

محیط مجازی (virtual env): یک محیط مجازی است که در برگیرنده پروژه پایتون بر اساس

ورزش آن و کتابخانه های مختلف است. نه به برنامه نویس اجازه می دهد آن پروژه را

در سیستم خارج از آن ورژن را اداره نمایند و با کتابخانه را ندارند اجرا کنند.

پوشه lib: کتابخانه های نصب شده در محیط مجازی در این پوشه قرار دارد

پوشه script: ورژن و مفسر پایتون در این پوشه اند

نفس کتابخانه (Package): دللم Python در قسمت راست پایین صفحه

نحوه پیکربندی Interpreter settings

لیست های میانبر: بر اساس اینها

چالیگزین کردن کلیک $ctrl + R$

سرچ لامه مورد نظر $ctrl + F$

جابه جابی خط به بالا یا پایین $ctrl + shift + Arrow up/down$

ایجاد جدید نشانه موس برای تایپ حمزه ای $Alt + click$

Rightclick → Refactor تغییر نام در هرجایی که باشد

$ctrl + Alt + L$ مرتب سازی متن

$ctrl + W$ انتخاب متن به ترتیب از نشانه

$ctrl + D$ پی آن متن به خط بعدی

ctrl + /

گافنت کردن متن انتخابی

ctrl + space

السینت، راهنما، پیشوازی نمودن

Alt + Enter

اصلاح خطای احتمالی

ctrl + Alt + O

پاک کردن لغایت اسکاره نموده

ctrl + Q

ستاندارد دال یومنت دستور حذف نمودن

ctrl + .

لولیس کردن چند خط

ctrl + `

بازگرد یعنی ساختن سازی

ctrl + shift + A

بلند کردن پیغمه Actions

ctrl + B

ستاندارد دال استریل دستور

shift + Enter

صفت به سطر بعدی بدون فرمودن کردن

پایان فصل اول

shift + A

انتخاب تمام کلمات برابر:

shift + Q

صفت به سطر برابر نمودن:

shift + E

انتخاب کلمه فعلی:

shift + S

ba-hora - شروع درس class : با هر آن -

MyClass . Pasca Case (Class) : بصورت Pascal case (Class)

myFunc . function (function) : حروف لوچ، جدا سازی کلت با آندر (سلو)

myFunc . camelcase : با بصورت camelcase نمای اوپن حرف اول لوچ و بعدی بزرگ

شیءگرایی (object oriented) : در مفهوم برنامه نویسی شیءگرایی یک چیز

کلی به نام class وجود دارد که نام داره احوال مادر آن هست class های

نفسی لیکن یک شیء هستند که انسان بر اساس اون ساخته هستند مثل نفس ساخت

ma-sin (attribute) . class ها از سه بخش شیء ویژگی (object) -

method (Method) تسلیع هست.

object : اینها همچنان نونهای هستند که بر اساس یک نفس ساخته هستند

دارای ویژگی ها و ریز هستند. مثل ما-sin که در کارخانه بر اساس نفس ساخته شده

Attribute : اینها ویژگی های شیء مورد نظر هستند. مثل زنگ ما-sin ، تعداد آئینه

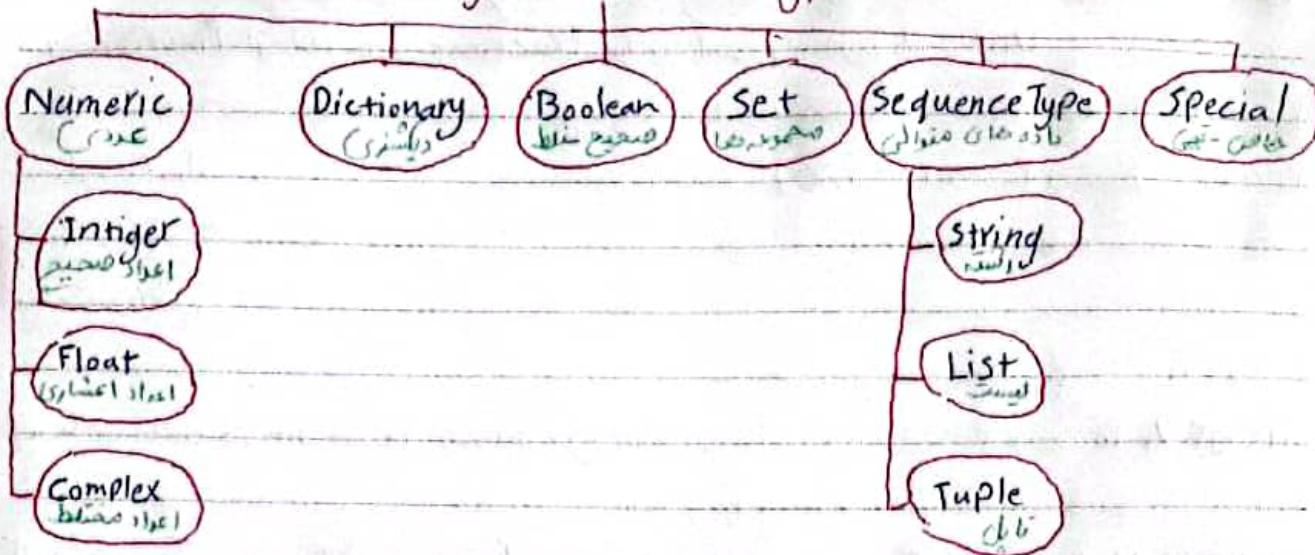
Method : اینها کارهای است که یک شیء بر اساس ویژگی های لاس خود می تواند

النجام بدم. مثل فرمان ما-sin ، یا مویر ما-sin .

* بصورت کلی الگویی بزم نفس ساخت : لاس ما-sin : شیء خصوصیات

ما-sin : ویژگی این راه کارهای یک ما-sin است *

Python Data Types



اعداد (Numbers): (Arabic) عددي در برابر نويسی به صورت (Numbers)

اسفاره می شوند. روشن ذپیره کردن آنها به سه زیر است.

Hexadecimal: num = 0x18 $\xrightarrow{\text{Join}}$ 0xa21c45f

Decimal: 10 اعداد

octal: 017 اعداد $\xrightarrow{\text{Join}}$ Num = 001732

Binary: 0b10100 $\xrightarrow{\text{Join}}$ Num = 0b10100

نهادها و توابع کاربردی اعداد:

bin() $\xrightarrow{\text{Join}}$ bin(12) => 0b1100 تبدیل عدد به باسیزی

hex() $\xrightarrow{\text{Join}}$ hex(12) => 0xc تبدیل عدد به هکسادسیمال

oct() $\xrightarrow{\text{Join}}$ oct(12) => 0o14 تبدیل عدد به اولی

import fractions
Fraction() → Fraction(0.75) → 3/4

از کتابخانه fractions برای کسری نشان دادن اعداد Fraction است

round() → round(5.8) → 6 گرد کردن اعداد

نکات ریاضی:

۱. باقی مانده هر عدد به ۱۰ رقم آفراز است.

۲. هر عدد تقسیم بر ۱۰ (یعنی ۰.۱) یک رقم ازین کم می‌شود.

رسانه‌ها (String): داده‌های رسانه‌ای را می‌توان از لغاظ‌های مختلف می‌باشد که

به عنوان هستن‌چاری کاربرد دارد. برای ذخیره داده باید کوئی (یا) قرار بگیرد.

برای استفاده از دستورات خاص در رسانه از یک اسلش (\) استفاده می‌کنیم؛ همچنین

برای لفظ آن هم استفاده می‌کنند (فیلر):

x = "Ali\nReza"
که اسلش (\) را در پیش از سطر دستور استفاده می‌کند (الفونت آن دو نویسه ای از حروف ایجاد شده)

نکته: اگر در متن از دستورات خاص لجذب‌هایم استفاده نکنیم، برای اینکه پاسخ آن را رسانه عادی

در نظر بگیرد، قبل از رسانه از حرف ۲ استفاده می‌کنیم.

