Python

پنجشنبه, ۲۸ اسفند ۱۳۹۹ ۰۱:۳۰ ب.ظ

پایتون

VSCوارد شدن به تنظیمات
ctrl + shift + p

VSCوارد شدن به بخش شورت کات های
ctrl + k + s

https://gist.github.com/bradtraversy/b28a0a361880141af928ada800a671d9

.....
'1' or 'mahdi' or "mahdi" -> string
1 -> int
0.1   -> flout

.....
' ' , " " برای رشته های کوتاه -> Ctrl + /
""" """ برای رشته های بلند -> Alt + Shift + A

.....
در داخل سینگل کوتیشن باید از دابل کوتیشن استفاده کرد و بالعکس یا
قبل کوتیشن داخل بک اسلش بزنیم هم کوتیشن اول و هم آخر برای پرینت کردن یک متن در خط بعد
(x1 = ' ali goft\'salam\' ')

.....
برای پرینت کردن متن زیر هم به صورت زیر
1
2
3
4
x1 = "1\n2\n3\n4\n" -> 1\n 2\n 3\n 4\n <= یا هرچیز دیگرhello\n
print(x1)
1
2
3
4
و اگر بخواهیم
1
2
 3
 4
x1 = "1\n 2\n 3\n 4\n"

_____
اگر بخواهیم بین دو عبارت فاصله بیندازیم
print('hello \t world') -> hello   world   9 space
print('hello \f world') ->
hello
      world
.....
برای پشت هم آوردن دو یا چند عبارت
name = mahdi
family = ebi
print(name + family) -> mahdiebi
برای پشت سر هم آوردن و با اسپیس(بین دو یا چند عبارت اسپیس باشد)
print(name + " " + family) -> mahdi ebi
print(name + "   " + family) -> mahdi   ebi

.....
string + integer = error ->
name = mahdi
age = 14
print(name + age) -> error (<- (استرینگ و اینتیجر نمیتوانند با هم جمع شوند
عدد با عدد و عبارت با عبارت))

.....
نوشتن متغیر در کنار عبارت(+=)
در اعداد (=+ , =- , =* , =/)
name = mahdi
name = name + "ebi"
print(name) -> mahdiebi
با استفاده از قابلیت داینامیکالی تایپ پایتون))
یا
name = "mahdi"
name += ebi
print(name) -> mahdiebi
در اصل می گوییم حرف ها را با هم جمع کن و بزیز داخل خودش

.....
توان = (**)

.....
باقی مانده عدد (۶۸ % ۶ = ۲) = %

.....
// = > بدست آوردن خارج قسمت کسرprint((۶۸//۶ = ۶) <) /۶% ۶۸ (-۶۸

.....
) *مهم*formatingنوشتن بقل هم (
username = "mahdi"
password = 'mahdi1384'
earmark = "username is " + username + " and password is " + password
print(earmark) -> username is mahdi and password is mahdi1384
یا
earmark2 = f"username is {username} and password is {password}"
print(earmark2) -> username is mahdi and password is mahdi1384

برای اعداد هم می شود
num1 = 2
num2 = 8
sum_number = f"sum numbers is {num1 + num2}"
print(sum_number) -> sum numbers is 10

.....
نوشتن ایندکس عبارت[]
0 1 2 3 4
mahdi = m a h d i
username = "mahdi"
index1 = username[1]
print(index1) -> a
به عبارتی دیگر
print(username[1]) -> a

.....
نوشتن شرطif
rank = 1
if rank == 1 :
  print("you win #1")
  print("GG man")
elif rank == 2 :
  print("#2")
elif rank == 3:
  print("#3")
else :
  print("You lose")
-> you win #1
GG man
حالا اگر مثلا ۴ (که موجود نیست در ایف و ال ایف) را در رنک قرار دهیم فرمان ایلس فعال می شود و متن
you

برای ما نمایش داده می شود

یا در یک خط

rank = 1
print("you win #1") if rank == 1 else print("you lose")
->
you win # ۱اگر رنک ۱ بود->
<-اگر رنگ غیر ۱ بود you lose

.....
تغییر دادن تایپ عبارت(...و int به str(از
age = "14"
print(type(age)) -> <'str'>
agefloat = float(age)
print(type(agefloat)) -> <class 'float'>
print(age) -> 18
print(agefloat) -> 18.0
وبالعکس

.....
برسی با شرطif
```
rank = 2
if rank is 2 :
   print ('are')
else:
   print('na')
```
-> are
برسی تایپ تایپ عدد
```
if type(rank) is int :
   print ('are')
else:
   print('na')
```
-> are
یا
```
if type(rank) == int :
   print ('are')
else:
   print('na')
```
-> are

.....
ایندکس در لیست index
```
            0       1         2
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```
->benana
```
thislist = ["apple", "banana", "cherry"]
print(thislist[-1])
```
-> cherry
```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```
->['cherry', 'orange', 'kiwi']

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```
->['apple', 'blackcurrant', 'cherry']
برسی تعداد اعضای لیست
```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```
->3

اضافه کردن به اعضای لیست
.append
اضافه کردن به اعضای لیست در ایندکس خاص
```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```
->['apple', 'orange', 'banana', 'cherry']
حذف کردن اعضای لیست
```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
```

```
print(thislist)
->['apple', 'cherry']
```
حذف کردن یکی از اعضا
```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
->['banana', 'cherry']
```
حذف کل لیست
```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
-> []
```
سلکت اخرین ایندکس
```
print(myColors[len(myColors)-1])
```
چک کردن بودن آیتمی در لیست
```
thislist = ["apple", "banana", "cherry"]
vogodbenana = "benana" in thislist
print(vogodbenana)
-> True
```
نوشتن کل اعضای لیست با استفاده از for
```
thislist = ["apple", "banana", "cherry"]
for color in thislist:
   print(color)
```

.....

جایگزین کردن چیزی در استرینگ

عوض کردن بخشی از استرینگ
```
string = 'hello hello hello'
print(string.replace("he", "jo"))
-> jollo jollo jollo
string = 'hello hello hello'
print(string.replace("he", "jo",2))
-> jollo jollo hello
```

.....

جداسازی استرینگ
```
txt1 = "welcome to the jungle"
txt2='hello,world'
txt3 = "apple#banana#cherry#orange"
x, y, z = txt1.split(), txt2.split(','), txt3.split('#',1)
z2 = x = txt3.rsplit('#',1)
print(x, y, z)
print(z2)
->
['welcome', 'to', 'the', 'jungle'] ['hello', 'world'] ['apple', 'banana#cherry#orange']
->
['apple#banana#cherry', 'orange']
```

.....

مشخص سازی استرینگ

```
txt = "    banana    "
```

```
x = txt.strip()
print("of all fruits", x, "is my favorite")



txt = ",,,,,rrttgg.....banana....rrr"
x = txt.strip(",.grt")
print(x) ->banana  #'banana '
```

روی اول و آخرش تاثیر میگذارد

.....

