ب عنوان آرگومان ارسال کرد.

کام ارت است دکوراتور باید جویی نوشته شود که تعداد آرگومان های تابع دلخواه باشد. برای اینکار قبل از هر امر اول \* و قبل از پارامترها

من کذا بایم.

ب طور ملاصه ارسال یک تابع ب عنوان آرگومان به یک تابع دیگر است.

دکوراتور

1 - یک دیکشنری داشتم که از این بزرگ ترین دلیل پسورد م باشد. تابع دیگر که یوزرنیم را فراخواهد پسورد را چاپ می کند.  
دکوراتوری تعریف کنید که یک بلک لیست بگیرد و پسورد های یوزرنیم ها را خالی بلک لیست را نشان ندهد.

c1

2 - دکوراتوری که زمان اجرای یک تابع را نشان م دهد.

from time import PerfCounter  
from functools import wraps  
  
def time\_counter(func):  
 @wraps(func)  
 def inner(\*args, \*\*kwargs):  
 start\_time = PerfCounter()  
 value = func(\*args, \*\*kwargs)  
 end\_time = PerfCounter()  
 runtime = end\_time - start\_time  
 return inner

ex2

: ex1  
| Passwords = ["ali", "1234", "reza", "7911"]  
| blackList = {"ali"}  
| def ban(func):  
| @wraps(func)  
| def inner(\*args, \*\*kwargs):  
| if args[0] in blackList:  
| print("blocked")  
| else:  
| func(\*args, \*\*kwargs)  
| return inner

import functools

def decorator(func):

@functools.wraps(func):

def inner(\*args, \*\*kwargs):

# Do something before

value = func(\*args, \*\*kwargs)

# Do something after

return value

return inner

**زیارتور (زیارت):** اگر مکنه که های مابینه ترا نوشته شوند. و یعنی یک تابع (آترزیارت) که از دستور پکش زیارتور می‌سرد از این زیارتور مانند ایتریتور بوده و با نوشتن next و آن غافر بعدی آزاد است آورده.

مکنه، زیارتور تقبل است، یعنی غافر بعدی را می‌سازد و در خود ذخیره نمایند (خلاف راده‌های دیگران). از این رو مدیریت حافظه بسیار دارد.

توجه در توابع زیارتور به جای return `yield` استفاده می‌کنیم. توابع زیارتور بسیار از اجرای `yield` صبری لسته‌ای دستور next دارند، چنانچه تابع بادام دستورات بسیار `yield` را دارد.

تفاوت `return` و `yield` این است که

تابع با `yield` قابل‌گمراحت است.

ex: def generator():  
    for i in range(10):  
        yield i\*\*2

g = generator()

print(next(g)) → out: 0

print(next(g)) → out: 1

1: زیارتور را می‌پند.

2: بحث خواهد شد.

3: send()

گروتین: مجموعه‌ای از دستورات عمل‌ها برای انتقال ورودی‌ها به خروجی‌ها است. بطوریکه به صورت همین

ورود و خروج داشت باشیم.

رفتاگ روتنی در ریزآتور

```
def my_gen():
    print("start")
    for i in range(10):
        name = yield i
        print("my name is", name)
```

```
g = my_gen()
```

```
print(next(g))
```

```
print(g.send(reza))
```

1 - ریزآتور ساخته شده  
2 - ریزآتور کاربرد  
ex

```
def SPL_gen(Splitter=" "):
    print("start")
    s = None
    while True:
        line = yield s
        s = line.split(Splitter)
```

```
-2 | def cen_gen(words):
      |     Print('start')
      |     w = None
      |     while True:
      |         word = yield w
      |         if word not in w:
      |             w = word
      |         else:
      |             w = '*' * len(word)
```

```
| g = cen_gen([' ', ',', ' '])
```

```
| next(g)
```

بهر تابع یک صفت دلخواه میتوان طبق مثال زیر اضافه کرد.

CN:  
def ave(Li):  
 return sum(Li) / Len(Li)

ave.name-School = "Shahed"

Print(dir(ave)) → بصفة تابع نام در سیزده اضافه شده است  
Print(~~ave~~.name-School-name) → OUT: Shahed

تابع Setattr: از طریق این تابع نیز برای صفت را اضافه کرده و مقدار میگیرد.  
ex: Setattr(ave, "name-School", "Shahed")

تابع Getattr: تابع دیگر برای دریافت صفت.

تابع has attr: برای ایندیکاتور تابع صفت را دارد یا نه. مقدار Bool را برگرداند.  
CN: has attr(ave, "school-name") صفت

محض حذف یک صفت از تابع {  
1- استفاده از تابع آناده  
2- دوست دست

1- برای اینکار از تابع delattr استفاده میکنیم این تابع دو مقدار میگیرد. اول نام تابع و دو ثانی صفت.

ex: delattr(ave, "School-name")

2- باید مثال توضیح می دهیم:

CN: del ave.name-School  
کلیدی در پایتون نام صفت

ترابع بازیست: توابع اند که در باخته خود، خود را هم زنده مساخته به شکل زیر است:

```
def recursive():
    ...
    if stop-condition:
        ...
        return ...
    ...
    recursive()
```

recursive()

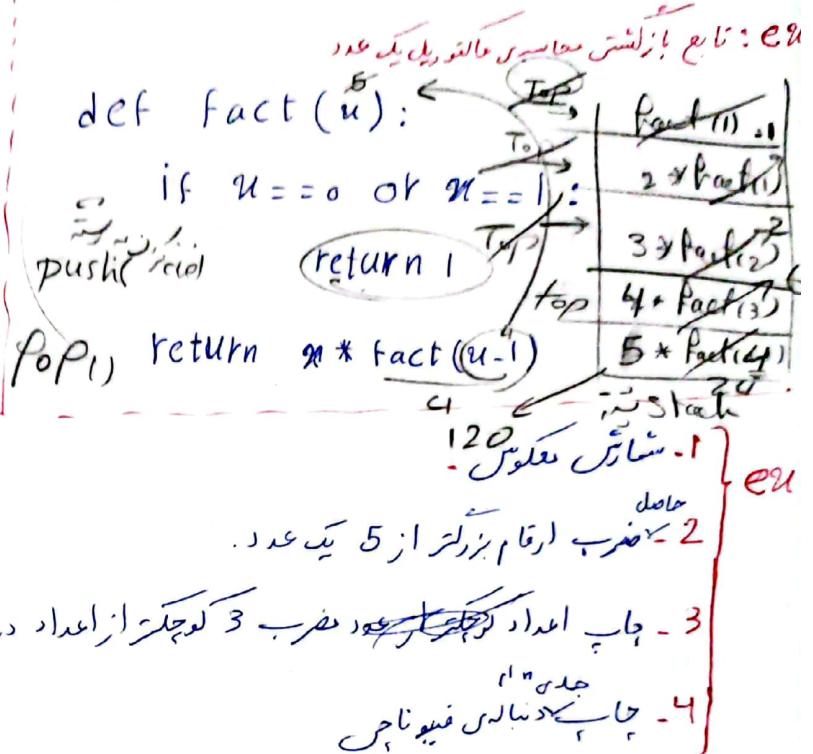
تجویز: آرین خواهد داشت تابع چیزی را برگشت  
که باید برازش شرط تابع، نیزه را خارج  
نماید، برای شرط تابع، نیزه را خارج  
نماید، برای شرط تابع، نیزه را خارج

```
def mul5(n):
    if n == 0:
        return 1
    elif n % 10 < 5:
        return 0 + mul5(n // 10)
    else:
        return n % 10 * mul5(n // 10)
```

```
def fib(n):
    if n == 0 or n == 1:
        return n
    return fib(n-1) + fib(n-2)
```

```
def memorize(func):
    memory = {}
    @wraps(func)
    def wrapper_decorator(n):
        if n not in memory:
            memory[n] = func
```

۱- طبقه زکر اشغال ساخته  
۲- در برخی موارد کند عمل رکنند (نسبت به تابع عادی)



-۱

```
def rev(n):
    if n == 0:
        return None
    print(n)
    n -= 1
    rev(n)
```

-۳

```
def func(n):
    if n < 3:
        return None
    elif n % 3 == 0:
        print(n)
    else:
        func(n-1)
```

نحوه: در عرض کافی اخواص تابع بازیگش دکوراتور اجرا می شود.

لذتگیر (memorization): باید این سرعت تابع بازیگش

```
det memorize(func):
```

```
    memory = {}
```

```
@WarpS(func)
```

```
def warapper_decorator(n):
```

```
    if n not in memory:
```

```
        memory[n] = func(n)
```

```
    return memory[n]
```

```
return warapper_decorator
```

۲: چهارم جلسه ۱۰ آم دنیالس فیلم نامه،

به چار آنده هر بار ۴۵ دقیقه در بالا معاشر شود

پس از معاشر شدن عدد در دیگر شرکت ذخیره

شده و در نتیجه سرعت بالاتر آید.

متدها

متدها (ارداد)

`divmod()`: دو ورودی عدد و آگرفته و حاصل  $\frac{a}{b}$  و سپس  $a \% b$  را برگرداند. (جوابها بصورت یک تابع است)

`Pow()`: دو ورودی آگرفته و ورودی اول را بترتیب درس می ساند. من توانم یک عورودی سوم نیز بگیرد که باقی بانده تو آن بر عروض سوم است

`round()`: عدد ورودی را گرداند. ورودی دعم آن تعداد رقم اعشار را نیاز است و دهد. - طور پیش فرض بعد صحنی روی می بلند.

`abs()`: قدر مطلق.

`A.S.-integer-ratio()`: انتسبت یک عدد را به صورت یک تابع برگرداند. مثلث ۱۷۵ هر شود  $\frac{3}{4}$  (به صورت  $(3, 4)$  نمایش داده شد)

`bit_count()`: آن عدد به صورت یک تابع برگرداند. نشان می دهد چند بیت ۱ دارد. هنلا عدد ۱۰ به صورت یک تابع

`imag()`: قسمت موهوس یک عدد مختلط را نشان می دهد.

`real`: قسمت حقیقی عدد مختلط را نشان می دهد.

`Conjugate`: مزدوج عدد مختلط را نشان می دهد.

برخی از متد های لیست: `from_bytes()`

آرایه های صاد و دوس `byteorder` باشند.

`hex()`: کد `float` را به مینیمیت `hex` در برمیگرداند.

`from_hex()`: کد `hex` را به عدد `float` تبدیل میکند.

`is_integer()`: کد `int` هست یا نه. امکان بررسی کرداند. (عدد اگر دو دسایت از زیر `float` باشد)

`bit_length()`: تعداد بیت هایی که برای نوشت یک عدد `int` نیاز داریم.

ستهای

`append()`: یک ورودی میگیرد. ورودی را به انتها ایست اضافه میکند.

`clear()`: تمام عناصر داخل لیست را پاک میکند.

`copy()`: یک کپی سطحی از لیست ایجاد میکند.

`deepCopy()`: یک لیست را به عنوان ورودی درست و آنرا یک عمقی در نظر میگیرد. (ترجیح این است که از طبقه `import copy` شود)

`count()`: تعداد تکرار ورودی در لیست را برگرداند.