در اینجا همان رسانه نمایند چهار فیلر وجود دارد، سطر اول از اینها بود

رسانه‌ها همان‌که در آن دارای تعداد (Index) هستند محل آن علاوه بر رسانه را مشخص

می‌کنند. که بعدها می‌توان برای این رسانه ای برای آن انجام (همیم برای) مثال رسانه زیر داشت

H	A	M	I	D	character
0	1	2	3	4	Index
S	A	B	A	S	

escape دستورات گفتگو = اصطلاحاً (S) = "Ali\007" : میل = (I)

برای دسترسی به آن ها باید متغیر را نوشت سپس اینکس مورد نظر را درون برآورده (برآورده) قرار دهیم. برای فضای دستور زیر حرف ز در رشته را طبیعتی لند $x = "Ninja"$

`Print(x[3])`

توضیح: مادی توانی برای دسترسی به قسمت های خاص رشته از عبارت کلمه بلیز `slicing`

روش استفاده آن: داخل برآورده از دو نقطه (:) بین اعداد استفاده می شود، نباید ترتیب

از سمت چپ به راست [1 : 3] به این صورت از اینکس 2 تا اینکس 5 از اینکس 5 تا اینکس 1 باز است. $x = "AliReza"$ نتیجه کام محدود کام از عبارت `Print(x[2:5])` بخواهیم جایز نباشد به این صورت است.

نحوه این کام می توان برایم یک در میان ۲۰ چند در میان از این روشن استفاده نمود. نتیجه این

۱- قرار دهیم رشته را در علیس جایی می کند.

در زبانی کامپیوتر هر کاراکتر را ای نمی نماید به آن Ascii لفظ می شود.

فقط متعدد کاراکترها ای ای و اعدادی باشند. تابع `ord()` کاراکتر را نشان می دهد

`ord(A) ==> 65`

`chr(65) ==> A`

نکته: این کد بسیار کسر و ترازو Ascii می باشد و من توانم به وسیله آن، به کاراکترها که در کیبورد وجود ندارند (دسترسی داشتم) پیدا نمایم.

با استفاده از \u میتوانم کاراکترها که در کیبورد وجود ندارند (دسترسی داشتم) پیدا نمایم.

`char = "char:\u0489" ==> char:`

Subject :

Year . ۰۲ Month . ۱۴ Date . ۱۱

عکس، ۰ هم در طول صحابه فنی خالی ۰ قرار می‌رهد در واقع flag بروی

۳ اعمال می‌گردید

داخل پرانتز می‌نویسیم (key)

: بروی کلیه مورد نظر لازم دلیل‌تری اعمال می‌گردید مثال:

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

Print("%(b)i and %(a)i" % (dic)) \Rightarrow 2 and 1

لذت: الگوی بخواهیم بجاورد کردن عملکرده بصورت دستی آن هارا از تغییر طیفی

باید به جای عدد از ساره استفاده کنیم برای مثال الگوی بخواهیم کاربر بصورت

دلخواه رقم اعشار را مشخص کند بصورت زیر انجام می‌گیرد مثلا:

$$x = 3.12345$$

ashar = int(input("ragam")) فرض کنیم براز اوارد شود

Print("%.*f" % (ashar, x)) \Rightarrow 3.123

سینه متد format (فرمت): در این روش برای مقدار دهندر رشته {} آنرا قرار می‌دهیم. مثلا:

print("x is {}".format(10)) \Rightarrow "x is 10"

همچنین برای دادن چند مقادیر ترتیب را روابط می‌کنیم و با داخل آنرا در حافظه بروش

دهی محل آن را مشخص کنیم مثلا: Index

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

SABA

* = ساختارها بوسیله دالیو صفت پایه‌ی مخصوص آن نویسند. در ساختار باشد بوسیله توپونیس

عایت شود. و هر چیزی که در داخل [] قرار گیرد اختیاری است.

Subject :

Year .

Month .

Date .

فرمت دهنده رشته‌ها: ما در مواردی نیاز داریم که از نوع داده‌های مختلف در رشته خود استفاده لینغ و اون هارو در قالب رشته منطبق دهیم. این کار به ۱۳. روش است:

۱. عملکرد % (دھن): برای فرمات دهنی بصورت لی ابتدا % را نویسند سپس نوع متغیر را با حرف اول آن مستحبه کنیم (منلا برای int من نویسیم %) سپس به از آن علامت % را خارج از رشته نویسند و داخل پرانتز متغیر مورد نظر قرار می‌گیرد. مثل:

$x = 10$

`print("x is %.i" % (x))` $\Rightarrow x \text{ is } 10$

این عملکرد را برای ساختار لی به شکل زیر داشت. (استفاده از بقیه آنها بصورت type اختیاری است)

۲. [Key] [Flag] [w] [P] type: این ساختار مانند عملکرد نویسندگ نویسند. لیکن اینجا نوع داده مستحبه نمود که اجبری می‌باشد. مثل: %i که نتایج int است

۳. اینجا نوع داده مستحبه نمود که اجبری می‌باشد. مثل: %f که نتایج float است

۴. تعداد رقم اعشار را مستحبه می‌کند. برای مثال برای نتایج دارن دو رقم اعشار از اینجا

عدد ۱۴۵۶.۳ باید %f را نویسند سپس \uparrow . را نویسند بعد از داده را مستحبه می‌کنیم مثل

$x = ۱۴۵۶$

`print("%.2f" % (x))` $\Rightarrow ۱۴۵۶$
نوع داده

۵. برای افزاش طول رشته استفاده می‌شود. آنها را با فضای خالی پر می‌نمایند

`print("Pi is %0.۳f" % pi)` $\Rightarrow (\text{x}).%0.۳f$ نتیجه

اعمال: سه حالت دارد: ن قبل عدد قرار می‌گیرد. مثبت + منفی - صفر ۰

+ برای از استن علامت مثبت استفاده می‌کند - طول رو به جای چیز از راست حساب

نکته: در عمل از **لیست** چرا **بلم اون** و **بصورت متوالی** در **نظر من تبره**، یعنی صورت **متوازی** را ده **هفای پیش سر هم** به **یک متغیر اختصاص بدهیم** مثلاً **A** به **دوفقطیه B** و **A** داده **بیشتر از دوباره**

Subject:

Year . Month . Date . : $A, *B = 1, 2, 3, 4 \rightarrow A=1 \quad B=[2, 3, 4]$

برای دارن داره هی هنرالی در **متود format** باید قبل از دارن داره هی هنرالی **از نکته ۱**

ستاره * استفاده نمیع. **Print("{{}} , {{}}".format(*[1, 2]))** $\Rightarrow "1, 2"$

برای استفاده از لیست های **متغیر** در فرمت باید از جفت ستاره * استفاده ننمیکنیم

Dic = {'x': 1, 'y': 2}

Print("{{}} and {{}}".format(Dic))** $\Rightarrow "2 and 1"$

ساختار الی:

{ " [field-name] ["!conversion"] [":format-spec"] }

"{{}}".format(x=2) : نام لیستی که نیازی نظر نظر ندارد fieldname

: برای مشخص کردن یا تغییر نوع دوستاره a, b (ascii) و c (رستاکوئین) و d (رسته):
"{{!c}}".format("5") $\Rightarrow '5'$

: مشخص کردن فرمت داده. بر اصل دارن بخوبی هست که خود دارن

دارای یک ساختار الی هستند:

نوع داده های انتقالی مرتب شده بعد از آن صیغه های پردازنده ای
[[fill]align] [sign] [#] [0] [width] [grouping-option] [.P] [type]
① ② ③ ④

(P) : برای مرتب سازی اعداد بزرگ با استفاده از کاما. یا آند، اسلو، -، مثلا:

grouping-opt

"{{:,.0f}}".format(10000000) $\Rightarrow 10,000,000$

(۲) پرکردن طول میان با صفر (قرار دارن صفر قبل عد).