برسی مساوی بودن با(==)و بولیان

```
x1 = 1
x2 = 2-1
x3 = 4-3
print(x1 == x2)
print(x1 == x3)
->
True
True
```

.....

) (==)فرق isدر مقایسه(

is -> ماهیت متغیر ها را برسی می کند

== -> مقدار متغیر های را برسی می کند

```
x1 = ["mamad yaghobi"]
x2 = x1
x3 = list(x1)
print(x1 == x2)
print(x1 == x3)

print(x1 is x2)
->
   True
   True
   True
   False
```

.....

(return boolean)

مساوی بودن ۲ متغیر

```
num1 = 1
num2 = 2
print(f'num1 = num2 ? : {num1 == num2}')
-> num 1= num2 ? : False
```

.....

مساوی نبودن یک متغیر

!= -> مخالف

```
num1 = 1
num2 = 2
print(f'num1 != num2 ? : {num1 != num2}')
```

-> num1 != num2 ? : True

.....
برسی بزرگ تر کوچکتر بودن دو متغیر
print(f'num1 > num2 ? : {num1 > num2}')
-> num1 > num2 ? : False

.....
برسی بزگ تر مساوی بودن دو متغیر
print(f'num1 >= num2 ? : {num1 >= num2}')
->num1 >= num2 ? : False

.....
and عملگر
((و
userage = '19'
usergender = "male"
if userage >= '18' and usergender == "male":
   print('vaght sarbazi')
else :
   print("bishin to khone")
->vaght sarbazi
.....
or عملگر
((یا
weather = "sunny"
if weather == "sunny" or weather == "cloudy":
   print("mirim shomal")
else:
   print("mi shinim khone")
-> mirim shoma
.....
not عملگر
((بولیان را برعکس میکند
isBrotherComming = False
if فقط بولیان ترو رو بر می گرداند
if isBrotherComming: (False)
---
if not isBrotherComming: (True)
   print("my sister said : i wont come")
->my sister said : i wont come
.....
age = 45
if (age > 1 and age <= 8) :
   print("1.99$")
elif (age > 8 and age < 70) :
   print("9.99$")
elif (age >= 70) :
   print("4.99$")
else :
   print("try again")
(یا (بدون ال ایف
if not ((age > 0 and age < 8) or age >= 70) :

```
   print("9.99$")
else :
   print ("try again")
```
.....
پروژه ای با input

```
print("hi my name is python \nesmet chie?")
username = input()
print (f"salam {username} kari az dastam bar miyad?(are-na)")
kar = input()
if kar == "are" :
   print("chi kar?")
elif kar == "na" :
   print("bash kari dashti bego")
kar2 = input()
print("anjam shod!")
```
.....
دستور شرطی یک خطی
```
x = "hello"
assert x == "hello"
```
اگر درست باشد یک مقدار ترو را برمیگرداند و برنامه رو ادامه میدهد
اگر غلط باشد کد همونجا وای می ایستد
```
x = "hello"
assert x == "goodbye", "x should be 'hello'"
```
مانند یک ifیک خطی عمل میکند
.....
randomبه کار گیری از رندوم
باید بالای کد
```
import random
```
را بنویسیم و با قالب انجام دهیم
```
random_adad = random.randint(0 , 2)
```
یا
```
print(random.randint(0 , 2)
```
-> ۰ یا ۱ یا۲

برای ننوشتن رندوم
random.randint
باید بالای صفحه به جای
```
import random
```
باید بنویسیم
```
from random import randint

import random
mylist = ["apple", "banana", "cherry"]
print(random.choices(mylist, weights = [10, 1, 1], k = 14))
```

.....
lowerتابع
برای کوچک کردن حروف های بزرگ
```
print("MAMAD".lower())
```

->mamad
این تابع در اینپوت
input
هم کار می کند
name = input(what your name?).lower()
اگر اسمی مثلا
Ali
)باشد را در پرینتprint(
به
ali
تبدیل می کند
.....
تغییر بخشی از استرینگ

txt1 = "My name is {fname}, I'm {age}".format(fname = "John", age = 36)
txt2 = "My name is {0}, I'm {1}".format("John",36)
txt3 = "My name is {}, I'm {}".format("John",36)
->
My name is John, I'm 36
My name is John, I'm 36
My name is John, I'm 36

.....
list = [, ,]
.....
loop حلقه و به کار گیری ازfor
num_1 = [1,2,9]
for mamad in num_1 :
  print(mamad)
->
1
2
9
---
num_range = range (0 , 100)
for number in num_range :
  print(number)
->
1
.
.
.
99
---
name_1 = "mahdi"
for horof in name_1:
  print(horof)
->

m
a
h
d

```
i
------------------------
می توان هر اعمال ریاضی هم اعمال کنیم
num_2 = [1,2,9]
for mamad in num_2 :
   print(mamad * 2)
->
2
4
18
.....................................................................................
نمایش دادنrange وتوضیحات تکمیلی در مورد        for
myNumbers = range(1 , 10)
Result = list(myNumbers)
print(Result)
->[1 , 2 , 3 , ... , 9]
یا
print(list(range(10)))

# print(list(range(1, 10)))  # [1, 2, 3, 4, 5, 6, 7, 8, 9]
# print(list(range(10)))  # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
# print(list(range(0, 15, 2)))  # [0, 2, 4, 6, 8, 10, 12, 14]
# print(list(range(10, 0, -2)))  # [10, 8, 6, 4, 2]
-----------------------------
تکرار در پرینت به کمکfor , range

for num in range(1, 10):
   print("*" * num)
->
*
**
***
****
.
.
*********
یا
for num in range(1, 10):  # [1,2,3,4,5,6,7,8,9]
   stars = ""
   for star in range(1, num + 1):  # [1, ..., num]
      stars += "*"
   print(stars)
.....................................................................................
از آخر به اولrangeنوشتن
# print(list(range(10, 0, -2)))  # [10, 8, 6, 4, 2]
.....................................................................................
نمایس استار در حالت سینوسی

for num in range(1, 10):
   if num % 2 == 1:
      for star in range(1, 6):  # [1, 2, 3, 4, 5]
         print("*" * star)
   else:
```

```
    for star in range(5, 0, -1):  # [5, 4, 3, 2, 1]
        print("*" * star)
->
*
**
***
****
*****
*****
****
***
**
*
*
**
***
****
*****
*****
****
***
**
*
*
**
***
****
*****
*****
****
***
**
*
هر یک عدد فردی که وارد شرط میشود یک
*
**
***
****
*****
میسازد و هر عدد زوج که وارد میشود
*****
****
***
**
*
.....
while
while True:
   print("ta abad ino minevise")
تا ابد جمله داخل پرینت را می نویسد
---------------
password = input("what is your password : ")
while password != "1234":
```

```
   print("your password is wrong!!!")
   password = input("what is your password : ")
print("your password is correct !!!!")
```
.....
دو مثال با while و for
```
for number in range(1, 11):
   print(number)
```
یا
```
num = 1 #1,2,3,...,10
num =-1 #0,1,2,...,10
while num < 11:
   num += 1
   print(num)
->
0
1
2
....
10
```
.....
مثال با if و for و while braek
```
num = 1
while num < 30:
   num+=1
   print("*" * num)
   if num == 5:        تا عدد قبلی ایف می نویسد
      break
->
*
**
***
****
```
.....
مثالی دیگر با braek
```
number = 1
while number <= 10:
   print(number)
   number += 1
   if(number == 5):
      break

print("tamam")
```
.....
تابع تبدیل کننده به list
```
adad = range(1,10)
print(list(adad))
-> [1,2,3,4,5,6,7,8,9]
```
.....
نوشتن کل اعضای لیست با استفاده از for
```
thislist = ["apple", "banana", "cherry"]
for color in thislist:
   print(color)
```
در بالاتر توضیح داده شده