`extend()`: یک لیست را به عنوان ورودی درست و اضافه لیست اصلی اضافه میکند.

ex: `1, [1, 2]`

`1.extend([3, 4])` print(1) `[1, 2, 3, 4]`

(pop) : یک ورودی دارد که آن بس است. عطفاً اندیس مورد نظر را حذف کرد و برعکس آن را در پایان پیگیرد، آنچه بعد نیست را حذف نماید.

(remove) : یک ورودی میگیرد عطفاً است. عطفاً مورد نظر را از لیست پاک میکند. اگر عطفاً مورد نظر در لیست وجود نداشت ورودی دارد. در صورت که یک عضو چند بار تکرار شود، فقط اولین عطفاً را حذف میکند.

(reverse) : لیست را برگشته میکند. ولی درست `reversed`، یعنی از لیست گزینه آنرا برگشته میکند.

(sort) : خود لیست را مرتب میکند (از کوچک به بزرگ). مرتباً با آرایه احتیاطی `reverse=True` مرتب کرده بازبینی شود. نکته: مرتباً بازبینی یک تابع مخصوصی مرتب سازی را مشغول کرد. تابع را بازبینی کردن آرایه احتیاطی `key` قرار دهید.

### ~~ستهای tuple~~

(count) : یک عطفاً را به عنوان ورودی میگیرد تعداد تکرار ورودی در تابه را مشاهد.

(index) : آنکه عطفاً به عنوان ورودی گزینه و آنلین اندیس آنرا برگرداند. اگر جزئی عطفاً مشاهده شود، اندیس این آنرا برگرداند. همچنان که `Set` میگذرد `tuple` همچنان که `Set` میگذرد `clear` را کامل نماید.

(copy) : یک کپی از `Set` گیرید.

(add) : یک عطفاً به عنوان ورودی گزینه و این را به لیست `Set` اضافه کند.

discard() : یک عضو را حذف می کند. اگر عضو موجود نباشد خود set را نپیش می دهد.

intersection() : یک مجموعه بعنوان ورودی گرفته و اشتراک آن را با set اصل به صورت یک set مایل می دهد.

intersection\_update() : مانند هست قبل است با این تفاوت که تغییر روی متد اصل نیز اعمال می شود.

isdisjoint() : یک مجموعه پرعنوان ورودی میگیرد، اگر اشتراک داشت باشد (set اصل) False و اگر نداشت True برگرداند.

subset() : یک مجموعه بعنوان ورودی میگیرد، آنرا با اعضا set اصل در ورودی وجود داشت True و غیر ایشان False برگرداند.

u.issubset(y) → True

isSuperSet() : یک مجموعه بعنوان ورودی گرفته که مجموعه ورودی را بمجموعه set اصل هست بدانه.

pop() : یک عضو را بعنوان ورودی گرفته آنرا پاک می کند. اگر خود عضو را برگرداند چون دو set ها لازم است آن را بعنوان ورودی آن ندهیم، اولین عضو را پاک می کند.

remove() : یک عضو را بعنوان ورودی گرفته آنرا پاک می کند اگر ورودی وجود نداشت، ارور می دهد.

Symmetric\_difference() : اشتراک متقارن دست را برگرداند، یعنی هر چه عضو غیر مشترک است را به عنوان یک set می برد.

Symmetric\_difference\_update() : همان متد قبل است با این تفاوت که تغییرات روی set اصل اعمال می شود.

union() : یک set بعنوان ورودی گرفته و اتحاد آن با set اصل را بعنوان یک مجموعه جدید برگرداند.

clear(): دیکشنری را پاک مایل کریں .

copy(): یک کپی سطحی از دیکشنری می‌گیرد .

fromkeys(): ورودی اول آن کلید و ورودی دیگر ارزش‌های آن کلید است (یک دیکشنری جو یہ می‌سازد) ورودی لیست می‌تواند یک آریه باشد که هر کارکرد آن یک کلید را پسند نمی‌دارد اما این مورد صحیق نیست، آنرا ارزش مشخص نمی‌نماید، ارزش‌های این است

get(): ورودی اول آن کلید دیکشنری می‌باشد که ارزش آن کلید را برابر کردن، در صورت عدم وجود کلید None برابر کردن، یک عوردو دوم دلخواه نیز دارد، اگر کلید وجود نداشت، ورودی دوم را برابر کردن (به جای None) .

items(): کلید و مقدار را به صورت یک تابل درون لیست برابر کردن .

keys(): کلید‌های دیکشنری را به صورت لیست برمی‌کردن .

values(): ارزش‌های دیکشنری را به صورت لیست برمی‌کردن .

pop(): کلید را بعنوان ورودی رفته، آنرا حذف کرده و ارزش آنرا برمی‌کرده، اگر ورودی نمایم ارور می‌دهد. می‌توان یک عوردو دوم نیز داد که به جای ارور کردن آنرا برابر کردن .  

$$\text{d.pop('key')} \rightarrow \begin{array}{l} \text{کلید} \\ \text{مورد نظر برای} \\ \text{عوین شدن} \end{array}$$

popitem(): شیز بوردو ندارد، آخرین عضو dict را پاک نماید و به صورت یک تابل ارزش و کلید آنرا برابر کردن .

setdefault(): یک کلید را بعنوان ورودی رفته و مقدار آنرا برابر کرده، اگر کلید وجود نداشت آنرا با مقدار None (یا آنرا دوم) به dict اضافه نماید .

update(): یک دیکشنری بعنوان ورودی رفته و به دیکشنری اضافه کنید، اگر کلید دیکشنری ورودی تکراری باشد، مقدار آنرا در دیکشنری اصل آپدیٹ نماید، ورودی می‌تواند یک لیست باشد که درون لیست دیگر است ایا برخان از این کلید استفاده کردد .  
 count = 5

سلسلہ Walrus شیر (شیر)

نکتہ: بہتر است ایسے علّکہ بارہ پرانتز استفادہ نہ کرو.

نکتہ: جو نام اسے بنو سکے کہ تعداد نام از کار برگرفتہ و دریک لیست ذخیرہ کند.

نکتہ: ۲- تھرینٹ یک دیکشنری کے طول، جمع متصارع و میانگین عناصر کی لیست را دارا ہے۔

91. [1, 2, 3, 4]

2- روشن (الف)

u. [ ]

- روشن (الف)

while True:

d. { 'L': Len(u), 'S': sum(u), 'a': Len(u)/sum(u) } s. input("name(q for quite: ")

if s.lower() == 'q':

92. [1, 2, 3, 4]

روشن (ب)

break

u. append(s)

روشن (ب)

u. [ ]

while (s := input(" ")).lower() != 'q':

u. append(s)

}

( خلاصہ سازن لیست ہے ) ( ادوارک ) Comprehension List

normal:

ایک مثال توضیح داد، تسریع:

s. [ ]

for i in range(1.):  
s.append(i\*\*2)

Comprehension

$s = [i ** 2 \text{ for } i \text{ in range(1.)}]$

Print(s)

نکتہ: اگر حلقات تودر تبا شرعاً درود بہتر حلقات باشد نیز تفاوت ایجاد نہیں کرنے۔ اب تا جبکہ در حواہم append کیم راستہ سپس حلقات و شرط ہمارا پشت سر ہم بدون نظر رکھیں۔

تجوید: اگر مخفی امہیہ کیم تابع را append کئی، حتیٰ پرانتز کیلئے ارید۔

: برنامه جاوه جا لست در سطرو ستدن ماتریس .

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

- 2

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

- 1

```
t = []
```

```
for i in range(3):
```

```
    t_row = []
```

```
    for row in matrix:
```

```
        t_row.append(row[i])
```

```
t.append(t_row)
```

Normal

```
t = list(zip(*matrix))
```

Normal

Comprehension

```
t = [[row[i] for row in matrix]
```

```
    for i in range(3)]
```

: بخرايند که در آن شکل و فرست را تغیر داده تا سیستم متوجه آن بشود ،

Encode کرد و به تبدیل فرست تغیر یافته ب شکل اصلی آن Decoder کرد و تویند .

در واقع Encoding تبدیل رشته هار کارکرده زیاده مانند که طوری که برای ماشین قابل فهم و ذخیره می باشد .

Decoding نزد فرآیند بر عکس Encoding است .

شرط کند ارس } 1- قابل بر لشت باشد .

بررسی شناسی کرده نباشد . } 2- Decoder نیازی به کلید نباشد .

## رشت های باینری

1 - رشت های معمول ( استرینگ ) ('a')

2 - رشت باینری ( داده های باینری ) b'r' b

3 - رشت های باینری تغیر پذیر ( byte arry )

تابع () : bytes ( ) کم رشت معمول را به داده های باینری تبدیل می کند . دو آرگومان دارد که اول رشت مورد نظر و دوی نوع آنکه چه است

ex: s = bytes('a', 'UTF-8')

→ out : <class 'bytes'>

Print(type(s))

**Capitalize**: حرف اول رشتہ را بے حرف بزرگ تبدیل کر لند.

**case fold()**: حروف بزرگ رشتہ را بے حرف بزرگ تبدیل کر لند. در تبدیل، حروف بزرگ از متد lower قوی تراجمت (اسکریپٹ) کر لند.

**center()**: دو ورودی میں گیرد، اول مطول رشتہ و دوسری کا رجیمہ است. اطراف رشتہ اصل را با دو دسی دم پر کر لند.

ex: `s='Reza'`      `space`      } → --- Reza ---  
`print(s.center(11))`

**count()**: یک ورودی سرفتوں کا تعداد آن درشتہ طاری بر کر داند.

**encode()**: یک رشتہ را encode می کر لند. یک ورودی نیز گیر کر کہ فرمت encoding را مشخص می کر لند.

**ends with()**: یک ورودی میں گیرد و مشخص می کر لند کہ رشتہ با ورودی تمام تسویہ است یا نہ.

**startswith()**: بر عکس متد ends with می باشد.

**expandtabs()**: space (tab) \t در رشتہ را با space جایگزین می کر لند. یک ورودی دارد کہ اسی ورودی نشان دهنده می تعداد

ex: `s='Reza\tolati'`      } → out: Reza -- tolati      جایگزین است.  
`print(s.expandtabs(2))`

**find()**: ~~ستاروں لئے~~ کا لئے ورودی درشتہ طاری پیدا کر داند و انہیں اولین تکرار آنرا بر کر داند. اگر کامن میں موجود نہ باشد

بائسہ ۱۔ را بر کر داند.