SABA

لستهای رسمت های تابله و

اعداد را بر اساس مبنای مسخه نمود با علامت مبنای سیان من درد. پنجم

مسخه

"{:#x}".format(15) \rightarrow 0xf

align

: رأسی چن وسط چن چی چن مسخه نمود. و می توان قبل آن

مسخه نمود که جاهای خالی را با رشته دلخواه پر کرد بوسیله \fill. علامت های align

عبارت از : < بزرگتر > و < کوچکتر > و تعان ^ و میں :

"x is {:+^10}".format(3) \rightarrow x is +***3+***+*

3. string (روشن رنگ) : جدید ترین روش است. تنها تفاوت این روش با روش دوم

این است که قبل از رشته حرف f را نوشتند سپس داده خود را داخل رشته درون برآورد

نمودیم.

متغیر

لیست ها (list) : نوع داده ای است که لیست از آیتم ها را درون خود ذخیره می کند.

List = [item1, item2] \rightarrow [داده] = نام متغیر

نکت: ما در پاسخ دهنده کیم کردن داریم shallow copy (ابی سطحی) و Deep copy (کیم عمیق)

(کیم عمیق) که در کیم سطحی الگوی معتبر رو که دارای لیست یاراده ای خاص نمود پاسخ دهنده

بصورت یک باره ذخیره می کنیم، اون مقدار صرفا به متغیر دوم ارجاع را داده می کند

و اگر متغیر دوم را تغییر بدم اون هم تغییر می کند. ولی در کیم عمیق این اتفاق نمی افتد

متغیر

Subject :

Year . ٢٠٢٣

Month .

Date .

لدار

امتحان

نهائي

٢٠٢٣

برای دسترسی به مقدار یک آیتم در دیکشنری، نام متغیر را نوشت و درون برآلت لدار این نویسید.

تایپ ها (Tuple) : نوع داده های هتوالی است که عیناً همان لیست است که دو تابع

بالست دارد: ۱- داده های درون tuple غیر قابل تغییر (immutable) هستند. ۲- بجهات

برآلت درون پرانتز قرار می شوند. tuple = (1, 2, 3)

(دیکشنری ها) Dictionary : نوع داده کلید-مقدار (key: value) می باشد، به هم

آیتم درون آن بصورت مقدار: کلید ذخیره شود. سینتакс: داده های درون آن

از قرار لفظی و با آیتم ها با {}()، از هم جدا شوند. dic = {"one": 1}

نکته ۱: کلید های باید از نوع داده Immutable باشند، اما مقدار هر آن داده می شود.

نکته ۲: از یک دیگر غنی تر از دیکشنری تباردار نموده چنانچه تباردار نمود مقدار قبل آیدی

می شود (کلید صعودی ساخته شود). dic = {"one": 1, "two": 2, "one": 3}

print(dic["one"]) => 3

مجموعه ها (Set) : نوع داده مجموعه ای است که از نوع mutable می باشد

سینتакс: داده های ذخیره شده و با آنها جواب نمودن بدل {1, 2, 3}

نکته ۱: داده های تباردار ذخیره می شوند، اگر دستی داده شود خود تبار حذف نماید.

نکته ۲: آیتم از نوع داده mutable ذخیره نمی شوند.

نکته ۳: از Index (لدار) و تکه بنده پستیابی می نماید.

نکته ۴: ترتیب را در ذخیره آیتم ها رعایت نمایند.

حلقة (١٠٠%): حلقة (ستوراي) تحيط بـ (ستوراي) (اتازماي) من خمسة تكرار، وهي كالتـ:

در پایه‌یون دو نوع حلقه وجود دارد: ۱- `for` - ۲- `while`. حلقه‌ها دارای ۳ رسم است:

کلید break: این دستور در زمان استفاده می‌شود تا آن فرآیند

جاءه دعاؤه في ذلك وعندما سمعه صرخة مهلاً هرداً من مدرسته أدرك أن هناك خطأً

٣) ملائكة الله يحيى (الملائكة المقربون): continue P. ٣٦٦

لیوی هم (ستود) اگر از continue نویسید و اجرا نمود (کامپیوتر مخفی) پرسیده باش

مثال ١٦: `if (x < 0) cout << "negative"; else cout << "positive";`

break (عکس) بی پایان برگشت (when break) ، (بستورات دادن)

دعا

..... می باشد. (ستور را نوشتہ سب ستر کھانے کے لئے سسیم دریا

صورة برقعه بدون شروط حلقه اجراءه تكرار. سينتيلس : while شرط :

2. for: یعنی به زبان هر آئیم در یک داده توالی. این حلقه برای کدام داده توالی دستور

بـ ترتيب مـ آيـم آنـ روـجـ آنـ هـيـاـسـ اـلـجـامـ مـ دـهـرـ دـيـتـلـ،

العنوان: كتاب المدخل إلى علم الأحياء

شروع ترتیب آزاد است

negative

range(1, 10): اعدادی میں از 1 تا 10 کے عدموں تک درج کر دیں۔ **range()**

for i in range(10):
 print(i, end="")

ZIP: چند توالی دریافت مکن، سبھ تباہی کوں دے دے دئے تو نیلے فراہمی دے

L1 = ["Ali", "reza"]

L2 = [1, 2]

ziped = ZIP(L1, L2)

Print(list(ziped)) \Rightarrow [(Ali, 1), (reza, 2)]

enumerate: یک لیست دریافت مکن و جو دلار صرائیم، و اذسیں لوگوں کی

map: دو آرگمن تابع و توالی (iterable) دریافت مکن۔ سبیں عنصر دادیں

map(func, list): syntax

def f(x):

 return x * 2

Print(list(map(f, [1, 2, 3]))) \Rightarrow [2, 4, 6]

دوسرا آرگمن تابع و توالی دریافت مکن۔ سبیں عنصر داد توالی را طبق

نتیجہ false یا true بونے تابع طبع دسوات تابع فیلٹر مکن۔ syntax:

def f(x):
 return boolean

 return (x > 5)

Print(list(filter(f, [1, 4, 7, 2, 9]))) \Rightarrow [7, 9]

تابع (functions) : تابع ها چیزهای هستند که یک ورودی دریافت کردن و براساس یک قاعده روس آن پردازش انجام داده و در نهایت خروجی را به صادر می‌کنند. مثلاً جرخ نوشت، کوشت را به عنوان ورودی لفظ و جرخ که در خرجی نام نهاد.

در پایتون تابع مانند ریاضی تعریف شود. دستور `def` نوشتہ سینه داخل پرانتز و موارد عملیات در زیر آن بصورت توزیعی نوشته شود.

`def f(x):` return : بمان این تابع بعد از انجام عملیات خروجی را به طور مسلسل داده باشد از دستور

`def f(x):` return Name: استفاده نمی‌کنیم. نکته: اگر `return` نوشتہ نشود بصورت پیش فرض تابع `f` برای این

`def f(x):` return x + 1: خواهد بود که `f(1)` دو برابر ۲ باشد و آن را با این معنی نمی‌گیریم. نتیجه این است که در اینجا تابع صفتی تعریف شده، بسیار استفاده شوند اما صفاتی بسیار اینکه تابع را فراخان کننند نمی‌توانند نوشتہ سینه برای تابع `f` درست نمایند. (برای دلیل دو دلیل)

نکته ۱: صفتی لازم نیست برای تابع وردی (پارامتر) یعنی نیست

* در ورودی هایی که تابع تکریف می‌کنند می‌رخدند از آن می‌رخدند، `argument` (گفته می‌شوند) برای این کلمه

* در ورودی هایی که تابع می‌صدزادن تابع را داده می‌سینه پارامتر Parameter (گفته می‌شوند).