.....
می خواهیم در لیست فقط عبارت استرینگ نمایش
داده بشوند و عدد نمایش داده نشود
```
myColors = ["red", "blue", "green", "gray", "yellow", "orange", 3.6]
for color in myColors:
   if type(color) == str:
      print(f"the color is : {color}")
   else:
      print(f"{color} is not a color")
```
یا
```
index = 0
while index < len(myColors):
   color = myColors[index]
   if type(color) == str:
         print(f"the color is : {color}")
   else:
      print(f"{color} is not a color")

   index += 1
```
مثال برای حالت ۱
```
favorit_phone = ["iphone 11" , "iphone 10" , "sumsung" , "soni" , "marshal" , 3.3]
for favorit in favorit_phone:
   if type(favorit) == str:
      print(f"my favorit phone : {favorit}")
   else:
      print(f"{favorit} : is not a phone")
```
نکته: بعد از ایف تایپ باید فیویریت را قرار دهیم نه فیویریت فون
.....
پیدا کردن ایندکس یک ایتم از لیست
)valueبا کمک ولیو(
```
myColors = ["red", "blue", "green", "yello"]
index_Mycolors1 = myColors.index("blue")
print(index_Mycolors1)
```
->1
.....
پیدا کردن ایندکس چند ایتم تکراری از لیست
)valueبا کمک ولیو(
و نقطه شروع
```
myColors = ["red", "blue", "green", "yello","red","gray"]
index_Mycolors1 = myColors.index("red")
print(index_Mycolors1)
```
در اصل عدد شروع آن ۰ است
-> 0
```
index_Mycolors1 = myColors.index("red" , 1)
print(index_Mycolors1)
```
یعنی از اینکس ۱ به بعد = 1
-> 4
.....
پیدا کردن ایندکس یک ایتم بین دو آیتم لیست
)valueبا کمک ولیو(
و نقطه شروع و پایان
```
myColors = ["red", "blue", "green", "yello","red","gray"]
index_Mycolors1 = myColors.index("red",0,2)
print(index_Mycolors1)
```
->error ((در تست ارور نداد؟

```
index_Mycolors1 = myColors.index("red",1,5)
print(index_Mycolors1)
->4
```
.....
بدست آوردن تعداد آیتم هایی که در لیست تکرار شده است
.count()
```
myColors = ["red", "blue", "green", "yello","red","gray","red"]
redofmycolors = myColors.count("red")
->3
```
.....
وارونه کردن اعضای لیست
از آخر به اول
.reverse()
```
myNumbers = ["1","2","3","4","5","6","7","8","9"]
myNumbers.reverse()
print(myNumbers)
-> ["1","2","3","4","5","6","7","8","9"]
["9","8","7","6","5","4","3","2","1"]
```
.....
مرتب کردن لیست
.sort()
```
myNumbers = [18,4,12,35,2]
print(myNumbers)
myNumbers.sort()
print(myNumbers)
->
[18,4,12,35,2]
[2,4,12,18,35]
```
.....

.....
نوشتن اعضای لیست پشت سر هم
.join()
```
myColors = ["red", "blue", "green", "yello","gray"]
mynewlist = " -".join(myColors)
```
در داخل کوتیشن هر چیزی را می توان گذاشت
```
print(mynewlist)
->red -blue -green -yello -gray
```
.....
کپی کردن از یک آیتم تا یک آیتم دیگر لیست و ریختن آن در یک لیست جدید
[start:end:step]
```
myNumbers = [1,2,3,4,5,6,7,8,9]
new_list = myNumbers[1:5:1]
```
end را یکی بیشتر میگیرم چون یک از آخرین مانند رنج حسای نمیشود
```
print(new_list)
-> [2,3,4,5]
```
.....
کپی کردن آیتم های لیست وقتی انتها مثلا تا بینهایت است
و فاصله ۱
```
myNumbers = [1,2,3,4,5,6,7,8,9]
new_list = myNumbers[1:]
```
حتما باید با ":" باشد = 1

print(new_list)
->[2,3,4,5,6,7,8,9]

.....

کپی کردن کل لیست

myNumbers = [1,2,3,4,5,6,7,8,9]
copyofnum = myNumbers[:] [0:]
print(copyofnum)
-> [1,2,3,4,5,6,7,8,9]

:نکته

[:1] ایندکس ۰ تا ۱
[1:] ایندکس ۱ تا آخر
[:] ایندکس ۰ تا آخر
[:1:2] ایندکس ۰ تا ۱ و با الگوی یکی دوتا دوتا
[::2] ایندکس ۰ تا آخر و با فاصله دوتا دوتا
[::-1] ایندکس ۰ تا اخر و وارونه کردن عبارت

.....

انتخاب آخرین حرف یک کلمه

a = 'mahdi'

```
print(a[-1])                          ->  i
print(a[ (len(a)-1) : len(a) ])       ->  i
print(a[::-1][0])                     ->  i
```

.....

نوشتن تک تک اعضای لیست استرینگ پشت هم در یک لیست به صورت کپسلاک

myname = mahdi
name_2 = [ghasem.upper() for ghasem in myname]
print(name_2)
-> ['M', 'A', 'H', 'D', 'I']

.....

list comprehensionپیدا کردن اعداد زوج و فرد یک لیست باکمک
for - if

mynum = [1,2,3,4,5,6,7,8,9]
newlist_2k = [num for num in mynum if num % 2 == 0]
newlist_1k = [num for num in mynum if num % 2 == 1]
print(newlist_1k)
print(newlist_2k)
1k -> [1, 3, 5, 7, 9]
2k -> [2, 4, 6, 8]
#[I for i in ...]

.....

list comprehension پرینت کردن اعضای لیست با کمک
list comprehension چاپ کردن کردن اعضای لیست با کمک

[print(i) for i in list]

.....