**format()**

**format\_map()**

**index()**: یک کارکت را بے عنوان ورودی سرفته و انہیں آنرا درشتہ بر من کر داند. اگر کارکت وجود نداشتم با شم خطا من داد

**isalnum()**: بر عکس می کر لند کہ آیا تمام کا کرستھاں رشتہ حرف و عدد اند یا خیر.

isalpha(): بروز می کند که آیا تکه کارکتر های رشته حروف الفبا هستند یا نه.

isalnum(): بروز می کند که تکه کارکتر های رشته امتحانه شده که اصغر هستند یا خیر.

isdecimal(): بروز می کند که تکه کارکتر های اعداد ۰-۹ هستند یا خیر (زیر مجموعه isdigit())

isdigit(): بروز می کند که تکه کارکتر های عدد هستند یا خیر (زیر مجموعه isnumeric())

isnumeric(): بروز می کند که تکه کارکتر های عدد هستند یا خیر.

isidentifier(): مشخص می کند که رشته اصلی برای نامگذاری یک متغیر مناسب است یا خیر.

islower(): بروز می کند که آیا تکه کارکتر های رشته حروف کوچک است یا نه.

isprintable(): بروز می کند که یک کارکتر قابل چاپ است یا خیر.

ex: `S='Saman\lt fardin'` } → out: False  
print(S.isprintable())

isspace(): بروز می کند که تکه کارکتر های رشته space چاپ می شوند یا خیر (۱۰ یا ۱۶ زیر مجموعه جاپس کند)

istitle(): حروف اول کلمات رشته را به حروف بزرگ تبدیل می کند.

istitle(): بروز می کند حروف اول کلمات رشته حرف بزرگ هست یا خیر.

isupper(): بروز می کند حروف بزرگ هست یا خیر.

join(): برعکس متد join عمل می کند.

join(): یک کارکت دریافت کرد (به صورت دست) و افغان فضه اصلی را بهم پیوند می زند.

ex: `S='-'  
A=['B', 'C', 'D']` } → out: B - C - D  
print(S.join(A))

ljust(): طانه دهنده center است با این تفاوت که رشته اصلی را بجانب وسط درست چسباند اما پس از آن می بینیم (زیر مجموعه str)

lowercase() : تمامی حروف رشته را به حروف کوچک تبدیل می‌کند.

uppercase() : بر عکس متد بالا است.

strip() : یک کارکتر را به عنوان ورودی گرفت و از طرف رشته اصل حذف نماید. با اضافه کردن ۲ کارکتر فقط از سمت راست و ۲ کارکتر از این این اول متد کارکرد ورودی فقط از سمت چپ رشته حذف می‌شود.

split()

split() : یک کارکتر را به عنوان ورودی گرفت و بوسیله آن اتفاقی رشته اصل را از هم جدا و به عنوان یک لیست تحویل می‌دهد. این عدد هم به عنوان ورودی دوم نیز مقدار مطابق می‌باشد. به تعداد عدد ورودی عضو لیست جدا شوند.

maketrans() : یک جدول ایجاد کرده که یونیک کارکترها را با ناشی از آنها متن جایگزین می‌کند.

translate()

ex: s, "Rezab"  
table = s.maketrans('b', 'ا', 'ا', 'ب') → out: { 98: 223, 97: none }  
print(table)  
print(s.translate(table))

partition() : کارکتر همچنان که شدت را به عنوان ورودی من کرده، سپس یک تابع ایجاد کرده که اتفاقی عضوی اول آن کارکترها قبل از آن را بخواهد و عضوی بعد از آن کارکترها را در ورودی می‌گیرد. سپس این تابع را باشد. عضوی دوم نیز کارکر ورودی است.

ex: s, "Saman 22 fardin1"  
print(s.partition("22"))

out: ("Saman", ("22"), "fardin1")

removeprefix() : پیشوند را از ابتداء رشته حذف می‌کند.  
ex: s, "-Sam"  
print(s.removeprefix("-")) → out: Sam  
print(s.removeprefix("SA")) → out: -SA

remove suffix() : پسوند را از انتها رشته حذف می‌کند.  
توضیح: اگر در ورودی در رشته وجود نداشت؛ شده بیکار خود رشته را برمی‌گرداند - تکرار: تفاوت دو متد بالا با strip() این است که strip() تا این کارکترها ورودی (دستور تکرار) را حذف نمایند ولی این دو متد خوب، و فقط یک تکرار را حذف می‌کند.

دکتر کارت در پشت را بعنوان ورد در گرفت و با هم جایگزین نمی‌کند. ورد اس ادل لایکنست در پشت وجود دارد و ورد اس در پشت جایگزین نمی‌شود.

Indent: یک لایکنست را بعنوان ورد در گرفت و اندیس اولین تکرار آن در پشت را برخواهند. (اذا رامی توان بسته افکاربرد.)

Swap Case: حروف کوچک را به حروف بزرگ و حروف بزرگ را به حروف کوچک تبدیل می‌کند.

Z Fill: مانند متن center است، با این تفاوت که جای space با صفر برسند.

## ماژول ها و پکیج ها

برامسنویس ماژولار: ب تقسیم کردن یک ہر روزه میز ریز به ہر روزه های کوچک (ماژول های کوچک) برنامه نویسی خاتمه گیرد.

نکت: هر ماژول خصوص نام جداگانه دارد.

اسکریپت (Script): همان فایل های که در آن ها کد های نوشته شده که قرار است مستقیماً اجرا شده و کارهای انجام داده (پسوند .js)

را اسکریپت می‌گویند.

ماژول (Module): همان اسکریپت است با این تفاوت که مستقیماً اجرا نیست. بلکه با استفاده از لاسکریپت های دیگر import و استفاده بشدید. معمولاً درون ماژول تابع های (متاستاخت تئوری) نوشته شود.

پکیج (Package): بوجود این از ماژول های مرتبط به هم (که درون یک پوشش قرار می‌برند) پکیج می‌گویند.

برای تبدیل یک پوشش های سهل به پکیج کافیست یک فایل با نام `__init__.py` درون این پوشش قرار دهیم.

نکت: مثلاً درون یک پکیج، پکیج دیگر قرارداد.

کتابخانه (Library): به مجموعه ای از پکیج ها، کتابخانه می‌گویند.

فریم ورک (Framework): شخصی می‌کند که کدهای کجا بر تواند نوشته می‌شوند و ماژول های باید در کدام پوشش ها قرار گیرند.

۳- مازول ها از اختصاص (توسط برنامه نویس ساخته شود)

۱- از پیش نصب شده مازول ها

۲- باید نصب شود : (مکانی PIP install)

cmd نموده نصب در

```
from مازول import  
      (math)  
نام متد  
(Sqrt)
```

-3

```
: import . نام مازول  
      (math)  
ست مازول - نام مازول  
(math) (Sqrt)
```

-1

```
from مازول ناچیخ import  
      (math)  
نام متد  
(Sqrt)
```

-4

```
import مازول a δ دلخواه  
      (math) m m  
ست مورد نظر . نام دلخواه  
(m) (Sqrt)
```

-2

```
en: from random import *
```

نیست هر بار مازول را تولید کنیم.

```
randint.  
random
```

داده های Private: متغیرها که با اندازه داده ای نامتناهن وجود دارند و در صورت import کردن تابع متد ها، این داده های خود از شوند. بلکه باید نامنادرم بر جای ستاره درج شود.

نکته: مذکون است که مازول زیرشاخه چند یعنی باشد، برای دسترسی به مازول نامنناهن زیرمحل می کنیم: (تجزیه برای تابع روش های صدای مازول ها، یک صورت است)

ترجم: گذاشتن است که مازول را تغییر دهیم، برای آنکه مقدار داده import lib را در صورت از مازول دسترسی داشته باشد، import lib استفاده می کنیم.

استفاده از یک مازول بون آنکه قسم اجرای آن RUN شود: یک مازول به شکر نوشته شود که در صورت اجرا import کرد موارد را خود بخواهد.

mod 1

```
en: def func(*)
      print(***2) / print(--name--)*
      if --name-- == "--main--":
          func(3)
```

(آشپز)

توضیح: آنکه main = شود، یعنی خود مازول اجرا شده و در نشیده قسم اجرای RUN می شود. و در غیر این صورت، یعنی مازول اجرا شده (در حالت دیگر) و نهایت قسم اجرای RUN شود.

ماژول ها یا چه کارگرند؟ بعض اوقات یک ماژول وجود دارد و بعدها آنها را شناسایی نمی کنند. علت آن است که دعیت محل های خصوص را جذب می کند که بعیند ماژول وجود دارد یا نه.

شل ها ۱- پیش فرض ها ( دفعه های ماژول های پیش فرض است )

۲- پوشش حادث اسکریپت ( except ) ماژول در آن انجام تمهی است

۳- محل هایی که خود محیط آنها را مشناسد. سرانجام از دستور شابل استفاده می کنیم:

```
import sys
print(sys.path)
```

اضافه کردن محل جدید را مشناسایی ماژول ها: ( از دستور زیر استفاده می کنیم )

```
import sys
sys.path.append("C:/Users/سید علی حسینی/Desktop")
```

پوشش ( --Py cache -- ): نسخه کامپایل شده های ماژول ها را در خود ذخیره می کند تا نیاز به کامپایل مجدد نباشد.

مشخص کردن محل قرار گیری یک ماژول: با استفاده از دستور مقابل:

```
import os
print(os.getcwd())
```

ساخت پکیج: درون یک پوشش فایلهای پایتون با نام --init-- ایجاد می کنیم که درین پوشش تغییل به پکیج می شود. سپس ماژول های مرتبط بهم را درون این پوشش ( پکیج ) قرار دهیم.

نکته: آندر فایل --init-- دستوری نویسید شود، به معنی import کردن پکیج مرتبه دستور اجرای می شود.

خطر: اگر بخواهیم از \* در import کردن پکیج خود استفاده کنیم، باید نام تمام ماژول ها را در یک متغیر ( \_\_all\_\_ ) ذخیره کنیم. در فایل --init-- آن پکیج ذخیره کنیم.

متغیر \_\_all\_\_: این متغیر نام یک لیست است که مکمل حاوی نام اسکریپت های ماژول مورد نظر است. این لیست مشخص کننده آن اسکریپت هایی است که در محیط اصلی ( سی ماژول Python ) قابل تسامی باشند. ماژول: \_\_all\_\_ [ 'y', 'x' ] --all--

$$x = 1, \quad z = 3$$

$$y = 2,$$

دستور برای مدیریت پکیج هاست. (Package installer for Python)  $\rightarrow$  PIP

نکته: علاوه بر این دستورات در cmd خواسته شوند

PIP install PackageName

1- نصب پکیج:

PIP Uninstall Package's name

2- حذف پکیج:

PIP install --upgrade Package's name

3- آپدیت پکیج:

Python -m PIP install --upgrade PIP : آپدیت خود PIP

لیست کراس پکیج ها را نصب شده:

محیط هایز: توابع متأذل های مورد نیازیک ہر روزہ دھان نصب نمود. این محتوا دختریں کی بروزه رہ جائے.

virtualenv



نحوه ساخت: دستور مقابل در cmd وارد کنیم:

virtualenv را از طریق دستور PIP نصب کنیم:

فعال سازی محیط هایز: در cmd آدرس کار دار را منزلم در خط بعد از کاریاب افزود. دستور  $c:\users\user>cd desktop\env2\scripts & activate$  می شود.

$c:\users\user\desktop\env2\scripts>activate$

(env2) - - - - -

نکته: با این غیرفعال سازی محیط هایز فقط کافی است از کاریاب استفاده کنیم.

C:\users\user>Python -m venv env  $\rightarrow$  دستور  $\rightarrow$  env\desktop\env

ویرساخت 2:



جسته ذخیره سازی داده ها در سیستم گذاری و روش

- |   |                |
|---|----------------|
| <p>1- باینری (binary) : فایل هایی که برای اسناد قابل فرم و خواندن نیستند.</p> <p>2- متن (text) : فایل هایی که برای اسناد قابل فرم و خواندن هستند. در واقع این فایل زیر مجموعه عبارت باینری است، با این تفاوت که بایت ها طبق اسناد کد ویدیو کد ذخیره شوند.</p> | از نوع<br>فایل |
|---|----------------|

1- با استفاده از تابع `open` یک فایل مبدل (اشتابکر فایل) ایجاد می شود.