نکته ۲: صفتی توانیم برای دادن آنرا هست از روی `*` (استفاده از `*`) دستگاه

لیست. می‌توانیم ساخته نوشت و تغایر می‌دهیم. (برای دلایلی دوستی داشته ام) مثال:

① `func(*iterable)` یا `func(**dict)`

(`set, str, tuple, list`) یعنی iterable = (دادهای توالی) ۱

نکته ۲: میتوان برای زمینه اگرچنان داده شد که باشد لور در ریاست مجلس برای پارامترها تابع مقدار پس فرض قرار داشم.

def f(a=0, b=1):

با بالعكس اگر تعداد آرگمنت‌ها بسته به آن میتوان پارامتر آخر را صورت تقریبی

def f(a, b, *c):

نکته ۳: الگو استاردر در پارامتر استفاده شود باید پارامترهای بعد از آن در آرگمنت

صورت نام-مقدار تحسین شوند.

f(1, b=2, c=3)

نکته ۴: الگو اسلن / در پارامتر استفاده شود باید پارامترهای قبل از آن بجهود آگهی

def f(a, b, /, c):

صورت معوقت تحسین نموده.

نکته ۵: از یادداشت (annotations) برای ایجاد این خود متغیر کنیم که در دسته‌ها دریافت و من کنیم در تابع از نوع قرار است باشد. و با خروجی از هم‌نوع

(اینستی) که دریافت و من کنیم در دسته‌ها در نهایت بتوانیم برای این باره (روی کرد همچنان تایپ (عنوان گذاشته). برای پارامتر از دونقطه به همراه

۱) متغیر او برای خروجی علایم‌های >- متغیری کنیم. در نهایت بتوانیم برای

۲) این یادداشت‌ها در دسته annotation استفاده کنیم.

def f(x: int, y: str) -> list:

SABA

Year: _____ Month: _____ Date: _____

فضاء نامه (Namespace): ذکر مبنای صورت (NameSpace) است که نامه ها و

متغیر ها و ... هایی که تعریف شده اند در آن قرار می کنند. این فضاهای از خود بزرگتر -

بنابراین فضاهای از خود بزرگتر (local < enclosed < Global < built-in) می باشند.

built-in: بروزترین فضای نامی است که تمام نام های تعریف شده پایتون در آن قرار دارد.

که واقع در چشم ما (ول ها وجود دارند) مثل تابع print، بازگشتن dir()،

آن ها را منسخه کرد.

Global: فضای نامی که مازول می باشد. یعنی هر متغیر یا متغیری که در آن فضای پایتون

تعریف شود در این فضای نامی کشید. با فراخان تابع (globals) این فضای نامی می شود.

enclosed: فضای نامی داخل یک تابع یا کلاس که خود درون مازول ماقرار دارد.

local: لوچلترین فضای نام که درون تابع یا کلاس وجود دارد. با تابع (locals) می تواند

حدوده نام (Namespace) که مخصوص لسته اینهاست دسترسی به متغیر را

نام فضای نام بگذارد که ترسی اولیت آن از لوچلترین تابع بالاترین (یعنی global) است.

از local به ترتیب (built-in) که مرتفع دسترسی:

def A():

x = 5 # enclosed

def B():

(x = 10) # local

Print(x) → 10 بدلیج و درجهای فضای

global

نکته: اگر بخواهیم از فضای local به متغیر فضای بالاتر (enclosed) دسترسی پیدا کنیم

باید متغیر که در فضای نامی بالاتر وجود دارد بخواهیم تغییر دلیم. از دستور def A():

$x = 10$

\uparrow def B():

nonlocal x

$(x) += 1$

دستوری به متغیر فضای global هم رعورت نماید فقط به جای دستور

از دستور global قبل از نام متغیر استفاده نمایم.

لambda (Lambda): یک تابع یک حضی و یکبار معرفی باشد از دستور کوتاهی.

را به عنوان یک تابع اجرا مکن. دستور lambda نویسند سین ورودی را مشخص کنید.

عملیات ورودی

بعد از دو نقطه: عملیات را صفت نماییم. syntax:

def func(x):

return x + 1

lambda x: x + 1

نکته: توافقی از lambda (ترابی) می‌شود filter و map استفاده کنید

لکارک (Iterator): آبجکت‌های هستند که آخرین وضعیت خود را در زمانه به طایفه سپرده

می‌باشند که آخرین بار کدام عضو از مجموعه iterable (مکالمه) در زمانه استفاده شده

این کار رو بجز از تبدیل iter (iterable) به iterator با استفاده از تابع (iterable) انجام

داده به این روش که هر بار از تابع next (iterable) استفاده کنیم برنامه خود را به عنوان داده

که میتواند حلقة for از نیام می‌کند

دکوراتور (Decorator): دکوراتورها بهم اجازه می‌دهند که تغییرات ساده‌ای

در اسیاه تابع فراخان) مانند قوانین مترکابی لاله‌ها ایجاد کنیم. یعنی می‌توانیم در راست

آن کسری دو و بار استفاده کرد. برای مثال دکوراتور می‌تواند تابع داریم که

لسته به عنوان ورودی دریافت کند و قبل و بعد از آن یا سه اضافه کند.

`def my_decorator(func):`

```
    def add_symbol(*args, **kwargs):
        Print("***")
        func(*args, **kwargs)
        Print("***")
    return add_symbol
```

`def print_name(name):`

Print(name)

`name = input("Name: ")` → Ali

decorated_func = my_decorator(print_name)

decorated_func(name) → Ali

Ali

۱ نکته: این نکته در دنده بررسی اینکه از دکوراتور استفاده نمی‌باشد متوجه جدید شد.

۲ نکته: در اینجا لاین قبل از تابع دور دنده نمایم دکوراتور باشد.

@my_decorator

def...

name = input("Name: ")
SABA

print_name(name) → Ali

Subject :

Year .

Month .

Date .

نکته : برای اینکه اطلاعات تابع مانند داده استریم (Stream) را سمت دکوراتور باز نویسی کنیم

اغلبین مره باید از مسائل functools استفاده کنیم. به این صورت که در حقیقت اول دکوراتور من نویسی from functools import wraps

def dec(f):

wraps(f)

...
حلقه

جنراتورها (Generator): تابع اس که عناصر یک iterator را تولید می‌کند، چن

بلای ویژگی نیازی به یون لیست و مассив آن از قبل نیست. جنراتورها همان فرآوری اصراف خود

بلای آنچه آن برمی‌گردد. برای تولید باید از حلقة استفاده کرد یا تابع next+ برای پیش‌گیراندن

از لکه‌لکی yield اجرای مولید عضو و درجاع آن به جای return استفاده کرد.

برای مثال مفاهیم توان ۲ هار اعداد یک تا ۵ را تولید کنیم:

```
def my_generator():
    for i in range(100):
        yield i**2
```

gen = my_generator()

```
for i in range(1, 7):
    print(next(gen))
```

نکته: وقتی از next+ استفاده کنیم تابع عده سمت دکوراتور سریع yield و اجرایش می‌کند سپس منتظر

من چون نیاز next+ بدم سبب دوباره yield اجرای

Print: می‌توانیم yield را دوست یا متغیر قرار بدم، اما مقدار اون متغیر

از صیغه ~~send~~ ~~to~~ send به اون داده می‌شود (send to ~~from~~ اون متغیر)

! (Return)

def gen():
 name = yield
 print(name)

g = gen()

next(g) →

g.send('Ali') → Ali

generator.throw(): Raise exception like StopIteration

generator.close(): close() من تونس جنایت رو بینم.