انتخاب کردن عدد در لیست تو در تو

numbers = [1,2,3,4,5,6]
new_list = [[1,2,3],[4,5,6]]

```
print(new_list[1][2])
->6
```
.....
نوشتن اعضای یک لیست تو در تو
```
new_list = [[1,2,3],[4,5,6]]
for num in new_list:
   for number_ in num :
      print(number_)
->
1
2
3
4
5
6
```

.....
تکرار یک آیتم در لیست

```
a = ['mmd'] *3
```

a -> ['mmd','mmd','mmd']

.....
ایجاد لیست تو در تو
؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟
قسمت ۲۸
؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟
.....
پیدا کردن چیزی در استرینگ
.find()

```
s = 'my name is mahdi ebrahimi'
item = 'mahdi'
res = s.find(item)
print(res)
-> 11
```

یعنی از اینکس ۱۱ شروع میشود
برای اسلایس کردن کل
```
m = s[res  :  res+len(item)]
print(m)
-> mahdi
```

.....
لیست دیکشنری
[{},{}]
لیست دیکشنری
```
x = [{"age":"14" , "name":"mahdi"},{"age": "999999" , "name":"janati"}]
print(x)
->[{'age': '14', 'name': 'mahdi'}, {'age': '999999', 'name': 'janati'}]
```
.....
ساخت دیکشنری

```
dict()
... = {key : value , key : value}

_____________
name = dict(age="14" , name="mahdi")
print(name)
->{'age': '14', 'name': 'mahdi'}
.....................................................................
)دسترسی به تمام اجزای دیکشنریvalue(
.values()
x = {"age":"14" , "name":"mahdi"}
for value in x.values() :
   print(value)
->
14
mahdi
.....................................................................
در یافت لیستی از ولیو های دیکشنری
x = {"age":"14" , "name":"mahdi"}
print(list(x.values()))
.....................................................................
جمع کردن تمام اعضای دیکشنری
x = {'x':100, 'y':200}  #values is int
print(x.values())
print(sum(x.values()))

_____
x = {'x':'100', 'y':'200'}  #values is str
print(sum(list(map, x.values()))))    #مبحث جلو جلو
.....................................................................
نکات دیکشنری
x = {"age":"14" , "name":"mahdi"}
for value in x.values() :
   print(value)
print("_____________________________")
x = {"age":"14" , "name":"mahdi"}
print(x.values())
print("_____________________________")
name = dict(age="14" , name="mahdi")
print(name)
->{'age': '14', 'name': 'mahdi'}
print("_____________________________")
.....................................................................
چک کردن بودن ایتمی در دیکشنری
me = {"age":"14" , "name":"mahdi"}
is_eleman = "age" in me
print(is_eleman)
->True
.....................................................................
پیدا کردن کلید های دیکشنری
x = {"age":"14" , "name":"mahdi"}
for key in x.key() :
   print(key)
->
age
```

name
.....
چک کردن بودن یک کلید در دیکشنری
if:
me = {"age":"14" , "name":"mahdi"}
if "email" in me:
   print(email)
else:
   print("not found")
.....
چک کردن بودن یک کلید در دیکشنری
in
if:
isname = "mahdi" in me.value()
print("isname")
-> True
.....
پاک کردن کل دیکشنری
me.clear()
.....
کپی کردن دیکشنری
cop_me = me.copy
.....
نوشتن لیست وقتی همه ی ولیو ها یکی اند
newUser = {"name": "unknown", "family": "unknown"}
newUser_2 = dict.fromkeys(["name","family"], "unknown") به جای نوشتن dict{}می توتن نوشت .
print(newUser_2)
->{"name": "unknown", "family": "unknown"}
??
.....
دریافت ولیو با کلید
me = {"age":"14" , "name":"mahdi"}
print(me.get("age"))
->14
در این روش اگر کلیدی را وارد کنیم اما آن کلید وجود نداشته باشد
none
چاپ میشود
print(me.get("email"))
-> none
.....
راهی دیگر برای فهمیدن وجود داشتن یا نداشتن یک کلید
.get
isEmailExist = me.get("email")
print(isEmailExist is None)]
-> ture
.....
پاک کردن آیتمی از دیکشنری با استفاده از کلید
.pop()
me = {"age":"14" , "name":"mahdi"}
me.pop("name")
نکته
test = me.pop("name)
print(test)

->mahdi
.....
پاک کردن آیتمی از دیکشنری از آخر
.popitem()
me = {"age":"14" , "name":"mahdi"}
me.popitem()
print(me)
->
{"age":"14"}
.....
آپدیت دیکشنری
.update()
me = {"age":"14" , "name":"mahdi"}
second = {
  "age": 50,
  "name": "milad"
}
print(second)
second.update(me)
print(second)
{'age': 50, 'name': 'milad'}
{'age': 50, 'name': 'mohammad', 'family': 'ordookhani', 'email': 'moh96ordo@gmail.com'}
.....
آپدیت و تغییر کلید در دیکشنری

me = {"age":"14" , "name":"mahdi"}
me["name"] = mamad
->
me = {"age":"14" , "name":"mamad"}
.....
اضافه کرد یک آیتم به دیکشنری
me = {"age":"14" , "name":"mahdi"}
me["email"] = blong.org@gmail.com
->
me = {"age":"14" , "name":"mamad" , email : blong.org@gmail.com}
.....
بزرگ ترین عدد یا حرف یک عبارت را برم گرداند
min
max
min("abc") -> a
max("abc") -> c
min("123") -> 1
max("123") -> 3
.....
ساخت و تعریف تابع
def
def MMd():
  print("gohesag")
MMd()
.....
مثال برای تعریف تابع با
return
def

پارامتر
```
def hoghoge(saatkar , pol):
   koledaramad = saat kar * pol
   return koledaramad
```

آرگومان
```
print(hoghoge(  1  ,  10   ))
```
return عدد را بر میگرداند
* * نکته
```
print(hoghoge(  pol = 10  , saatkar = 1   ))
```

در این به محض اینکه به ریترن رسیدیم دف پایان خواهد یافت

اگر داخل دف از پرینت استفاده کنیم برای فراخوانی آن
```
test()
```
کافی است ولی اگر داخل آن از ریترن استفاده کنیم برای فراخوانی آن
```
print(test())
```

ما امکانی داریم که میتوان یک Return خالی هم نوشت
که آن ریترن هیچ چیزی بر نمیگرداند و فقط از تابع خارج میشود

.....
تبدیل اینپوت به لیست
```
num = str(input())
num2 = [num]
```
->['num']
```
num = str(input())
num2 = list(num)
print(num2)
```
->['n','u','m']
.....
اضافه ، اتحاد و یکی کردن دولیست به هم
```
list1 = ["a", "b" , "c"]
list2 = ["c","b","a"]
list1.extend(list2)
print(list1)
```
-> ['a', 'b', 'c', 'c', 'b', 'a']
.....
فهمیدن چند تا از آن آیتم را در لیست موجود داریم
```
fruits = [1, 4, 2, 9, 7, 8, 9, 3, 1]
x = fruits.count(9)
print(x)
```
->2
.....
ایجاد دیکشنری

```
numbers = dict(first=1, second=2, third=3)
squaredNumbers = {key: value ** 2 for key, value in numbers.items()}
print(squaredNumbers)
```
.....
list comprehension فرد یا زوج بودن عدد با کمک دیکشنری
```
simpleNumbers = {num: ("even" if num % 2 == 0 else "odd") for num in [1, 2, 3, 4, 5]}
print(simpleNumbers)
```

-> {1:"odd" , 2:"even , ...5:"odd"}

.....