- |  |                                |
|--|--------------------------------|
| <p>2- به کد اشاره کر فایل عملیات های مورد نظر روی فایل (مانند خواندن یا نوشتنت) را انجام می دهیم.</p> <p>3- اشاره کر فایل را باید صحیح ممکن (close) کرد.</p> | مرحله<br>کار و بازگشته<br>کلیک |
|--|--------------------------------|

مرحله 1:

تابع `(open)` دو آرگومان میم دارد (این نوع آرگومان های زیادی دارد). آرگومان اول نام فایل یا مسیر فایل می باشد.

- |  |   |
|--|---|
| <p><code>r</code>: خواندن فایل (پیشفرض) متن -</p> <p><code>w</code>: نوشتنت (معنیان قبلاً پاک شود)</p> <p><code>a</code>: نوشتنت در انتها فایل (پاک نمی شود)</p> <p><code>w+</code>: نوشتنت در ابتداء فایل</p> | آرگومان دوم نوع باز کردن فایل (file mode) می باشد. از نوع<br>بران فایل های<br>مسنون (پسوند) |
|--|---|

نکته (آرگومان 1): آگر نام در میان پوشش اسکریپت بازگشته باشد، نیازی به آدرس محل فایل نمی داشته و فقط کافی است نام فایل را بزنیم.

- |  |   |
|--|---|
| <p><code>r+</code>: هم بران خواندن و هم خواندن (در نوشتنت ممکن نیست)</p> <p><code>w+</code>: هم بران خواندن و هم نوشتنت (مسئل ای قبلاً پاک شود)</p> <p><code>a+</code>: هم بران خواندن و هم نوشتنت کل انصهار فایل</p> <p><code>w+</code>: بران خواندن و نوشتنت (مسئل ای قبلاً پاک شود)</p> | نکته (آرگومان 2): دو سیوند دارد + برای فایل های مخفی که می توانند نوشتنت شود<br>و خارج از میان بازگردان که باید نوشتنت شود. |
|--|---|

- |  |                          |
|--|--------------------------|
| <p>وجود نداشت باشد: <code>Mod: های {r, r+, a, a+, w, w+}</code> خطا می دهد</p> <p>وجود داشت باشد: <code>Mod: های {w+, a, a+, w, w+}</code> خطا نمی دهد</p> <p>وجود نداشت باشد: <code>Mod: های {a, a+, w, w+}</code></p> <p>آنرا بازگردان (یکباره و مسیر بازگردان</p> | آرگومان فایل (آرگومان 1) |
|--|--------------------------|

`* encoding = "B"` : نکرهن ایندکردن متن

- |  |             |
|--|-------------|
| <p>0: باز راهنف می کند</p> <p>1: یک خطا را نمایند و از بینه ایک خطا را در فایل می نمایند</p> | آرگومان های |
|--|-------------|

- |   |                             |
|---|-----------------------------|
| <p>-: چند مقدار می کرد</p> <p>-: صاف نهضه در برخاسته باشند، است</p> | <code>bluffering = 4</code> |
|---|-----------------------------|

Errors - 5

: newline = 6	آرگومان های دیگر تابع
: close fd = 7	open

محول 2: برای اپلیکیشن ایندکس  $\{ \text{write}, \text{read} \}$  استفاده نکنید.

متدهای `write()`: برای نوشتن در فایل متن و اینستاده می شود و با پنهان و رود داشته باشد. این متدهای طور پیشفرض در استاد سطر متنی و با بدایار `\n` استفاده نمی کنند بلکه بعدها خطا بدهند. توجه کنید که این متدهای تعداد بایت های معرف را برای تراکم کردن فایل متن پیشگیری نمی کنند و تعداد بایت های استفاده شده بر اندازهای `fd` نگذارند.

ex: `f = open("file", "w")` → رشتہ `saman` اب فایل مورد نظر اضافه خواهد شد  
`print(f.write('saman'))`

متدهای `writeable()`: برای این کنترل کننده یک فایل (طبق متد `isatty` تابع `open()`) قابل نوشتن است یا نه.

متدهای `writelines()`: جو ویس این متدهای را تواند اشیاء تکراری را (مانند لیست) را بگیرد و به ترتیب که در فایل بنویسند. توجه کنید که اعضا این `writelines()` هستند که به ترتیب بیشتر سفرهم در فایل نوشته شودند. در تابع `\n` را بعنوان یک عضو در لیست جای داد تا در هر سطر یک عضو لیست نوشته شود. آن ورودی دیگری نداشته باشد که متد `close` آن را در لیست ننویسند.

متدهای `read()`: برای خواندن محتوای فایل به کار می رود. کل محتوای فایل هر بار حداقل `1KB` را بیچاره صورت پذیرفته برقرار کرده باشد که آرگومان ورودی نیز دارد که تعداد کاراکترها را که باید خوانده شوند را مشخص می کند.

ex: `f = open('file', 'r')` → `out: saman`  
`print(f.read(5))`

توجه: آن دستور خط دادم دوبار نوشته شود.  
 محتوی دستور آخر اداری دستور خط دادم را چاچ می کند.

متدهای `readable()`: برای این کنترل کننده یک فایل (طبق متد `isatty` تابع `open()`) قابل خواندن هست یا نه.

متدهای `readline()`: در هر دو یک سطر فایل را خوانده و پس از آن بخواهد. یک ورودی میگیرد که نشان دهنده تعداد بایت های دیگر سطر است توجه کنید که آن ورودی از این بایت های سطر پیش ایشان سطر را جدا نمی کند و پس از آن فضای خالی `space` پر کنند. آن ورودی کمتر از بایت های سطر مورد نظر باشند در این بعدی همان سطر را خواهد نمود لذا کارخواه آن سطر تمام نشود.

متده ReadLines(): یک لیست برگزیده که در عنوان آن یک خط از فایل است.

بافر ( buffer ) : یک حافظه میانجی موقت (میان این داده های خارجی و این داده های داخلی) است که اطلاعات را از CPU میگیرد. به عنوان مثال وقت Print رخداد چاکر ( اطلاعات ( بجا به ۰ ) از زیرگر ( printer ) به CPU بفرمودن ) باز در نوع داخل و سیستم مامل دارد.

عملیت سوم: با استفاده از close() اشارهگر را بزدیم.

لکته: در عملیت دوم هر دوی های بلافاصله نوشته شوند، بلکه ابتدا به بافر فرموده و پس از تمام شدن برنامه یا بست شدن اشارهگر فایل نوشته شوند.  
نکته: CPU → buffer → file

متده flush(): این متده محتوا را در فایل نمایش داده است و سپس باز داخل را خالی میکند. توجه کنید هر جای متده flush() نوشته شود. محتوا درون فایل نوشته شده را کاری ببست شون اشارهگر ندارد. خطا: میکن است اطلاعات متده flush() دریغ باز داخلی ذخیره شوند.

حال جای اشارهگر فایل

متده tell(): محل اشارهگر فایل را برگرداند. (نشانی می دهد که اشارهگر چند بایت را پشت سرگذشت است.)

متده seek(): اشارهگر فایل را عوض نمایند. این متده دو ورودی دارد ورودی اول محل اشارهگر فایل (طبقه بایستی) و ورودی دوم مقداری میگیرد. ۱: از موقعیت فعلی هنوز شروع به حرکت نکن. ۲: از آخر فایل شروع به حرکت کن. ۳: از ابتداء فایل شروع به حرکت کن (بیش فرض).

قانون: در فایلهای متنهای از آرایه های باید صفر باشد.

لکته: وقت mod "ambah" با append() هم باشند، متنهای اول استفاده است.

یکسری دستورات مستند که قبل و بعد دستور `with` انجام می‌دهند (باشد) کو اونر نسبیت دارند

--enter-- ۱  
دوسته دارد  
--exit-- ۲

دستور `with`: این دستور قبل از اجرا دستور `--enter--` و بعد از اجرا دستور `--exit--` را اخراج می‌کند.

`ex: with expression:` نشان دهنده دستور `with` با این دستور خالی پنجه را فراز می‌گیرد.

نکته: هر فایل یک Content manager است.

نکته: حافظه `Context` دستور `with` فعال شده، باشد `enter` قابل اجرازده و باشد `exit` آزاد نبند.

`ex: with open('file.txt') as f:`  $\rightarrow$  `ex: with open('file.txt') as f:`  
عملیات روی فایل (سروحد) یک دستور است

با این دستور قابل خود بخود بعثت مرئی و سیاری به بسته فایل می‌باشد.

نکته: من توان چند قابل را در یک خط اجرازد. مطابق دستور مقابل: `open()` و `close()` دستورات

نکته: به جای خواندن از فایل، از دوی کمپرس (چندر کرده و زیپ) برخواند.

نکته: به جای نوشتن در فایل، در صفحه نمایش می‌نویسد. (نمایش نه اند)

نکته: به جای نوشتن در فایل، در صفحه نمایش می‌نویسد.

شیوه قابل استفاده: ۱. `Stdin`  
ماژول مستند: ۲. `Stdout`  
خطیه نه اند: ۳. `Stderr`

۱ - همان ورودی ها را در چاپ است .  
مشهود . پس زدن آن space است .

۲ - file : میتوان به جای آنکه ورودی چاپ شده آزاد  
فایل ذخیره کرد . برای اینکار کافی است یک مایل وارد یک متغیر  
باشد و نام آنرا (ترکیبیش رو بجز آن) نمایم .

۳ - end : آفر چاپ را  
معنی میکند که بسطر بعدی  
بروید . پس زدن آن : ۱۷

۴ - flush : دوستان

میگیرد . False یعنی اطلاعات  
درین باز بروند و True یعنی بر میکنند آن .

### برخی متدهای فایل

file : به آن توصیف کر فایل (

truncate() : بتنداد بایت فایل را بشیخ می دهد . یک ورودی میگیرد که مشخص میکند چند کارکرده باشد و بازیگرد دارد و بقیه را حذف میکند  
( اگر بازیگرد هایی عمل نمیکنند که باشد نیشند )

seekable() : مشخص میکند که اجازه جابجا کردن (نکره) فایل را داریم یا نه .

import os

os.path.exists('file')

برای وجود را تشخیص میکند : با استفاده از دستور مقابل :

import os

os.remove('file')

حذف کردن یک فایل : با استفاده از دستور مقابل :

## ( JavaScript Object notation ) JSON طبلہ حاصل

برای انتقال داده‌ها بین سیستم‌های مختلف (جنس با سیستم عامل و زبان متفاوت) پرکارس رود. این نوع فایل بیشتر در زمینه وب کاربری دارد. عملت محرومیت این نوع فایل‌ها این است که نوع داده را حفظ نمی‌کند.

تعریف: یک قالب استاندارد بازار، اعلان تبادل داده‌ها در بین باشندگان از جفت‌های خصوصیت - کلید، فرآموده است.

نکته: جفت‌های خصوصیت - کلید مانند دلیل‌های در پایتون می‌باشد.

- ۱ - سبک وزن  
۲ - حواندن و نوشتن مدادهای  
۳ - مستقل از زبان  
۴ - قابل استفاده در آثار زبانها

- 1- داده ها در جفت های پر صورت نام اعداء رخوار میگیرند.  
 2- داده ها بکالا از یکدیگر جدا نشوند.  
 3- علامت های [ ]، آرای های رانده مارند.  
 4- علامت های { }، اشی رانده مارند.