صیغه ایضاً: هر آبجکت صفت های مثل name و اینها را دارد

نه تنظیم جذو شدن هستند. برای دین صفت های می آبجکت اون رو داخل تابع

(attr) می نویسیم. برای ساختن صفت نام تابع نوشتہ بین قرار و نام صفت و مقدار مثل func.new_attr =

برای اینجا چنین تابع وجود دارد. افتدانه لرن(Chen): setattr(): ۱- برسی کردن

delattr(): ۲- حذف کردن وجود دیده صفت (hasattr(): ۳- بارگذارن: getattt(): ۴- هست

تابع بازگشتی (recursive): توابی هستن که درون خود اخود را فراخانی می کند. (خطه

). در ساختار این تابع ناید سُرط توقف اول می خورد و فرخانی خود نایمهای نشود.

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

SABA

تواتری و متعدد اعداد:

١. تابع divmod: دو آرگمنٹ رسمی و خروجی نسبی و باقیمانده دو عدد میں دھر.

٢. تابع sum: تعداد دو عدد میں (۱۵)

٣. تابع round: عدد را کم منند.

٤. تابع round(3.642, 2)

٥. تابع as_integer_ratio: نسبت عدد نشانی عدد (سری عدد) میں (۱, ۴).

٦. متداتی لیسٹ: List.methods()

٧. اضافہ کردن یا عنصر بے لیسٹ: append - ۱

٨. تعداد تکرار یا عنصر را نشان میں دھر: count - ۲

٩. اضافہ کردن یا لیسے بے لیسے مود رکھر: extend - ۳

١٠. نشان دادن اینسلس عنصر را دئے: Index - ۴

١١. اضافہ کردن عنصر در اینسلس موردنظر: Insert - ۵

١٢. عنصر را حذف کر دے و اُن را برمیں لرداز. آخرین عنصر (list.pop([index])) پس فرخ: ۶ - ۴

١٣. صرت ساز (list.sort([key])) لیسے بر اساس دوست داد دھر: sort - ۵

متداتی دیکشنری:

١٤. dict.fromkeys([key, val]) / جو دیکشنری ساز (dict) کا کام: fromkeys - ۱

Subject :

Year .

Month .

Date .

۱- get : لیست درایت مقدار آن را برمی‌گرداند و صورت وجود نداشت آن لیست

ب) جاری خطا مقدار None برگردان

dict.get([Key , [default_value]])

۲- items : یک لیست از آیتم‌های دلیلشتری برمی‌گرداند. هر آیتم مقدار داده شده تابعی باشد

Keys : یک لیست از کلیدهای دلیلشتری برمی‌گرداند.

values : یک لیست از مقادیر دلیلشتری برمی‌گرداند.

۳- setdefault : لیست دریافت می‌کند، اگر آن لیست دلیلشتری وجود نداشت مقدار

مقدار آن را برمی‌گرداند در غیر این‌صورت آن را با مقدار داده شده بدلیلشتری اضافه می‌کند.

dict.setdefault([Key , [value]])

۴- update : یک ساختار لیست مقدار دریافت شده و دلیلشتری را بر اساس آن بروزرسانی می‌کند

خلاصه نویسی (Comprehension) : نویسن دستورات حلقه و شرط در یک خط، به معمولاً

در لیست ها استفاده می‌گذرد. ساختار آن به این صورت است که دستورات را

(جدا از آن) با اسپیس

به این ترتیب درون لیست می‌نویسیم : ۱- عضوی د فراست append (بعد (آخرین دستور)

۲- حلقه یا شرط بدون دو نقطه (:) می‌باشد

comprehension :

lst = [i for i in range(3)]

normal :

lst = []

for i in range(3): <

 lst.append(i) <

اللّا إِلَهَ إِلَّا رَحْمَنُ الرَّحْمَنُ

Subject :

Year . Month . Date .

ساعات عملك

لقد اذ از این روشن ترین پروتوكول استفاده نماید جنباً تر برای ما ایجاد ننماید :

`gen = (x for x in range(100))` ؟ `print(next(gen))`

نحو ۲ - برای ساخت دیلیست ری داخل آنکارا لیست صورت مختار : للیتر سبیس حلتم یا سرچ . قابل

`key_val = [('Ali', 1), ('omid', 2), ('aref', 3)]`

`dic = {key:value for key,value in key_val}`

برای التود کردن `bytarray`, `bytes` دوتابع `decoding`, `encoding` داشته است.

برای `bytes('string', 'utf-8')` روش باستادی (یعنی بازبندی) نیست . صریح .

برای `str('3', 'ascii')` دوتابع `decode` و `encode` داشته است .

نحو ۳ - مدارک : الگوی چنین پلیج نو در تو داشته باشیم و بخواهیم بدونیم .

داریم سیس طولانی نیست تغیری ماژول خاصی دارد ایمپورت نمی میتوانیم به حالت خارجی داشت .

مسنون : اون رو تو فایل `init` پلیج اول ایمپورت کرد . سبیس استفاده ننمایی .
`from . import` هرچیز `init` فایل داخلی داشت .

`from Pkg import` هرچیز `init` داشت .

فصل هشتم: مارول ها و ستریپت

Subject: Year . Month . Date .

مارولها را پایتون نه خودش مستقیماً (صریح) نمی‌نگیریم بلکه در اینجا می‌توان script (سکرپت) نامید که با استفاده از مارولها بگوییم (فایل).

در حالفا برای ما مارول را در میان اسلوبیت module می‌دانیم که مجموعه‌ای از مارول‌های مرتبط به دلیل تقریباً همانند شود، توسط اسلوبیت مارول اینبورت شود و استفاده شود. یعنی مدل است.

Package: مجموعه‌ای از مارول‌های مرتبط به دلیل تقریباً همانند شود، توسط اسلوبیت Package معرفی می‌شود.

در آن سازیم. سپس پس از مارول مارول قابل اینبورت خواهد بود.
برای اینکه یک پوشه تبدیل به پلیج نمود باید یک فایل پایتون با عنوان `init.py` در آن بسازیم. سپس پس از این مارول قابل اینبورت خواهد بود.

ساختن مارول: یک فایل پایتون ایجاد کنیم برای استفاده آن در اسلوبیت خود از `import` دستور.

استفاده از آن: برای اینکه مارول دسترسی به توابع باشند آن مارول نام مارول نوشته نظر.

`import mod`

`print(mod.func(x))`

کام کرون مارول

ملته ۱: این اینبورت مستقیم نام می‌توان لاز دسترسی `from mod import func` استفاده کرد.

ملته ۲: از مارول `mod` `import` نمی‌توان مارول را در بازه اینبورت تغییری در آن ایجاد نهاد.

`from mod import func`

`reload(mod)`

بعد به حالت اول برگردیم.

ملته ۳: از داخل مارول مارول نام نه اصراف نه در حقیقت وقت اینبورت من لست اجرایش.

اونها باید داخل مارول سُرخ بزیسیم: `"__main__"` که

بازگشتن فایل: برای بازگشتن فایل از نام open استفاده می شود. آنرا این امثال کرد:
`open(file, mode='r', buffering=1, encoding=None, errors=None, newline=None, closefd=True, opener=None)`

آنرا mode: مستحب می کند خالی که برخواهد بازگشتن به جهت حالتی بازگشته مثلا برای

خواندن یا نوشت. در ادامه مقادیر این آرگمندان مطابق می شود:
باشد خلاصه این

۱.۱ (۲۱): برای خواندن، فایل هست، اسکار (cursor) در ابتدای فایل، از فایل وجود نداشته

۱.۲: فایل را باحالت باسیز (binary) بازمی نمایم.

۱.۳ (۲۲): برای نوشت در فایل، محتوا قبلی پاک نمود، اسکار را ابتدا فایل، از فایل

وجود نداشته باشد آن را ایجاد و سپس بازمی نمایم.

۱.۴ (۲۳): برای نوشت در انتها فایل، اسکار را در انتها فایل، از فایل

۱.۵ (۲۴): عین سایر منابع، فرق آن اینست که فایل نباید از قبل وجود داشته باشد تا باز نوشته شود.