نام ها ومدل های لیست و توپل و دیکشنری

```
listNum = [1,2,3,4,5]
dictNum = {1:1, 2:4 , 3:6 , 4:8 , 5:10}
tupleNum = (1,2,3,4,5)
setNum  = {1,2,3,4,5}
```

ساخت های لیست و توپل و دیکشنری

```
listNum = list(1,2,3,4,5)
dictNum = dict(a=1 , b=4 , c=6 , d=8 , e=10) #به جای حروف انگلیسی نمیتوان عدد گذاشت
```

.....

بدست آوردن ولیو یک کلید توپل

```
locations = {
   (35.67, 45.78): "Tehran",
   (40.30, 69.92): "Shiraz"
}
print(locations[(35.67, 45.78)])
```

-> Tehran

.....

به توان دو رساندن اعضای لیست

```
# numbers = [1, 2, 3, 4, 5]
# doubledNumbers = [num * 2 for num in numbers]
# print(doubledNumbers)
```

.....

برسی تعداد آیتم هایی که تکرار شده در توپل

```
print(tupleNumbers.count(3))
```

.....

تعریف متغیر شرطی

```
x = 24
y = 30

z = y if (x  > y) else x
```

z -> 24

.....

SET ها

در آن نمی توان آیتم تکراری گذاشت

به ایندکس آن دسترسی نداریم

آیتم های داخلش را سورت می کند

.....

تبدیل لیست به ست

set()

تبدیل ست به لیست

list()

نکته**

وقتی تبدیل به لیست می کنیم آیتم های تکراری حذف می شود اما

آیتم ها سورت نمی شود.

.....

اضافه کردن آیتم به ست
.add(2)

_____
حذف کردن آیتم از ست
.remove(2)

_____
حذف کردن آیتمی از ست در صورت وجود داشتن آن در ست
.discard()
نکته**
.discard(2)
در اصل همان
```
if 2 in myset:
   myset.discard(2)
```

_____
کپی کردن ست
.copy()

_____
حذف کردن تمام آیتم های درون ست
.clear()

_____
یکی کردن و اجتماع اعضای ست
print(set1 | set2) تکراری ها را حذف می کند

_____
اعضای مشترک بین دو ست
print(set1 & set2) مشترک های بین دو ست را می نویسد
.....
نوشتن تابع زود یا فرد بودن عدد با def
```
def even_num(number):
   if number % 2 != 0:

      return True
   else:
      return False
```
یا
```
def even_num(number=2):
   if number % 2 != 0:
      return True
   return False
```
مقدار پیش فرض ۲ را به نامبر داده ایم.
اگر چیزی وارد نکند ۲ وارد تابع ما می شود
.....
گرفتن مجموع بی نهایت عدد در def
*args
```
def sum_num(*number):
   total = 0
   for num in number:
      total += num
   return total
```
* * نکته
اگر ما یک لیستی را داشتیم و می خواستیم اعضای انتیجر آن را جمع کنیم
```
listN = [1,2,3,4,5,6,7,8]
print(sum_num(*listN))
```
در اصل

```
print(sum_num(listN)) # ([1,2,3,4,5,7,8],)
print(sum_num(*listN)) # (1,2,3,4,5,6,7,8,)  استخراج یک ستاره ای
```
.....
```
سازنده ی دیکشنری با def
def showUserInfo(**kwargs):
   for key, value in kwargs.items():
      print(f"{key} : {value}")
showUserInfo(name = "mahdi" , family = "ebrahimi" )    داخل یک دیکشنری
```
.....
```
def display_names(name,family):
   print(f"name is {name} and family is {family}")
* * نکته
person = {"name":"sara","family":"moradi"}
display_names(**person) # استخراج دو ستاره ای
```
.....
```
 list of range دادن لیست یک رنج
 (۱ روش Asterisks(
My_list = [*range(10, 21, 1)]
print(My_list)
-> [10,11,12,...,20]
# اگر ستاره نداشته باشد ↓↓↓
#My_list2 = [range(10, 21, 1)]
#print(My_list2)
#-> [range(10, 21)]
#print([*range(10, 21, 1)])
```
_____
```
(روش ۲
a = range(10,21)
print(list(a))
-> [10, 11, 12, ..., 20]
#print(list(range(10,21)))
```

مبحث Asterisks در جلو کامل توضیح داده شده

.....
```
وجود داشتن آیتم در لیست و ست با ۰ و ۱
num = {1,2,3,5}
isNum = 4 in num
if isNum == True:
   print(1)
else:
   print(0)
-> 0
```

.....

اعمال روی لیست ها، تاپل ها، دیکشنری ها، ست ها، لینکد لیست ها

فایل :

Part2(1).ipynb

.....

bitwise operation
^ , &&, >>, <<,

اپریتور ها روی بیت ها کار میکنند
نه روی اعدادف نه روی اینت و استرینگ

&	AND
|	OR
^	XOR

~	Complement
<<	Left Shift
>>	Right Shift

Bitwise AND

0	0	0
0	1	0
1	0	0
1	1	1

a = 20 0 0 0 1 0 1 0 0
b = 4 0 0 0 0 0 1 0 0

0 0 0 0 0 1 0 0 -> 4

a & b = 4

Bitwise OR

0	0	0
0	1	1
1	0	1
1	1	1

a = 20 0 0 0 1 0 1 0 0
b = 4 0 0 0 0 0 1 0 0

0 0 0 1 0 1 0 0 -> 20

a | b = 20

Bitwise XOR

fir	sec	res
0	0	0
0	1	1
1	0	1
1	1	0

a = 20 0 0 0 1 0 1 0 0
b = 4 0 0 0 0 0 1 0 0

_____
0 0 0 1 0 0 0 0 -> 16

a ^ b = 16

Bitwise Complement

0	1
0	1
1	0
1	0

x = 20 0 0 0 1 0 1 0 0

_____
1 1 1 0 1 0 1 1 -> -21

~x = -(x)-1
~x = -21

Bitwise Left Shift

a = 20 0 0 0 1 0 1 0 0
b = 2 0 1 0 1 0 0

_____
0 1 0 1 0 0 -> 20

??