<pre>{   'name': 'Saman',   'age': 21 }</pre>	<p>نوت‌نامه از سون</p>	<p>۱- در جا<sup>گ</sup>های اسکار اعداد از نوع number هستند</p>
		<p>۲- نوع را ده اوه! Booleans در JS با فرآورده</p>
		<p>شروع من شروع</p>
		<p>۳- معادل None پاسیون در JS است</p>
		<p>۴- در JS دیکشنری، شی ای است</p>
		<p>۵- در JS لیست ها آرایه ای است.</p>

متد (Load SC) : برای تبدیل یک رشته با قالب  $\text{REG}$  به فرمت  $\text{LD} \text{ Reg}, \text{Mem}$  وجود دارد ، ورود آن رشته با  $\text{LD}$  ممکن است

Py-type = json.loads(jn-str)

`Print(jn-str, '\n', type(jn-str))` → out: [1, 2, 3]  
<class 'str'>

```
Print(P-type, '\n', type(Py-types)) ----, out: [1, 2, 3]  
<class 'list'>
```

**مکت:** توجہ کیجئے ایں ملک رشتر کا ان را بے کیونکوں دوں شے در پائیں تھے بلیں کئے۔

متده ( ) Load() . تقدیم کننده فایل json را به گلوبال بایتون تبدیل میکند . قوچ لغتی که برای تبدیل افایل json به یار بازبیند .  
import json محتوا را

ex: with open('file.json', 'r') as jf:  
s = json.load(jf)

print(s)

→ out: None

print(type(s))

<class 'NoneType'>

متده ( ) dump() : برای تبدیل یک داده های توکن به یک شیوه JSON کاربرد دارد .

import json

ex: d = {'a': 5, 'b': 6}

j = json.dumps(d)

print(j)

→ out: {"a": 5, "b": 6}

print(type(j))

<class 'str'>

متده ( ) dump() : حینکه یک گلوبال بایتون را تا یک فایل JSON بنویسید که در آن داده نویسید .

import json

ex: d = {"c": 1, "a": 16}

نکته: لایب های دیکشنری پیشنهاد تبدیل برخودند .

with open("myfile", "w") as jf:

json.dump(d, jf) → محتوا: d = {"c": 1, "a": 16}

نکته: دندر لارن است یعنی مشخص میکند چند فاصله بین محتوا خالی بود . indent = -3

نکته: براساس حروف الفبا مرتب کند . sort\_keys = -4

نکته: یک تابع میگیرد . عنصر اول تابع را ترتیب با چیزی کند و عنصر دوم که در دندر دارا جواب میگیرد محتوا را دستگذیر میکند . separators = -5

نکته: آرایه

متدها

نکته: پسیل داده های توکن پیشنهاد میگیرد . برای پسیل Parse

import json

name = 'Saman'

ex: age = 21

ترنک: تبدیل تعداد زیاد داده های توکن به یک فایل .

b = False

d = {'name': name, 'age': age, 'b': b}

with open('myfile.json', 'w') as jf:

\*, json.dump(d, jf)

ادامه ترجمه قلبی: محتوای یک فایل JSON به داده های سیوس.

```
import json  
with open('myfile.json', 'r') as f:  
    p = json.load(f)
```

```
name = p['name']
```

```
age = p['age']
```

```
b = b['age']
```

## فایل های CSV (Comma Separated Value)

یک نوع فایل متن است که برای انتقال اطلاعات بین چند نرم افزار را بردازد.

ماژول CSV: برای خواندن و نوشتگی فایل های CSV استفاده می شود (به صورت خواکار نسبت می شود).

reader(): یک متد برای خواندن فایل های CSV می باشد. دو ورودی دارد که اول نام اشیاء که فایل را دارند و دویی کارکرده باشند.

نحوه: رول خروجی های این متد از تابع یک حلقه در مرور برآید که در آن فایل CSV اشاره دارد.

DictReader: فایل CSV را به صورت دیکشنری می خواند. (هر لینه یک مستوی است)

writer(): برای نوشتگی در فایل های CSV استفاده می شود. بالای متد اشاره کرده ایم (بجای مجموعه).

writerow(): یک لیست به عنوان ورودی دارد که این لیست یک خط فایل است.

writerows(): یک لیست دارد که هر چند لیست دیگر است هر لیست از لیست اول که خط فایل است.

```
import csv
```

```
ex: with open('myfile.csv', 'w', newline='') as cf:
```

```
    wrwriter = csv.writer(cf)
```

```
    writer.writerow(['id', 'name', 'lastname'])
```

```
    data, [[1, 'saman', 'hardin'], [2, 'ALI', 'najati']]
```

```
    writer.writerows(data)
```

توضیح بعض مفاهیم بامثال ساختن کارخانه: یک کارخانه طرح مشخص را ساخت یک ماشین دارد، به این طرح کلاس (class) می‌گویند.

و به ماشین ساخت شده از روی این طرح یک شیء (Object) یا نمونه (Instance) از آن کلاس گویند. اما این ماشین می‌توان به اطراف حرکت کرد و به ال را فشار داد که به این حرکت رفتار (behavior) می‌گویند. این ماشین بایس ویژگی‌های خود را دارد که آن خصوصیات (Attributes) گویند. توجه: رفتارها (متدهای ماشین) برگشتن و خصوصیات (نوع داده پیاده‌سازی شوند).

مثال حساب بانک: حساب بانک یک کلاس است. رفتار آن انتشار وجه و شاهده موجودی و صفات آن تمام صفات حساب بانک موجود، شاره حساب آن است.

<p>(Object oriented Analysis) OOA - ۱ : چالانهای رفتارهای و صفاتی که بر اثر آن چکار انجام دهیم</p> <p>(Object oriented Design) OOD - ۲ : نام‌گذاری صفات و تعیین رفتارهای ... چکونه انجام دهیم</p> <p>(Object oriented Programming) OOP - ۳ : مرحله پیاده‌سازی</p>	راهل برنامه نویسی شرگردان
---	------------------------------

تعجب: در این راهل قسمت کوچک از برنامه را ریزگرفته و راهل را انجام می‌دهد. سپس سراغ قسمت کوچک بعدی می‌روند.

لپسول سازی (Encapsulation): یک لپسول (دار) را در گلبلیرید، مستورات ایک لپسول در آن جمع شده‌اند. در واقع، فرایند کارهای قراردادن متد ها و خصوصیاتی که متد های آن ها کاری کنند لپسول سازی کریں. درین حس - بانک کارهای قراردادن رفتار و صفات نیز لپسول سازی است. این فرایند بایس ساخت کلاس بکار می‌رود.

بینان سازی دارد (Data hiding) : به آنکه بخوبی داده را بکسری از داده از کاربر مخفی نشونه باهد ف آنکه کاربر ندانه این داده ها را تغییر دهد، بینان سازی دارد گویند. بعنوان مثال، در حساب بانک، داده از کاربر مخفی نشونه باشد، اما این داده ها از ایش دهد.

رابط (Interface) : یک توضیح است که هر شخص باید چه کاری را انجام دهد و در انجام کارها بفرایند انجام آن وظیفه را کنیم.

## نحوه انتزاع ( abstraction ) :

یعنی توجه به کلیات بدون توجه به جزئیات. بعنوان مثال خودروییک مفهوم انتزاعی است. یعنی چیزی که چار جزء از میتوان و نمایه دارد. وقتی من در می خودم یک دلیل ماضیت به ذهن مام آمده، این بهان معناست که خودرو وجود خارجی ندارد بلکه دلیل که به ذهن مام آمده وجود خارجی دارد. در این مثال خودرو آنکه ( abstract ) او یعنی مسون واقعی ( concrete ) میگویند. لکن: هر چند جزویست که انتزاع سطح انتزاع باشد.

## خواص ها

ارث بری ( Inheritance ) : شرکت روابطها و متقرها از یک کلاس بالاتر و در عین حال داشتن خصوصیت و رفتارهای مخصوص خود.

در مثال ماضیم یعنی از کلاس ماضیم ارث بری کن، یعنی ویریش‌های مثل داشتن چار جزء، زمان ذهن را از کلاس ماضیم گیرید و در عین حال ویریش‌های مخصوص به فرد نیز دارد. در مثال دیگر یک ماهر را از نظر بگیرید، این ماهر از کلاس میتوان ویریش داد. همان ماده فنا حداچی داشته، حرر و رنگ و نمایه را بر اثر من برده، ولی خود نیز ویریش‌های مخصوص به فرد نیز دارد. ماده تنفس درآب.

کلاس ( class ) : وقتی بخواهیم به اشیاء بصورت انتزاعی ترکیب کنیم، از کلاس برمی‌بریم. توجه کنید که سطح

انتزاع بودن کلاس‌ها باهم متفاوت است. انتزاع کلاس  
1- کلاس ( class ) ( concrete class ) : کلاس‌های که قابل پیاده‌سازی هستند  
یعنی نمونه‌ری واقعی از آن را تخلیه ممکن و وجود دارند مثل ماهر.

2- کلاس انتزاعی ( abstract class ) ( abstract class ) : کلاس‌های که قابل پیاده‌سازی

نمایند: کلاس‌های از کلاس انتزاعی ارث بری می‌کنند.

هم: سطح انتزاع کلاس‌ها نسبت به هم سنجیده می‌شود.

متده انتزاع ( abstract method ) : کلاس‌های انتزاعی دارای متدهایی به نام مده انتزاعی هستند، این مدها ترتیب شوند بلکه کلاس‌های پاسخ‌گوی از آن ارث برده‌اند، این متدهای این خود شخص‌سازی می‌کنند. بعنوان مثال متدهای دستگیری در کلاس انتزاع حیوان برای هر گونه حیوان متفاوت است، بندهای حکمت، ماهر و غیره باهم متفاوتند.

ترکیب ( composition ) : جمع آوری، چند شیوه کلی باهم برای ایجاد یک شیوه جدید، برای رسایس استفاده می‌شود که یک شیوه

بعض از این دیگر باشد. برای رسایس این مفهوم انتزاع کاربرد دارد. بعنوان مثال کلاس یعنی کلاس یعنی از کلاس خودرو ارث بری کنند، ولی کلاس موقوف باشند. برای طبقه بین کلاس موقوف چنین وغیره که باعث بوجود آمدن کلاس یعنی از کلاس می‌شوند، همچنانکه کلیس گویند.

1- : aggregation  
روابط سی اسپار

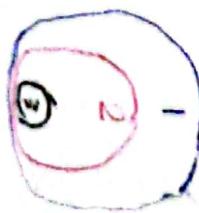
1- : association  
روابط سی اسپار  
درینہ . پاٹنہ رابطہ بین راندہ موہاشیں । (شہ اول یعنی ازشہ دوم نیست )

- 2 : aggregation  
روابط سی اسپار

بین دو شے دوستی نہیں زیاد وجود نہار . یعنی ارشن اول تاہر شود شئ دوہ تاہر نہار .

میں رابطہ بین بالک و میل اسٹے . باستہ ایڈس بیٹھ دیا شیں اسٹے .