۱.۶ (۲۵): با اضافه کردن + من توان همچنان هم خواند و هم نوشت. مثلا:

نوشت در فایل: برای نوشت از هند write استفاده می کنیم. (w و w+)

f.write()

۱.۷ (۲۶): یک iterable دریافت می کند و در فایل می نویسی. (مقادیر با پرسش باشند)

۱.۸ (۲۷): اگر فایل قابل نوشت باشد مقدار True برسان لردن

خواندن فایل: برای خواندن از هند read استفاده می شود. این هند حجم صنیع

SABA

۱.۹ (۲۸): این لست فایل را برمی خواهد داد استفاده می کنیم

الرداده مقدارش از بازیز اکثر باند، در نهاده دیگر سه تون هست رام نویسی
ولکن آنکه بیشتر باند همچون بازیز پیر نموده آن را حالت میکند. لیکن با این نویسی

Subject : _____
Year : _____ Month : _____ Date : _____

که در فایل وجود دارد با همتای از اندیمه میتوان قدرات ایجاد کننده مورد نظر برای

f.read([2]) خواندن را به عنوان آنکه این را در عین حال

readline() هم یک خط از فایل را در خواند.

methd readlines(): همان متن را خوانده و همه خط را به عنوان یک عنصر لیست برمیگردان

با فرمان (buffer): یک حافظه میان میان دو سهندت افزای مجزا. که به صورت هولت

اطلاعات را میگیرد. برخاسته زمانی تو فایل هم نویسی، لذا تمام عموم منظمه باقیماند

نه نویسی) لوسن آنی می تفسم با فرمان رو خانی لیست با متد flush درجا نویسی شود

یا می تود به جای اینلار آنکه f.flush() را بلبر میگردان و آر اراد

ملهنه ۱: مقدار ۱ - ساینر پیش فرض با فریسل می باشد که ۱۹۷۸ است.

ملهنه ۲: مقدار ۱ با فریسل می باشد که مورد تصور خنچی محاسبه کند هنین هر خط را در نامه ذخیره کند.

ملهنه ۳: مقدار دلخواه می باشد که بسته برای mode با سیزی فایل استفاده شود

اشاره کر (cursor): همان خاندن و نویسن اشاره کر در فایل جایه باش کود

برای نشان دادن موقعیت فعلی از متد tell و برای تغییر از متد seek استفاده میکنیم که قدر این مکان

مقدار seek: آنکه اول موقعيت را مشخص کرده بین و در آنکه دوم اینم

f.seek([3, 0]) از کجا بخواهیم که مساحتی کیم.

SABA

اول فایل موقعيت

دستور with : کاریت آزاد کردن منابع سیستمی مانند، یعنی بله از اجرای فایل دستورات بدین و اگر منبع نه میتواند آنها اشغال شود، باز هم تا مکانهای را به اعماق منابع دستورات (منبع) میگذرد باز لبردن فایل، در وقت دستورات بدین هم میتواند فایل را بسته و میتواند باز باز کند.

with open('file') as variable:

دستورات باز کردن فایل هم باز کرد دستورات

فایل های استاندارد : پاپولر دارای سه ۳ نش. اصلی است `in` که توافق Print و

`input` هم توافق آنهاست. `stdin`: هر آنچه در میخواهد `stdout`: روز صفحه نمایشی نویسی

`stderr`: ارور را در صورت خطا این فایل را باید از ماژول `sys` ایجاد کرد.

ملکت : اینها فایل هستند و توسيع متدهای فایل استفاده میکنند فیل:

ناملن json: نوع فایلی است که برای انتقال اطلاعات در سطح وب (الانترنت) بود. این فایل

قابلیت حفظ نوع داده ارسالی را را را که در فایل به صورت لیل-مترا ذخیره میکند، دارد

و بروول از هم جدا نمیشوند. تفاوت نوع داده های باین و همیشی: مدل / Python

`int/float/number`, `True/False/true/false`, `None/null`, `dict/object`
`list/tuple/array`

۱. تبدیل از json به python : این کار به ۲ روش انجام میگیرد . ۱- تبدیل رشته json

(رشته ای از قوانین نوشتاری JSON پیویسی شده) به فرمت پایتونی که با استفاده از متد `loads`

در ماژول json این کار انجام میگیرد. مثال صدای بزرگ

```
import json  
js = 'null' → json.loads('null')  
print(json.loads(js)) → None
```

۱. تبدیل فایل js به فرمات پایتون . ابتدا فایل را باز کرده و بین از مت

```
with open('test.json') as j:  
    py_format = json.load(j)
```

۲. تبدیل از python به json: این هم دو مت دارد . ۱- تبدیل داده پایتون به رشته

```
py = None ; j = json.dumps(py) # json.dumps() با استفاده از مت
```

print(j) → null

```
with open('myj.json', 'w') as jf:  
    json.dump(None, jf)
```

۳. نوشتن داده پایتون در فایل json ، با استفاده از مت

file: میتوانیم با استفاده از آن رکار نمودن فایل json را مستحکم کنیم . indent=4

فایل CSV: یک فایل است که اسماً جدا شده اند (comma separated values)

لینک در سطر اول آن ویژگی دارد که سطر های بعدی مقادیر قرار میگیرند که با ویژگی

name, age از هم جدا می شوند . همان

Ali , 19

۱. خواندن از CSV: برای خواندن از مواردی باید استفاده میکنیم . ابتدا فایل را باز

کرد و سپس از دستور DictReader استفاده میکنیم . این دستور یک iterator

بین ایوانز نم میکند که میتوانیم پیمایش کنیم .

2. نوشتن در CSV : ابتدا باید آبجکت writer را ایجاد کنیم . سپس با استفاده از متد `open` with `open('f.csv', 'w') as c:`

`writer = csv.writer(c)` در فایل نویسی . `writer.writerow(['name', 'age'])` مقدار داده را در یک خط از فایل نویسید .

چند لیست درون یک لیست کلم در ماتریس کند و در عنصره را در یک خط از `writerows` داده و آن را در فایل نویسید .

فصل دهم: کلاس

Subject : Year ۱۳ Month . ۵ Date . ۹

کس دلایل: در پایتون هم چیزی نیست، است. مثلا وقتی مایک متفاوت عددی می سازیم

$x = 5$ در واقع یک خوبه (instance) از کلاس int می باشد که مقدار ۵ را

آن داده شده. رفتارها یا Behavior آن هم توابع متدهای آن قابل انجامات

کپسول سازی (encapsulation): یعنی کارهای قرار دارن method و attribute،

همه این مساحت سی جدید. مثلا جمع شوندگان فضاهای بروی ساخته می شون

پنهان سازی (Data hiding): بعد از فرازین لپسوک سازی، ممکن شیوه ای به قوی

متفاوتی برای داده ها private یا public بدان دسترسی نداشته باشند. یعنی خارج از کلاس قابل

دسترسی نباشد.

انتزاع (abstraction): استرالیک معنی کلی از مفهوم های باشند مثل حیوان یا

ماشین کلی معنی کلی از چیزی که مختلفی های باشد. انتزاع یعنی توابع به الگوهای بدون توابع زیاد

به جزئیات. مفاهیم انتزاعی هودش کلی معنی هستند (الکو) یعنی خود حیوان

abstract

وجود ندارد مثلا نسیر کلی غونه از حیوان است وجود ندارد؛ حیوان کلی معنی انتزاع هست

concrete

ترکیب (composition): جمع آوری چندین نسی، باهم برای ایجاد یکی نیست، جدید، برای زدن

استفاده می شود، نه می شود، مثلا از شرک دیگر باشد در composition انسان، وابسته

بالایی به حجم دارد. ویژگی ها: استرداد ممتد از هم کاربرد ندارند - خالیان دارند

نحوی نشانه ای

تجمع (aggregation): رابطه یک شیء با شیء دیگر که مخفیت ندارد ترکیب است. مثل رابطه بانک (Bank) با مشترک (Customer). (مشترک ها: بانک های مختلف از جمیع موجودات هستند)

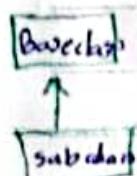
وابستگی (Dependency): رابطه یک شیء با شیء دیگر که مخفیت دارد تجمع است. مثل رابطه بانک (Bank) با مشترک (Customer).