.....
کردن دو رشته متنی XOR

```
s1 = "hello"
s2 = "12345"
a_list = [chr(ord(a) ^ ord(b)) for a,b in zip(s1, s2)]
# print(a_list)
s3 = "".join(a_list)
print(s3)
```

point

در این قطعه کد باید سایز یا len دو رشته متنی s1, s2 با هم برابر باشد

در ضمن خروجی s3 هر چیزی بود
اگر با s2 دوباره XORبگیریم، حاصل برابر با s1 میشود

حالا میتوانیم s1 رو مثلا یک متنی که میخواهیم رمزنگاری شود بدانیم
s2 رو پسوورد
و s3 متن رمز شده

.....
تعریف فانکشن ساده در یک خط
def squar(num):
  return num**num
یا
def sum(first,secend): return first + secend
یا
myfunction = lambda num : num * num
**نکته
اگر درlambda
از کلید میانبرalt shift f
استفاده کنیم اتوماتیک لمبدا را به یکdef
تبدیل می کند به این صورت:
def myfunction(num) : return num * num
و برای فراخوانی لامبدا مانند تابع معمولی است
print(myfunction(2))
-> 4
.....
**نکته
lambda
فقط برای تعریف کردن تابع یک خطی است
.....
پیدا کردن اسم یک فانکشن
print(myfunction.__name__) -> myfunction
print(mylambdaFunc.__name__) -> <lambda>
.....
دوبرابر کردن اعضای لیست به کمکlambda & map
map -> دو ورودی میگیرد ، یکی فانکشن و یکی ایتربل
ایتربل یک چیزی است مثل لیست ، استرینگ ، توپل و غیره
mynum = [1,2,3,4,5]
doubles = map(lambda x : x*2 , mynum) -> استmynum های x نماینده ی تک تک آیتم
-> یه چیز عجق وجق می ده که باید به لیست تبدیل کرد
doubles = list(map(lambda x : x*2 , mynum))
**نکته

یکی از قبلیت های مپmap
این است که فقط یک بار می توان آن را پرینت کرد
مثلا اگر همانdoubles
بالا را یک بار
print(doubles)
print(doubles)
: کنیم
[2,4,6,8,10]
[]
.....
تبدیل یک لیست با اعداد استریک به اینتیجر
اعضای استرینگ لیست به اینت
str list item to int
x = ['1', '2', '3', '4']
print(type(x[0])) #--> str
print(list(map(int, x)))
print(type(list(map(int, x))[0])) #--> int

point
دو ورودی میگیرد که اولیشmap همانطور که گفتیم
فانکشن است و دومیش ایتربل
استفاده میکنیمint حالا ما به جای فانکشن خودمون از
که میاد اینت رو روی تمام اعضا در نظر میگیرد
.....
بزرگ کردن حروف اعضای یک لیست با کمکmap & lambda upper
names = ['ali','mohammad','mahdi']
upper = map(lambda name : name.upper(),names)
.....
در آوردن فامیلی های یک دیکشنری درون یک لیست
map
lambda
people = [
  {'name': 'mohammad', 'family': 'ordookhani', 'age': 23},
  {'name': 'mahdi', 'family': 'ebrahimi', 'age': 14},
  {'name': 'iman', 'family': 'madaeny', 'age': 30}
]
families = map(lambda person: person['family'], people)
print(families)
دردر سورتد باید بهش یک کلید دهیم اما اینجا لازم نیست.key
نکته**
برای خروجی گرفتن مپ که در اینجا فمیلیز هست باید
یا در همان متغیر فمیلیز و یا در پرینت آن را تبدیل به
لیست کرد

_____
نوشتن یک خطی ->
print(list(map(lambda person: person['family'], people)))

_____
نوشتن با حلقه فورfor loop<-
for person in people:
  families_2.append(person['family'])

_____
list comprehension <- نوشتن با
print([i['family'] for i in people])

.....
جدا کردن ولیو های یک دیکشنری با کمک list comprehension
dictionary = {20: 16668, 30: 16795, 40: 16755, 50: 16582, 60: 16463, 70: 16737}
inverse = [(value, key) for key, value in totals.items()]
-> [(16668, 20), (16795, 30), (16755, 40), (16582, 50), (16463, 60), (16737, 70)]
.....
دسترسی به کلید ها و ولیو های دیکشتری با کمک list comprehension
dictionary = {'mmd' : 20, 'kazem' : 10, 'hsn' : 14}
a = [val for key,val in total.items()]
print(a)
-> [20, 10, 14]
b = [key for key,val in total.items()]
print(b)
-> ['mmd', 'kazem', 'hsn']
.....
فهمیدن بودن آیتم در دیکشنری با کمک list comprehension
dictionary = {'mmd' : 20, 'kazem' : 10, 'hsn' : 14}
a = {key: ('khode mamad' if key == 'mmd' else 'mamad ni') for key,val in dictionary.items()}
print(a)
-> {'mmd': 'khode mamad', 'kazem': 'mamad ni', 'hsn': 'mamad ni'}

مثل همچین چیزی را در
فرد یا زوج بودن عدد با کمک دیکشنری''
داشتیم
.....
پیدا کردن بزرگ ترین ولیو در دیکشنری
total = {20: 16668, 30: 16795, 40: 16755, 50: 16582, 60: 16463, 70: 16737}
inverse = [(value, key) for key, value in totals.items()]
-> [(16668, 20), (16795, 30), (16755, 40), (16582, 50), (16463, 60), (16737, 70)]
print(max(inverse))
-> (16795, 30)
point
1627
برای این اینورس میکنیم چون در تاپل ها
a = [(4,2), (3,1000), (1,4), (1,5)]
max(a)
-> (4,2)
به اولین آیتم نگاه میکنید بعد میرود آیتم دوم
اگر برعکس نکنیم بزرگ ترین سن رو به ما میده که در اصل بزرگ ترین
کلید رو میده ولی ما بزرگ ترین مقدار یا ولیو رو میخوایم
.....
فیلتر کردن . بدست آوردن اعداد زوج یک لیست
map
lambda
fiter -> مانند مپ
مپ میاد و روی همه ی آیتم ها عملیات رو انجام میده و همرو بر میگردونه
ولی فیلتر اونایی که درست باشه رو برمیگردونه

numbers = [1,2,3,4,5]
even = filter(lambda num : num % 2 == 0 , numbers)
-> [2,4]
even = map(lambda num : num % 2 == 0 , numbers)
->[False, True, False, True, False]