Q-38



: composition  
رابطہ بین دو شے کیے آنے آنے عالم پیدا کیتے اسٹے . اسکے نہیں تاہر شود دیکھیا جائے . میں تھماں سے باہل کریں دیکھیا جائے .

لکھن : اسی روایت در مقام پیادہ سائی ایسیست جہنمی نہار و کوئی تھار نہیں باشنا .

## انواع وراثت

که در این دو شیوه با توجه به برقراره متفاوت است.

- 1- وراثت مفرد (Single inheritance): یک کلاس از یک کلاس پایه ارث بری می‌کند. مثلاً ارث بری سگ (کل جمل)
- 2- وراثت چندگانه (multiple inheritance): یک کلاس از چند کلاس پایه ارث بری می‌کند. مثلاً ارث بری فرزند از والدین
- 3- وراثت چند سطحی (Multilevel): یک کلاس از کلاس دیرگاه ارث بری می‌کند و کلاس پایه نیز از کلاس دیرگاهی دیگر ارث بری می‌کند. مثلاً بارگاه بودن نام خانواده.
- 4- وراثت ساخته راتیب (Hierarchical): چند کلاس مختلف از یک کلاس پایه ارث بری می‌کند. مثلاً چند تابع از کل حاشری بطری همچنان ارث بری می‌کنند.
- 5- وراثت ترکیبی (Hybrid):

انواع کلاس

ارث بری

- 1- کلاس پایه، کلاس خارجی، سوپر کلاس (Superclass, Base class, Parent class, super class): از این نوع ارث بری می‌شود
- 2- کلاس مستقر شده، کلاس خرمند، کلاس (Derived class, child class, & sub class): از این نوع ارث بری می‌شود

نکته: استفاده از وراثت چندگانه <sup>تغییر صفت</sup> نیست مفید، زیرا ممکن است برنامه (چهار اشتباه) شود: # و ممکن از این نوع وراثت هستیابی نمی‌کند

## ترتیب جستجوی متد (Method Resolution Order (MRO))

ابناد در کجا (کام کس) جستجوی متد است. در هایتون اولیه از بایس جیا و اچپ براست است.

تکلیف: همان کلاس ها در یک توکن از یک متد برای نام object است. برای معرفت.

چند ریخت (Polymorphism): واژه ای ترکیب از دو لغت پولی (Poly) و متوف (morph) می باشد.

یک تابع یا متد است که متاتراوی متفاوت دارد. در برنامه نویسی یک زیر کلاس یک متد از کلاس مادر را برای خود شخص ساز معرفت می کند و گرایش حرکت را برای خود (override)

1- Subtype (زیر نوع): شخص ساز متد ها در کلاس زیر کلاس ها عمل overriding را انجام می دهد.

2- Parametric: (متغیر) مقدار ثابت است زنوع داده تغییر می کند. مثلاً جمع کردن ثابت است زنوع آن و داده دارد. روشی که زنوع داده را در معرفت می کند.

3- Ad hoc: برای هر زنوع داده تابع بپالدیه متفاوت دارد. مثلاً جمع متاتراوی متفاوت را در معرفت می کند.

4- Coercion: (تغییر نوع) زنوع داده عوض می شود.

انواع  
چند ریخت

Overriding: یعنی یک شخص ساز متد هار کلاس پایه توسط یک زیر کلاس

Overloading: یعنی رفتار عمدلر یا که با توجه به زنوع داده عوض می شود. مثلاً مجموع اعداد را با هم جمع حساب می کند، ولی دو رشت را به هم پیوند نمی دهد.

Duck Typing: آن فاصله یک آرد را بروی و صدالن پس آردی، حتی آن آرد نباشد. این یک شیوه است که براست.

خواص و تابع یک کلمه نوع آن را شخص می کند، نوع آن یا کلاس که آن را از آن ارت برده کرده است. وقتی یک زیان برنامه نویس از این تابع دهن استفاده می کند، آن تابع بعنوانی و متد هار که را صدای پنجه، تابع یک شنازه نوعی می تواند بگیرد. پایتون برای این اساس مبتدا

بعنوان مثال تابع داخلی Len که چندین نوع داده می گیرد.

توانیم شرکای در پایتون یک شرک است . ۱- همه جیز در پایتون یک شرک است .

در پایتون

۳- یک متفقیر یک ارجاع به شرک است . (ین متفقیر فقط یک اسم است مانند بحسب برخورد شرک )

۱- هر کلاس خود یک شرک از کلاس type است و همچنانه از کلاس object ارت برخورد کند .

نکته

۲- در ساختار جدید پایتون مفهوم کلاس برابر مفهوم type (نوع داده) در نظر رفته شده است .

پیشنهاد ساز کلاس : Class name: *Point*  
متغیر = class's name (ورودی) : ایجاد کرد شرک از کلاس :  
کلasse (ساخته از کلاس)

ex: class Point :

Pass

نکته: برخلاف توابع، دستور pass کلاس را بدل کاربرد نمی کند ،

بله با همین کلاس ساده مقابله برخوان شرک ایجاد کرد .

P = Point()

P.x = 5 or (P.x: int = 5)

P.y = 4 or (P.y: int = 4)

صفت = نام . نام شرک صفت (attribute) یک شرک :

من توان مقادیر صفت را در آن عوض کرد .

تایپ دهن صفت: با بعد افراز کردن (:) و نوع داده های داشت مقدار = نوع : نام . نام شرک

امانه کردن : دو روش (رسم) ( ) نام دهن . نام شرک  
متتبش متتبش  
ب کلاس (مش) روش 2:

امانه کردن رفتار : Class. Point: ( ) نام دهن . نام شرک  
det self: دنگار دنگار

با خودت حرف بزن . یک متفقیر است که شرک را جیبور کند با خودت حرف بزن مین صفات خود را

یتوانند بینند و تغییر دهند . نام self یک قرارداد است به معنای آنکه من از متفقیرهاش آنکه نیز استفاده کرد . (self) توصیه شده برخشد

کاربرد: همان شرک است یعنی با این طریفه امانه کردن متتبش (شن به همان self ( )) قرار گیرد . وجودش اجباری است .

ست (property): از بازول ریاض صدرازده می شود و مالکیت بین دو طرف را داشت و ترک یک مثلا فهم را بسیار کند .

from math import hypot

یک شود که از صفات قبل:

class Point:

def remove(self, x: float, y: float) → None:

self.x = x

self.y = y

پایاد سازی متد در کلاس

def reset(self):

self.move(0, 0)

def dis(self, other: 'Point') → float:

return hypot(self.x - other.x, self.y - other.y)

\*

P1 = Point()

P2 = Point()

P1.move(1, 2)

P2.move(3, 4)

Print(P1.dis(P2))

ایجاد یک شی از کلاس

پایاد سازی متد در شی

(\*) بعده که اشتاد در فاصله آوار و وجود دارد و باید

خارج از بین کلاس باشند

مقادیر اولیه شی: در بین زبان های مانند C++ یک متادرون کلاس تعریف می شود که دظیفی مقادیر اولیه را دارد، و از

در پایتون دو متد --new-- --init-- باشد که کلیدی آنها را (انجام می دهند).

متاد --new--: یک شی جدید معرفی می شود، یعنی دلیل ساخت شدن یک شی جدید است.

متاد --init--: عذر بالا بس از مختص شی و قابل از بارگذاری (return) نیست، این متاد را صراحتاً می توان تابع دعوه

(این متاد خروجی شود) مثلاً صفات بعد

class name:

def \_\_init\_\_(self):

نکته: هنگام ایجاد یک شی آنکه وجاها همان متد init می روند

متد پیش مرش خارج شری  
 متد پیش مرش خارج شری  
 برای این کلاس متد `move` دارد که مختصات `x` و `y` را دریافت می کند و آنرا با مقادیر `nx` و `ny` بروزرسانی می کند.  
`def class Point:`  
`def __init__(self, x: float = 0, y: float = 0) → None:`  
 `self.x = x`  
 `self.y = y`  
`}`  
`P1 = Point(1, 2)`

**دستور `dir`:** بین الممکن ترین نوشته های کاربرد توابع کلاس ها و فهره را توضیح می دهد.  
`dir:` **توضیح**  
`1` متد `>>>`  
`2` متد `__init__`  
`3` نام متد ها  
`4` توضیح در مورد آنها  
`5` فرود

**لذت:** بستر است ابتدا در پدر کلاس توضیحات در پدر کلاس داده یک مثال بزنید.  
 سپس در بخش هر کد از متد ها داک جواب بخوبیم.

**تابع `help`:** هم بعنوان دو دو دنام کلاس را می برد و توضیحات و داک اسٹریکت های آن کلاس و متد هایش را به صورتی منتظر برگرداند.

**متدهای `testmod`:** از بزرگترین فایل های اسال و نوع داده های ما اسال را چک کنند که غلط نباشد در cmd باید استفاده شود و نتایج بتصب را در.  
`import doctest`  
`doctes.testmod()`

**از بعض کلاس ها فقط یک نشی ساخته می شود که می توان این شن را `import` کرد. مثلاً کلاس های `str` و `list` هم از این طریق استفاده می شوند.**  
**توجه!** بعض `import` شن ماخته می شود. برای جلوگیری از ماخته شدن در اینجا `from` می توان از استور زیر استفاده کرد:

`en: Point = None`  
`def get_Point:`  
 `if not Point:`  
 `Point = Point()`  
 `return Point`

**ساخته خلاصه از ساخت ستر ایک شن از کلاس:**  
**شن را در محیطی می بینیم که کلاس وجود دارد.**  
**سپس آنرا در محیطی دیگر `import` می کنیم.**

`{from Point import get_Point`  
`P1 = get_Point()`

[Optional] : بین متناسب است که یک داده منقول و نوع None باشد (اردولاس باشد) . None تایپ باشد یا تایپ نباشد

from typing import Optional

optional[نحوه داده] = None : امتنع

درین [ ] شخص رشود .

حافظت از داده ها در مقابل دسترس غیر مجاز

public :

pub = ...

private :

pri = ...

protected :

pro = ...

Public : چه داخل چیزیون کلاس در دسترس آن در زبان های دیگر: نمایه سطح دسترس

Private : فقط داخل کلاس در دسترس آن در زبان های دیگر: نمایه دسترس

Protected : داخل کلاس نه درون آن ترا را در و کلاس

هر چند از آن ارث بری می کند، قابل دسترس است.

توجه: پایتون چندی تا بهینه ندارد

در پایتون: تمام داده ها در پایتون قابل دسترس در همه جا هستند .

ex: for \_ in range(1,):

print('hi')

1- در حلقه: نام متغیر دفعه را (-) مگرایم نیز لاریوس ندارد . (-)

2- در تابع: فقط یکی از عنصرهای تابع یه درجه هفتم می خورد . (12 و 8)

3- در ترتیبی: آخرین داده مورد استفاده مادربری شده .

Under Score 5 کاربر

2- استفاده از \_ قبل از نام متغیر: متغیر را تبدیل به یک داده Protected می کند . توجه: آگرای متغیر از یک ماتریس دیگر با روشن ... شود، اجزای دسترسی آن را ب صحیح و جنخواهیم داشت . نکت: باید جو امری که داده مخاطب شده است در پایتون قابل دسترسی باشد .