وابستگی (Association): رابطه یک شیء با شیء دیگر که مخفیت ندارد تجمع است. مثل رابطه بانک (Bank) با مشترک (Customer).

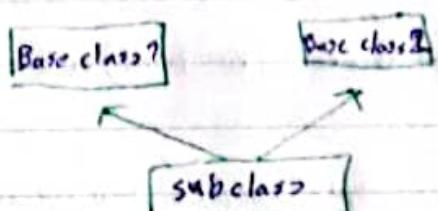
ادش بری (Inheritance): یعنی یک شیء بازیم که علاوه بر داشتن خصوصیات و ویژگی های خود، از شخصیت دیگر اوت بری (Inheritance) دارند. و اینها از خواص دارند.

راهنمایی داریم. مثلاً انسان، شکر و زنبور عرب از کلاس حیوان اوت بری دارند.

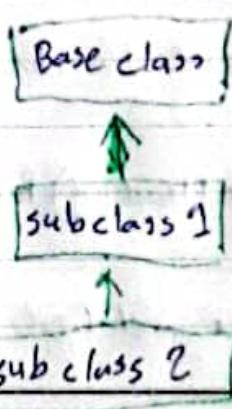
علاوه بر خصوصیات و ویژگی های خود هر دو دارای خصوصیات لاین حیوان دارند. انواع اوت بری:



۱. اوت بری از یک کلاس بالاتر: single.

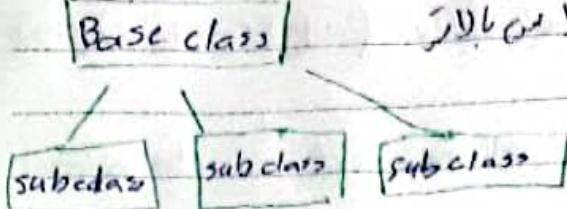


۲. اوت بری از چند کلاس بالاتر: multiple.

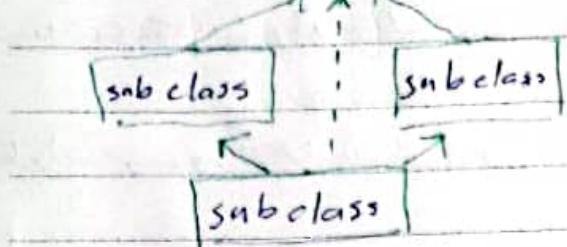


۳. اوت بری زنجیره ای: multi-level.

اولت برسی هندل اس ارث و اینو با لارج : hierachial ۴



اولت برسی ترنسیپ (جنید) : Hybrid . ۵



جنید ریضی (Polymorphism) : یعنی پیاده سازی یک رفتار به شکل های وریثی های مختلف

مثل رفتار حمل کردن در دو حیوان گربه و ماشه به دو روش متفاوت پیاده سازی شده.

یا مثل کلاس مایبی به اسم شکل نه هست محاسبه مختص داره و کلاس های دایره و مربع ازش

اولت برسی کن این کلاس های محاسبه مختص و عر کدام به روش فرد پیاده سازی شدند.

Polymerphism عبارت است:

معمولاً این نوع جنید ریضی استنادی است. یعنی کلاس پایه داریم که

کلاس هایی که از آن اولت برسی می شوند لازم است منتها آن را برای خود بازنویسی

کنند مثل مثال حیران نموده شد. **override**

در این نوع یک تابع نسبت بتواند بروزی نوع های مختلف

نماید. مثلاً تابع `add()` می تواند `int` یا `string` باشد دستور واحد

(complete time) Ad hoc ۳ : همانه پر امتیز است ولی به باید تابع دارند چند نام
با نام هستند است که بسته به نوع راهه ارسالی تابع مخصوص با خود ام (برگردان در زبان C++) مفهوم

method overloading

کلاس های

همه چیز در پایتون یک شی است. مثلا وقتی من نویسی `x = 5` که شی از کلاس int ساختم

هر شی، خاکی باشد غیر از کلاس باشد.

هر کلاس خود یک شی، از کلاس type است و همه کلاس ها از کلاس باشند object این بقیه اتفاق

مفهوم کلاس در پایتون برابر با مفهوم type است

ساخت کلاس (Class) : برای ساخت کلاس از دستور class استفاده می کنیم. مثال:

class Ali : class child(P):

حالا که کلاس داریم برای استفاده باید از کلاس یک شی، (instance) بسازیم. برای

اینکار یک متغیر ایجاد کرده و مقدار آن را کلاس خود با پرانتز قرار دیم. مثال:

`obj = Ali()` آنرا با نام Ali بآنام زده داریم

همانطور که قبل از نزد برای شی خود صفت (attribute) می توان اضافه کرد با متن

`obj.width = 10` نام شی سپه نقله و نام صفت. مثال:

Subject : متد های کlassen و کلاس های وارث قابل دسترسی است
Year . ۰۳ Month . Date . ۱۲

* برای اضافه کردن هستی به کلاس خود باید درون کلاس تابع تعریف نماییم

و پارامتر self را به آن داده و استفاده کنیم. مثلاً من خواهیم زماین که از متدهای

less استفاده شده از وزن سیزه زده به واحد کم شود.

class Ali:

def less(self):

 self.width = 1

init: یک متد خاص (special method) است که خطیب مقادیری اولیه بهش

دادار. برای اینجا بخواهیم هر شیخی که می سازیم باید attributes اولیه داده شود باید

از init استفاده ننماییم بلطفاً در کلاس زیر هر شیخی، این مساحت شود داشتیم

class car: color

def __init__(self):

 self.color = "white" → هر چند داریم صفت color حاوی بود

بعد از این متد

نکته! این متد همچیج return ندارد و آن تعریف شود اگر در برخاسته ننماییم

کاربردهای () underscore: استفاده از خط ترمه در نامگذاری ها

۱. استفاده نکلمه های: زمانی که متغیرها را می سازیم استفاده از نواره

۲. استفاده قبل از اسم هستی های: مسفعی من یعنی که متغیرها از نوع ^① protected

۳. استفاده پس از اسم هستی های: زمانی که این متغیرها با الیه لایی های مشترک است

و طبقاً تابع built-in پاسخون می باشد

SABA

* = هستی های کلسن دو آند، اسلو، فرازی یا پرمیوم. -- init --

۴. استفاده دولایه خطیر قبل از اسم متغیرهاست: مستحب من لد متنها از نوع `Private` است

Class A:

`--max = None`

`Print(-A--n102)`

متدهای خاص `str` و `repr`: وقت طاز درستور `(x)str` استفاده نمی‌کنیم، یعنی

در واقع هند جادویی `-str` - از شیء `x` رو اجراء نمی‌کنیم هر لایسنس دارای متدهای `-str` -

و `-repr` - می‌باشد. `str` در واقع چیزی است که برای کوچولان باید و دیگالر نماین `Print`

`repr`: خروجی آن از شیء `x` کاملاً خروجی ای است که با همان صورتی داشته باشد

لکن از زمانی که بخواهی و قیمتی شنی، رو داخلی برینت قرار گیرد چیز خاصی (مستحب ساز) نیست

چاپ پس از `-str` - رو در لایسنس تعریف نمی‌کنیم و اون رو تغییر بلدم؟ برای `repr` هم به همین صورت

تفاوت `instance` / `class` `attribute`: `instance` صفات فقط مخصوص اون شیء

هستند، اما در دیگران صفات مختلف تریم شده از آن لایسنس

هستند. برای ساخت `class` اگر فیضت یک متغیر در لایسنس خود سیاریم

ارت بری از لایسنس `built-in`: ما من توانیم برای ساخت نوع داده خود از لایسنس

را خلی ارت بری لنمی‌کنیم و قابلیت دلخواه به آن اهتمام نمی‌کنیم، مثلاً در منطق زیر یک

Subject :

Year : Month : Date : ۲۰۲۱

عنوان و نام و نسخه از دلایل استفاده از دلایل این کلاس را درست نمودند.