.....
filter
numbers = list(range(1,6))
def even_or_odd(num):
   return num % 2 == 0
filter(list(filter(even_or_odd, numbers)))
-> [2, 4]
.....
higher order function چیست
function(argument)
وقتی میایم در ارگیومنت تایع از یک فانکشن دیگه استفاده کنیم
مثلا درfilter
.....
نشان دادن اسم افرادی که سبد خرید آنها خالی است
users = [
   {'name': 'mohammad', 'shopCart': []},
   {'name': 'ahmad', 'shopCart': ['tof','mags kosh']},
   {'name': 'iman', 'shopCart': []}
]

result = filter(lambda user: len(user['shopCart']) == 0, users)
**نکته
اگر در بالا
len
را ننویسیم خودش خود به خود حالت های ترو (درست) آن را
چک می کند و می گوید یعنی
result = filter(lambda user: user['shopCart'], users)
سارا را نمایش می دهد چون او فقط در سبد خریدش بیشتر از
صفر آیتم دارد ولی اگر بخواهیم آنهایی که ۰ آیتم دارند را
نمایش دهیم باید قبل از آن از
not
استفاده کنیم
result = filter(lambda user: not user['shopCart'], users)
-> این حالت تمام مشخصات را به ما میدهد
اما ما فقط نام او را میخواهیم پس از مپ و فیلتر باهم
استفاده میکنیم
result2 = map(lambda user: user['name'],
   filter(lambda user: not user['shopCart'], users)
)
و این کار را میتوان با لیست کامپرنشن نیز انجام داد
result3 = [user['name'] for user in users if len(user['shopCart']) == 0]
.....
"all" function
اگر اعضای داخلی آن ترو بود ترو را برمی گرداند و اگر
نبود فالس را بر میگرداند
print(all[0]) → False
print(all[1]) → True
print(all[]) → True
.....
میتوان لیست کامپرنشن را به این گونه نیز نوشن
باقی مانده بر ۲
print(all([num for num in numbers if num % 2 == 0]))

print(all([num % 2 == 0 for num in numbers]))
.....
فهمیدن زوج یا فرد بودن یک لیست با بولیان
all
print(all([num % 2 == 0 for num in numbers]))
→ [True,True,True,True]
.....
"any" function
اگر فقط داخل یک ترو باشد ترو برمیگرداند
print(all[0]) → False
print(all[1]) → True
print(all[0,0,0,0,1]) → True
print(all[]) → False
.....
فهمیدن زوج یا فرد بودن یک لیست با بولیان
any
numbers = [2, 4, 6, 8]
print(any([num % 2 != 0 for num in numbers]))
→ False
numbers = [2, 4, 6, 7]
→ True
.....
تغییر بخشی از استرینگ
name=input('Enter your name : ') ← mahdi
print("Hi %s Let us be friends!" % name);
→ Hi mahdi Let us be friends!
print("Hi %r Let us be friends!" % name);
Hi 'mahdi' Let us be friends!
print("{'name':'%s','family':'%s'}" % ('mahdi','ebrahimi'))
{'name':'mahdi','family':'ebrahimi'}

num = 3
print('this number is %i and it int' %num)
->this number is 3 and it int
num = 3
print('this number is %5i and it int' %num)
->this number is   3 and it int
num = 3
print('this number is %05i and it int' %num)
->this number is 000003 and it int

.....
صبر کردن در خروجی پایتون
library time
time.sleep(ms)
import time
time.sleep(2.0)
print("Hi after 2s")
.....
ایمپورت کردن تمام تابع های یک کتابخانه
from random import *
.....

سورت کردن از آخر به اول سن اعضای یک دیکشنری
sorted

```
users = [
   {'name': 'mohammad', 'family': 'ordookhani', 'age': 23},
   {'name': 'taha', 'family': 'ordookhani', 'age': 40},
   {'name': 'ali', 'family': 'ordookhani', 'age': 30},
   {'name': 'sara', 'family': 'ordookhani', 'age': 80}
]

print(sorted(users, key=lambda user: user['age'], reverse=True))
```
→
```
[{'name': 'sara', 'family': 'ordookhani', 'age': 80},
{'name': 'taha', 'family': 'ordookhani', 'age': 40},
{'name': 'ali', 'family': 'ordookhani', 'age': 30},
{'name': 'mohammad', 'family': 'ordookhani', 'age': 23}]
```

نکته**
reverse = True
نتیجه را از آخر به اول تبدیل می کند
نکته**
key=
lambdaاما در اصل با کمک
کلید اصلی دیکشنری را به او معرفی میکنیم
"nameمثلا کلید اصلی "
و حامل باید به صورت
key =
تعریف کنیم نمیتوان
moz =
تعریف کرد
.....
sort & sortedفرق
.sort = خود لیست را عوض می کند
sorted() = به خود لیست یا... اصلی کاری ندارد
```
num = [2,4,5,1,3]
num.sort
print(num)
```
→ [1,2,3,4,5]

_____
```
num = [2,4,5,1,3]
result = sorted(num,reverse = True)
print(num)
```
→[2,4,5,1,3]
```
print(result)
```
→[5,4,3,2,1]
.....
فهمیدن طولانی ترین اسم یک لیست
```
names = ['mohammad', 'milad','iman','ali','zahra','ahmad']
res = [len(name) for name in names]
print(res)
```
→[8,5,4,3,5,5]
```
print(max(names))
```
→ zahra

```
print(min(numbers))
→ ahmad

print(max(names, key=lambda n: len(n)))
→ mohammad  بیشترین کاراکتر

print(min(names, key=lambda n: len(n)))
→ ali    کمترین کاراکتر
```
.....
. فرقreverse & reversed()
.reverse = لیست اصلی را تغییر میدهد
reversed()= با لیست اصلی کاری ندارد
نکته**
reversedوقتی
را پرینت می کنیم یک چیز چرت و پرت میدهد
که باید آن را تبدی به لیست کرد
mapمانند
.....
بر عکس کردن یک عبارت از دوروش
اسلایسینک
reversed
```
print("hello"[::-1])
→ olloh
print((list(reversed("hello"))))
['o','l','l','o','h']
print("".join(list(reversed("hello"))))
→ olloh
```
.....
lenمیتوان لن را به این صورت نیز نوشت
```
print(list1.__len__())
```
.....
قدر مطلق عدد با دو روش
abs

روش اول

```
a = -5
print(abs(a))
→ 5
a = 5
print(abs(a))
→ 5
```

روش دوم

```
a = -5
if a>0 :
   print(a)
if a<0:
   print(a*-1)
```

.....