3- استفاده از \_ بعد از نام یک متغیر یا متد: وقت بخواهیم از یک کلاس کلیدی بعنوان نام متغیر استفاده کنیم، به صورت ترا را در داده under مگذاریم .

4- استفاده از (--) قبل از نام یک متغیر یا متد: متغیر را به یک داده Private تبدیل می کند . (در واقع متدی است که اتفاق نمی رند) . \*: نام کلاس را به اول نام متغیر کرد (--) قبل از آن وجود دارد اما در کارهای کارهم دسترسی به داده سخت تری شد . دم (زیرا فعل نام جمله هر کده می شود) .

5- استفاده از (--) قبل و بعد از نام یک متغیر یا متد: متغیر یا متد را تبدیل به متد خاص یا متد جلاویم می کند .

لیست کلاس

class ... :

```
def set_... (self, ...):
    if ... :
        self. .... = ... >
```

```
def get_... (self):
    return ...
```

تغییرات را اعمال مکن و آنرا ذخیره کن   
 دوباره برداش.

کامپون کردی که صفت در کلاس: با استفاده از این متد اینجا نمود

ex: class User:

def \_\_init\_\_(self, phone):

self.phone = phone

def set\_phone(self, phone):

if len(phone) == 11:

self.phone = phone

def get\_phone(self):

return self.phone

تابع متغیر str: یک راه را به شکل زیر برای کاربر می‌آورد. مثلاً یک رکورد را بعنوان کوئی نشان می‌دهد.

متغیر repr: یک راه را برای برنامه نویسی نمایش می‌دهد.

نکته: در کلاس آندر str تغییر نمود، بین فرض repr در نظر قرار نمی‌شود. حتی اگر همان کوئی شن درون آن فوارمینگ بازم بصریت repr بگذارد.

```
{ def __str__(self):
    return ...}
```

```
{ def __repr__(self):
    return ...}
```

ex: class Person:

```
def __init__(self, name, age):
    self.name = name
    self.age = age
```

```
def __str__(self):
    return f'{self.name} {self.age}'
```

```
def __repr__(self):
    return f'{self.__class__.__name__}({self.name}, {self.age})'
```

شخص ساز str در کلاس:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def __str__(self):
        return f'{self.name}'
```

from \_\_future\_\_ import annotation  
class BankAccount:  
 : ~~select own notes~~  
 all\_account\_numbers & list[int] = []  
 last\_account\_number = 999  
 def \_\_init\_\_(self, name: str) -> None:  
 BankAccount.last\_account\_number += 1  
 an = BankAccount.last\_account\_number  
 self.account\_number: int = an  
 BankAccount.all\_account\_numbers.append(an)  
 self.name = name  
 self.balance: float = 0  
 def display(self) -> None:  
 print(f'Hi, {self.name}! Your current balance: {self.balance}')  
  
 def deposit(self) -> None:  
 amount = float(input('Please enter amount to deposit: '))  
 self.balance += amount  
 self.display()  
  
 def withdraw(self) -> None:  
 amount = float(input('Please enter amount to withdraw: '))  
 if amount > self.balance:  
 print('Insufficient balance!')  
 else:  
 self.balance -= amount  
 self.display()

**برای کلاس** **انواع صفت**

<code>en: self.name = name</code>	<code>instance attribute - 1</code>
هر چند و شن برای خود مقدار متفاوت دارد. در بین متد ها تعریف نشود.	
<code>en: class attribute - 2</code>	
یک صفت مشترک برای همه شن ها می باشد و در بین کلاس دخایل از جنس متد ها تعریف نشود.	

متد PPrint : از مارکیل این پرینٹ برای خود بصورت جالبتری درود را چاپ کنید.

**ارتیبری:** از تکاری چیز که جلوی پری می‌کند، هدف ایجاد یک لاس یا پر است که باقی لاس‌ها را آزاد ارتیبری کنند.

برای آنکه یک زیرکلاس از یک کلاس پایه ارث بری کند کافیست اسکرپت کلاس را در فون ہر اندر رونمایی کرده و روی زیرکلاس پھینگ بینیسیم:

**نکته:** تمام کلاس ها در پایتون از `object` برگزینیده اند.

**اصطلاح:** زیرکلاس، کلاس پایه را توسعه می‌دهد. **نتیجه:** تغیر کلاس تعاون متدها و صفت‌های کلاس پایه طا دارد و در توان برای فریکلاس شهادت نیز تأمین شد.

**نکتہ:** ارکیوٹس میں ہاس داخل ہائیکوون (مانند لیسیے) نیز من توان ارتھ برمی کرد۔

**Over riding**: تغییر رفتار (تغییر اراده یا کدام) در یک زیرساز که از کلاس پایه ارث برده است. یعنی، کدیک متد ارث برده شده را در زیرساز دوباره (با کد متفاوت) تعریف کنیم. توجه کنید که رفتار متد فقط در زیرساز مجدد نظر تغییر کند.

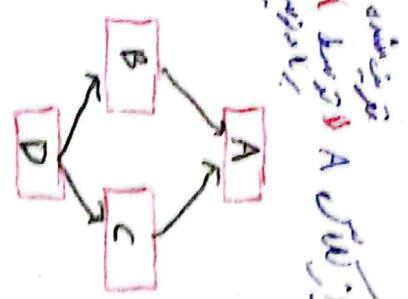
(Proxy object) Super(): باعث دسترسی موقت مایوس پایه شود، تا از تکرار که درست نباشد. جلوگیری

شود، وقتی رخداد overriding می‌شود، یک متده استفاده کنیم، از SUPER در زیرنویس استفاده می‌کنیم.

براساس الگوریتم K-Means کار کند.

برای جلوگیری از آنج چشم ارشاد رک مته کالا س یا چرا درون هاک زیر لباس باشد مدارس زنی.

def متا:



اڑتے ہوئے کیلئے ہم نے ہمیں فوجی اسٹریکٹس کا ارتھ بڑی اس سے مونین۔

کلاس های را مشخص نمایند.

متدهای اولیه در وراثت چهندان: با استفاده از `super`, `**kwargs` و `SUPER` این کار را بگام انجام دهیم.

ex: `super().__init__(self)`  
برای شده

دسترسی پنهان مورثیسم: با استفاده از `__init__` تابع به صورت مقابل انجام گیرد:

ترجمه: چندریختی در وراثت که جدید ندارد.

فقط `Minin`: کلاس شامل نموده است که این متدها در صورت نیاز باقی زیر کلاس ها ارت بری می شوند. هر چند که رابطه بین `minin` و زیر کلاس ها نیست. نکته: از این کلاس ها، شئ ایجاد نکنیم. آنرا این کلاس `mixin` استفاده نمود (تصویر شده). این کلاس ها معملاً از اولیه نداشته اند.

عملت استفاده: `المجموعی از تکرار کد - 2`. وریزک های اختیاری که برا برخی کلاس ها استفاده می شود. (بایک ارت بری مماده قابل دسترسی است)

گریف نوع داده (رتور اند) خاص داشت باشد، توسط یک تابع به خصوص قابل دریافت می باشد.

(Look Before You Leap) LBYL: قبل از صراز در یک تن، بروی کن که آن دهن خاص را دارد یا خیره.

ex: `def check_len(obj):`

```
if '__len__' in dir(obj): → if hasattr(obj, '__len__'):
    print(len(obj))
else:
    print('Sorry...')
```

(it's easier to ASK for forgiveness than Permission ) EAFP

For

```
ex: def check_len(obj):  
    try:  
        print(len(obj))  
    except:  
        print("Sorry...")
```

یک شئ را صادر کر متوجه اشت کردی ایشت عذرخواهی کن.

نه: این حالت از حالت قبل بینه تر است.

نکل سایر محتوای درس اشغال هندس (جلسه 29 فعل ۱)

class Shape:

```
def __init__(self, **kwargs):  
    self.area = None  
    self.perimeter = None  
    for key, value in kwargs.items():  
        setattr(self, key, value)
```

def area:

pass

def perimeter:

pass

def show(self)

info = ""

for key, value in self.\_\_dict\_\_.items():

if value > 0:

info += f'{key}: {value}\n'

print(info)

اُسپاچه با قابلیت فراخوان (callable objects) : کلاس های توپ از این اُسپاچه هستند. (هتلام ساخت یک تن آش، آنرا فراخوان رکنیم)

تابع (callable) : شخصیت رکنیت یک شئ قابلیت فراخوان دارد یا خیر.

تجویی که نظر ساخت شده از یک کلاس در حالت عادی قابلیت فراخوان ندارد. برای آیا که هدایت فراخوان را بین انتها

def \_\_call\_\_(self):  
 (تفصیل و)  
 مقدار = تفاصیل  
 self

کافی است متد مقابل را درون ہدن کلاس مورد نظر بنویسیم:

ex: class A:

```
def __init__(self, n):  
    print("init")  
    self.n = n  
  
def __call__(self, z):  
    print("call")  
    self.z = 5
```

برده: در طول کد بخواهیم مقادیر دهن اولیه را  
دوباره انجام بخیم. برای اینکار کافی است  
شئ را دوباره فراخوانی کنیم.

با این رویش، یک کلاس دکوراتور نیز می توان  
ساخت.

## انواع متد کلاس

instance - attribute دسترس دارند و روی آنها تغییرات اعمال می کنند.  
همان متد هایی اند که تابعی جای کار از آنها استفاده کرده ایم و (self) را بعنوان درودی می گیرند.

class - attribute به instance - method - 1 : متد هایی که به class - attribute می ساختند.

برای ساخت آن از دستور @classmethod استفاده می کنیم و زیرا این دستور هر متدی بنویسیم از این نوع می باشد. ورودی این  
متدها (cls) می باشد. (cls حذف کردن می باشد)

Static method - 2 : می تواند ورودی تغییر دهد چنین من توانم آنرا خارج از بین کلاس نویشت. درون ہدن تابع بالکل دسترسی  
کلیبی دارد @staticmethod می شود.

Abstract method - 3 : متد هایی که در کلاس پایه (کلاس انتزاعی) پایه سازی نمی شوند بلکه فقط نام اکتساب و پدر کار است. برای آنها در کلاس پایه  
وجود دارد و جزویت آنها در زیر کلاس ها تعریف می شود. از کلاس لیست @abstractmethod استفاده می کنند

ex : def set\_r(self, r):  
self.r = r

getter, setter  
getter, Setter

def get\_r(self):  
return self.r

Ex: class Color:

def \_\_init\_\_(self, name):  
self.name = name  
self.rgb = rgb

def \_set\_name(self, name):  
self.\_name = name

def \_get\_name(self):  
return self.\_name

@ Property  
def name(self):  
return self.\_name } -getter

@ name.setter  
def name(self, name):  
self.\_name = name } -setter

name = Property(\_get\_name, \_set\_name) مقدمة بـ Python بـ نوشته خطوة خطوة



چیزی که می‌نویسی؟ (Composition) ساختاری

ex: class question:

```
def __init__(self, q: str, a: list):
    self.q = q
    self.a = a
```

class exampaper:

```
def __init__(self):
    self.question = question("What's your name?", ['Saman', 'Ali', 'Reza'])
```

ex: class student:

```
def __init__(self, name, number):
    self.name = name
    self.number = number
```

aggregation  
که متال بادست را

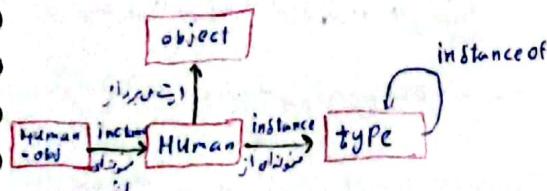
تفاوت باشد: در متال رو بروش بردن کس ایجاد شود، برگش دوم بروند و لی برشل باید مش (از کلاس اول در کلاس دوم ساختیم).

class university:

```
def __init__(self, *students: List[student]):
    self.students = students
```

st = [Student('Sam', '12'), Student('Ali', '13')]

un = University(st)



متا کلاس (Meta Class)

بکلاس های که انسیاد ساخته شده از آنها یک کلاس هستند، کلاس کویند.  
در شکل رو برو، type، یک متا کلاس من باشد.

لذت: تمار کلاس های داخل هایتیون (باند ۱۰) یک نوی از کلاس type هستند.

class meta(type):  
 pass

class dog(metaclass=meta):  
 pass

کلاس انتزاعی (Abstract Class)

یک کلاس که استنکرده داشت جزویت نویسنده نیست و اینه بزیست. تو سه کلاس هایی داشت، (انگر، انگلی، ارٹ) هر یکی که پاده می شوند.

در این کلاس نام نهاد و که درسته آن نوشته شده و در زیر کلاس های مادرها کاملاً پاده می شوند، لذت: از این کلاس های نوشی نمی بازد. مقدار دهن اولیه ندارند.