A class method

def m():

pass

A class method

هیچ کلاس باشی برای ورودی دریافت نمی کند اما معمولاً می تابع static methods

استفاده از دلایل این کلاس معمولاً بجهت مرتبط بودن با کلاس ساخته می شوند. با استفاده از دلایل این کلاس

برای فراخانه property استفاده پیش پرده بدون استفاده از دلایل این کار بر این منظمه می شود.

صیغه کار بر از این منظمه استفاده می شود. یعنی اگر بر لازم نیست این متد را از دلایل

جدا نماید مثمرت عادی و صیغه مقدار attribute را در نظر دو در پیش پرده

این منظمه خود کار غرضمند نمی شوند. برای این کار ابتدا با class attribute

محاسبه و ناتایج property را از نزدیک و منظمه ای setter و getter

class attr getter setter method

نام = property(get-name, set-name).

این دستور دستیاری است

پس از آن دستور دستیاری است

Print(class.name)

کلاس انتزاعی (Abstract class) : کلاس های انتزاعی خود منظمه خود ندارند و تنها از

آنها ساخته خود و فقط از آنها ارث بری می کنند. برای ساخت این کلاس از دلایل این کلاس

from abc import ABC, abstractmethod

سین کلاس خود را از ABC ارث بری می کنیم و برای منظمه از دلایل این کلاس از دلایل این کلاس

SABA

استفاده می کنیم

برای تعریف عملکردن برای لایس (type) خود باید متدها: operator overloading

و جادوی مربوط به عملکردار لایس خود را باید override کنیم: برای اینکه ستد

class A:

def __add__(self, other)

برای عملکردار + می باشد.

slots: یک صفت جادوی است که برای محروم کردن و مستحب کردن خصوصیات

قابل ایجاد در لایس استفاده می شود و مزایای آن صریحه جوی در حافظه و دسترسی

رسویچ ترکیب اتاریسوتی باشد. مثلاً: class A:
الذی هر چیزی از این لایس فقط مجاز است این ('b',
لایتھا این داشته باشد)

نکته: با استفاده از صفت dict برای آن لایس غیر قابل برداشتن

نکته: در اینجا نمایم خود و برای هر لایس خواست.

ساخت لایس iterable: برای ساخت لایس iterable باید از صفت جادوی

iter در این لایس استفاده کنیم. برای iterator نیاز است next استفاده کنیم.

نکته ۱: زمانی که حلقه for می شوند، در پیشترین مرتبه صفت iter را اخراج کرده سپس

next اجرا می کند.

نکته ۲: برای اینکه وقتی جدا next را اخراج کنیم، حلقه استفاده کنیم و هر دو می

نتیجه بعد از iter در صفت return self را برمیگردانیم و در اینجا و در

صورت next میاده شو.

SABA

(۱) برای معرفی صفات می شود که در قالب دلخواه استفاده می شود

لوع داده (باید خود را ساختم نه متن) فتحت عنوان search هم دارد.

class myList(List):
 def search(self):

Poss

: برای حالت خوب کوئن یک متد از سوپر کلاس در زیر کلاس باشد overriding
نایاب باشان نام ساخته شده و جایگزینی انجام شود

نکته: برای دسترسی به متد اصلی سوپر کلاس از زیر کلاس بعد override کردن

bاید از تابع super() استفاده کنیم.

انساید قابل فراخوانی (callable object): برای اینکه بسیم آینده قابل فراخوانی هست

با این از تابع callable استفاده کنیم. (انساید) هستنند به بود از انساید پرانتز قرار

گرفته و وجودی دریافت می کنند یا بدن ورودی فراخوانی می کنند)

برای اضافه کردن قابلیت فراخوانی به شی خود، باید متد از متد های خاص بynam

call — در کلاس خود می سازیم.

انواع متد: در کل ۳ نوع متد وجود دارد.

۱: هیچ متد هایی هستند که در کلاس معمولا ساخته می شوند و در

۲: self را به عنوان ورودی (آیندلت) در گافتند و مثلا برای آیندلت صفت تغییری کنند

۳: همه هایی که در کلاس ایجاد می شوند class method

SABA

Class = type

۱. مس = نوع را

Subject : _____
Year . Month . Date .

لایسنس است که در زمان استفا context manager : context manager

(ا) از دستور `with` در لینک پرده اجراء شود، نه قبل و بعد انجام دستورات

داخل بمناسبت اجراء شوند. که این `with` context manager (with context manager)

و هنگام خروج `exit` و `enter` از `with` فیکت `context manager` دارند. برای ساخت `exit` و `enter` از `with`

را برای کلاس خود تعریف ننم. سپس هر

موقع خروج این دستور این دستور اجراء شوند. که `with`
with `class()` or `obj`:

Pass

نطٰل یازهم : مدیریت خطا
Subject: مدیریت خطا

Year: ۲۰۲۳ Month: آگسٹ Date: ۲۸

مدیریت خطا: برای مدیریت خطا اصولی نہ حسلام ترنا ممکن است رفع دهد، با این از دستورات try و except استفاده کنیم. روش پیاده سازی به این شکل است که کوچکی که استفاده ایجاد خطا را دارند در بدنه try و except نویسم سپس برای هنل کردن

try: کوچکی صورت نظر را در بدنه except من نویسم.

Print("x" + 10)

except: ➔ (except ValueError) کوچکی صورت خطا را در بدنه except نویسم
Print("error")

دستورات بعد

try: دستور else بعد از except برای زمان استفاده من تواند بدهی try کامل اجرا نمود خطا

دستورات بعد

finally: دستور finally که خطا را رفع نمایند در هر صورت اجرا من تواند

raise: (سینه raise برای فراخانی ارور استفاده من تواند. فنا را مجاز نمایی) نویسم

که بقیه هم استفاده می کنند، در صورت استفاده اینها raise کوچکی برای خطا برای اینها raise <errorname>

ساخت استثنای exception: برای ساخت استثنای ساختی با قیمت لایس سازی کردن

کلاس exception ارت برای کندو سپس من توانم آن را raise کنم یا در except استفاده class my_error(exception):

def __init__(self, msg):

self.msg = msg

super().__init__(self.msg) ➔ (کوچکی پیام که کلاس بالاتر برای من می سازد)

ساخت دکتور بالام : برای اینکه باید از همان کلمه init و call استفاده نماییم.

ابدا از اتفاق تغیرات آن را معاذ بزنیم.

```
class dec:  
    def __init__(self, func):  
        self.func = func
```

```
    def __call__(self):  
        do something before  
        obj = self.func()  
        do something after  
        return obj
```

و set, get است که خاصیت از منهای جادویی descriptor کند.

کرده باش وظیفه این لاس این است که برای override و delete

که سری رفتارهای مستحب نباشد، مقدار دهن شود و با حفظ class attribute

برگردانده شود (دقیقاً مانند مکالمه property)، تفاوت این با این است

که دیگر نیاز نیست برای هر یونیک این property این کلیل است.

```
class dec:  
    def __init__(self, name):  
        self.name = name
```

```
    def __get__(self, instance, owner):  
        return self.name
```

```
    def __set__(self, instance, value):  
        if value != "ninja":  
            return "permission denied!"
```

SABA

Self, name - value