```

from abc import ABC, abstractmethod
class Vehicle(ABC):
    * {
        @abstractmethod
        def move(self):
            """ this method should be implemented """
            print('moving')

```

این کلاس که از Vehicle ارث می کند تعریف شود. این متد یک مقادیر بین فرض نیز توان تغیر کرد: اما باز هم مسأله باشد از زیر کلاس در هر دو متد move و move() استفاده ننماییم.

```

from abc import ABC, abstractmethod
class m(ABC):
    @abstractmethod
    def move(self):

```

### برای نمونه مثمرها (Operator Overloading)

یک عمل (operator) بسته به آینده با چنین داده ای کار می کند رفتار مستفاده دارد. مانند عمل جمع (+). برای ایند مشخص کنیم که عمل این دو کلاس فوکوس شده توسط خودها، پھلویه رفتار کند باشد آن عمل را در دو کلاس دوباره نمیں کنیم. در واقع های متاداوم را باید این کلاس

class name:

def \_\_add\_\_(self, other):

→ e.g. class Point :

def \_\_add\_\_(self, other):

self.x + other.x

Slots

## Slots

از اضافه کردن ارزیوبت جدید به شئ جلوگیری کرده و همچنین ممکن است در آنکه دلخواه ساخته شود

class name:

--slots-- = ('**شایعه**)

**en**

class myclass:

--slots-- = ('**a', 'b')**

def --init\_\_(self, a, b):

self.a = a

self.b = b

اگر **obj** = مقدار باشد، آنرا در **obj** = مقدار بخواهید

نکت: اگر کلاس پر **slots** داشته باشد، زیرا ارزیوبت های مشخص شده در **slots** را برای این داشتند و وجود **slots** از خیر کلاس **dict** نداشتند باشد، همان‌آن ارزیوبت اضافه کرد. حین اگر کلاس پر **slots** داشتند باشد

با دادن سازه iterable و iterator در کاملاً خواهد

یک شیوه برای آنکه iterable باشد باید هم **iter** و **next** را اضافه کرد. برای آنکه مشخص کرد که این دسته ساز قابل تهاجم باشد، ممکن است **iter** را با **\_\_iter\_\_** نام دهد. سازنده کنیم:

class name:

def \_\_iter\_\_(self):

for i in self:

yield i

برای تبلیغ این ایجاد کار است که شئ ایجاد شده را درون ممکن است

قرار دهیم. این دسته قادر است **next** را ایجاد نماید.

```
eu: class PowTwo:  
    def __init__(self, max_pow):  
        self.n = 0  
        self.max_pow = max_pow  
  
    def __iter__(self):  
        return self  
  
    def __next__(self):  
        if self.n <= self.max_pow:  
            result = self.n * 2  
            self.n += 1  
            return result  
        else:  
            raise StopIteration
```

دکوراتور در کس - ترا باتابع یک دکوراتور ساخت باین تعداد که ورودی آن کلس است یا ایند دکوراتور را در کلس  
تقریب کرد.

روش اول (تک)

```
from functools import wraps
```

```
def decorator(c1s):
```

```
    @wraps(c1s)
```

```
    def wrapper(*args, **kwargs):
```

```
        print("-" * 40)
```

```
        c1s(*args, **kwargs)
```

```
        print("-" * 40)
```

```
    return wrapper
```

```
@decorator
```

```
class test:
```

```
    pass
```

روش دوم (تک)

```
class Decoratorc1s:
```

```
    def __init__(self, func):
```

```
* Update-wrapper(self, func)
```

```
    self.func = func
```

```
    def __call__(self):
```

```
        print("-" * 40)
```

```
        self.func()
```

```
        print("-" * 40)
```

```
@Decoratorc1s
```

```
def my_func():
```

```
    """ Sam """
```

```
pass
```

نکته: هر خط دکوراتور است یعنی name داشته و شود.

برای ارتیبیت ها ایجاد می شود. مثلاً `Property` با این قواعد که برای خذار میرت باید فرمانها بکار نوشته شود.

ex: class name field:

```
def __init__(self, owner, name):
    self.name = name

def __get__(self, instance, owner):
    return instance.__dict__[self.name]

def __set__(self, instance, value):
    if 0 < len(value) < 20:
        instance.__dict__[self.name] = value
    else:
        raise ValueError('invalidname')

def __delete__(self, instance):
    print('deleting....')
    del instance.__dict__[self.name]
```

: دستورات، متدهای قابل ویرایش (دستورخانه)، کار را (نحوی درجه) (مانند کوئرانر)

ex: class C:

```
def __enter__(self):
    print('start')
    return self

def __exit__(self, exc_type, exc_val, exc_tb):
    print('end')
```

: دستوراتی مسند و قابل وجد یک دستور خاص برای انجام این دهنده (مانند کوئرانر)

Content manager

a: class C:

def \_\_enter\_\_(self):  
 print('start')  
 return self

def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):  
 print('end')

```
class filemanager:
    def __init__(self,filename,mode):
        self.filename = filename
        self.mode = mode
        self.file = None

    def __enter__(self):
        print('the file was opened!')
        self.file = open(self.filename, self.mode)
        return self.file

    def __exit__(self,euc-type,euc-val,euc-tb):
        print('the file was closed!')
        self.file.close()

with filemanager('...'):
```

With file manager (- - )

**data class**: کلاسی که از سیستم های در خود ذخیره کرده و خود کار می کند. init را پیدا ده سازی می کند.

```
use : from dataclass import dataclass, initvar
```

## @dataclass

## Class Person:

Name : Sri

family : sir

age : int

gender : initvar[ str ]

**ملکت:** حتماً باید نوع داده‌ی ارزیوسترا

مکتبہ

مکانیزم این ایجاد می‌شود (معنی)  $\rightarrow$  initVar

کاربرد داد و جزو اخراجیت های کلی

مکتبہ نشریہ

\* def \_\_Post\_init\_\_(self, gender):

if gender == 'man':

```
self.name += 'x'
```

$P = \text{Person}(\text{'saman'}, \text{'Fardin'}, 21, \text{'man'})$

prints(p)

## خطاها و استثناءها (فصل ۱۱)

کمین رفتارهایی هشخض برای خطاهای بگویان که برنامه مختل نشود.

- ۱- خطاهای ساخته شده ( Syntax Error ) : به این نوع خطای نیز گویند. این خطاهای را در داده قرار دادن فواید زبان رعایت نشود و قبل از اجرا برنامه دفسر متوجه آن می شود.
- ۲- استثناء ( runtime Error or exception ) : هنگام اجرای برنامه هشخض می شود.

نکته: مکرری خطا نیز وجود دارد که اجرای برنامه مختل نمی شود بلکه محاسبات آن استثناء در باشد و بر آن خطای منطبق می گویند.

## مدرسی استثناء (Exception Handling)

برای حلول بر از مختل شدن برنامه هنگام روز یک استثناء استفاده می شود.

```

intau:
    try:
        دستوری که
        احتقال خطا دارد
    except خطأ:
        آر با خطا وابسته
        چکنیم.
    except:
        دستور متابد با خطای
        نامشخص
→
eu: u = int(input())
y = int(input())
try :
    Print(u/y)
except:
    Print('errone')

```

نکته: آر در try + بخطاب بخوبیم، ادامه دستورات بین اجرانش شود و بعد except برود. در غیر این صورت دستورات بین except با استفاده از تساوی

توضیح: بین try و except عوشه نام بعد آن ندارند.

```

eu: u = int(input('number1:'))
y = int(input('number2:'))
while True:
    Try:
        Print(u/y)
        break
    except ZeroDivisionError:
        Print('try again!')
    except:
        Print('Unknown Error')

```

نکته: در پایتون همه جیزشی است به عنوان خطا نیز شناخته می شوند

و در نهاد آنها (از اس خلاصه) مربوط صادرند:

بعضی مثال!

```

except ValueError as ve:
    Print(ve.__class__.__name__)

```

توضیح: اینجا چند خطای دیگر قرار داده ایم

که سیم من می شود.

```

scint(input())
eui: g.int(input())
try:
    Print(wg)
except:
    print('Error')
else:
    Print('OK')
finally:
    print("Finally")

```

ساختار else

در صورت اجرا شدن except بذریعه else اجرا نموده

دستور finally

بدنبی این دستور در هر صورت چه خطا رخ بده جذب نموده اجرا خواهد شد.

try:

syntax: ---

finally: ---

مدیریت استثناء تردید ترمه

نافرمان ممکن است دستورات بدنبی except نیز نیاز به مدیریت استثناء داشته باشند. در این صورت ساختار مدیریت استثناء را در بعد از except نیز پیاده سازی می کنیم. البته این نوع مدیریت توصیه نمی شود چرا که هم اینجا را بیچیده می کند.

نکت: بدنبی ساختار مدیریت استثناء مانند شرط (if, elif) می باشد.

ساختار یک استثناء شخصی با استفاده از کلاس

برای اینکه این کلاس را نویسیم که از کلاس های استثناء ارث برداری کند.

```

eui: class digitError( Exception ):
    def __init__( self, s, message = "Error" ):
        self. s      = s
        self. message = message
        super(). __init__( self, message )

```

syntax: import warnings  
warnings.warn("پیغام")

(warnings) نظرها

برنامه را متوقف نمی کند، فقط توصیه می کند که دستور است که استثناء نشود.

مانند استثناء، این توان اخطار اتفاق افتادن ساخت (روز یک دست)

```

eui: def func(x, int, y, int):
    if not isinstance(x,int) and not isinstance(y,int):
        warning.warn("x and y Can't be strings!")
    Print(x+y)

```

assert دستور

اگر شرط جلوی این دستور ننوشته باشد، اگر آن را درست باشد برنامه اجرا می شود، در غیر این صورت Assertion Error می دهد.

Syntax: assert "نمایم" و شرط

این پیغام و ویرایی ارور ننوشته می شود.