عملگر های یگانه/ unary operator

- x	- (x)
+ x	+(x)
~ x	BITWISE NOT

print(+ - - -1) → -1

.....
فهمیدن مجموع اعداد درون یک لیست
num = [1,2,3,4,5]
print(sum(num))
→ 15
num = [1,2,3,4,5]
print(sum(num,5))
→ 20 می کند 5عدد نهایی مجموع را به علاوه ی
.....
roundروند کردن عدد با کمک
num = 4.4
print(round(num))
→ 4
num = 4.5
print(round(num))
→ 4
num = 4.6
print(round(num))
→ 5

num = 4.6783
print(round(num,2))
→ 4.68
.....
zip object چیست
num1 = [1,2,3]
num2 = [5,6,7]
result = zip(num1,num2)
print(dict(result))
→{1:5,2:6,3:7}
result = zip(num2,num1)
→{5:1,6:2,7:3}
**نکته
اگر تعداد اعضای دو لیست برابر نباشد تا
آنجایی که مشترک دارند پیش می رود
num1 = [1,2,3]
num2 = [5,6,7,8,9,10]
result = zip(num1,num2)
print(dict(result))
→{1:5,2:6,3:7}
**نکته
زیپ مانند مپ فقط یکبار قابلیت اجرا دارد
بار دوم یک لیست یا یک دیکشنری خالی تحویل می دهد
.....

unzip کردن
myList = [(1, 5), (3, 7), (6, 4), (7, 9)]
print(list(zip(*myList)))
→[(1, 3, 6, 7), (5, 7, 4, 9)]
point
اون ستاره ای که گذاشتیم را کامل توضیح داده ایم
Asterisks
.....
بدست آوردن بهترین نمره بین دو نمره ی یک دانش
آموز همراه با اسم در قالب دیکشنری
zip()
students = ["mohammad", "iman", "sara"]
midterm = [78, 80, 94]
final = [90, 88, 92]

finalGrades = {t[0]: max(t[1], t[2]) for t in zip(students, midterm, final)}
print(finalGrades)
→ {'mohammad': 90, 'iman': 88, 'sara': 94}
.....
بدست آوردن مینگین نمره بین دو نمره ی یک دانش
آموز همراه با اسم در قالب دیکشنری
lambda
map
students = ["mohammad", "iman", "sara"]
midterm = [78, 80, 94]
final = [90, 88, 92]
average = zip(
  students,
  map(
    lambda pair: (pair[0] + pair[1]) / 2,
    zip(midterm, final)
)
)
print(dict(average))
→ {'mohammad': 84.0, 'iman': 84.0, 'sara': 93.0}
#print(dict(zip(students ,map(lambda i: (i[0]+ i[1]) /2 ,zip(midterm, final)))))
.....
ایجاد ارور ساختگی
raise
raise ValueError("invalid value")
هر اروری با هر متنی
.....
نکته
if type(x) is not str:
  .....
if input1 not in number:
  .....
.....
در صورت اینکه انجام نشد ارور بدهد:
try:
except:
finally:
try:

```
   print(x)
except:
   print("x is not difind")
finally:
   print('try except is ended')
->
x is not difind
try except is ended

try:
   print(x)
except KeyError:
   return KeyErrot : x is not difind
except ValueError as err:
   return err
   return ValueError : x is not difind
else:
   return unknown
finallu:
   return all time
```

اگر کی ارور اتفاق افتاد آن را گرینت کن و اگر ولیو ارور اتفاق افتاد
آن یکی را پرینت کن . همچنین هر چند تا را می تاونیم لگذاریم
الس اگر اکسپت انجام نشد انجام میشود و فاینالی همیشه
تحت هر شرایطی انجام میشود.
err : ارور خود سیستم

.....

.....

.....

python debugger

```
pdb
import pdb
pdb.set_trace()
```

l → line list
n → next
c → end debugging and continue program
p → print
میتوان به صورت زیر هم نوشت
p l پرینت کن متغیر ال را
نه لاین لیست

.....

استفاده از کتابخانه در یک خط

```
def sum(a,b,c,d)
   import pdb; pdb.set_trace()
   return a + b + c + d
```

.....

دوره ی پیشرفته

شئ گرایی
رفتار و خصوصیت

```
class User:
   userName = ""
   userFamily = ""
   def __init__(self,userName, userFamily):
      self.userName = userName
      self.userFamily = userFamily
   def ShowFullName(self):
      return f"{self.userName}  {self.userFamily}"
```

point

1) وقتی فانکشن را فراخوانی می کنیم = __init__
اولین چیزی که اجرا می شود و روی آن اعمال می شود
اینیشالایزر

2) برای هر فانکشنی که تعریف می کنیم باید یک
self
درون پرانتز بذاریم . این سلف به خود متغیری که وارد کلاس می شود اشاره داره

نام گذاری های خاص پایتون

1) _neme
2) __neme Name mangling
3) __name__

معمولا در کلاس ها به کار میرود

1- این نامگذاری وقتی در متغیری استفاده می شود
مخاطب برای دسترسی به آن متغیر باید اسم کامل آن را
وارد کند یعنی این متغیر پرایوت می شود و برای دسترسی
به آن با زدن
n
کل name
را نمایش نمیدهد
این نوع نام گذاری به صورت قرار دادی برای داخل خود کلاس
به کار میرود

2- اگر در متغیر از این نوع استفاده شود برای
دسترسی به متغیر باید

```
class User:
   def __init__(self,username):
      self.__name = username
x = User('MahdiEbi')
x._User__name
```

کاربرد این نوع در وراثت ها است که مثلا اگر دو
کلاس یک فانکشن به نام name
یکسان داشتند این دو قاطی نشوند

---

point
به جای
self
می توان از هر چیز دیگری استفاده کرد

point
وقتی متغیری را در کلاس و خارج از تابع می نویسیم
برای دسترسی به آن باید:
className.variable
مثال : User.activeUsers

تغییر متغیر در کلاس ها مهم

```
def User():
   activeUsers = ['MahdiEbi','MmdEbi']
   def __init__(self, userName):
      self.name = userName
me = User("MahdiEbi")
User.activeUsers = ['Kazam']
print(me.activeUsers) -> ['MahdiEbi','MmdEbi']
print(User.activeUsers) -> ['MahdiEbi','MmdEbi']

_____
me = User("MahdiEbi")
me.activeUsers = ['Kazam']
print(me.activeUsers) -> ['kazam']
print(User.activeUsers) -> ['MahdiEbi','MmdEbi']
```

اگر دست به
برنیم متغیر های وابسته هم تغییر می کنند
اما اگر دست به متغیر های وابسته بزنیم
تغییرات روی متغیر کلاس اصلی اعمال نمیشود

برگرداندن فانکنشن درون کلاس به خود کلاس نه مدل شبیه ساز شده ی آن

```
decorator
@classmethod
class User:
   activeUsers = 0

   def __init__(self, name, family):
      self.name = name
      self.family = family
   @classmethod
   def getActiveUsers(clss):
      return clss.activeUsers
me = User("Ali","Arjmandi") -> در اینجا سلف به این متغیر بر میگردد
#اما در کلاس متودآیتم اول به خود کلاس بر میگردد
در این تابع دیگر نیازی به نوشتن سلف نداریم و در عوض
برای آیتم اول باید یک حرف دیگر دلخواه بنویسیم
که این به خود کلاس بر میگردد
```

ساخت نمونه با استفاده از کلاس متود

```
class User:
   activeUsers = 0
   def __init__(self, name, family):
      self.name = name
      self.family = family
      User.activeUsers += 1
   def getActiveUsers(self):
     return User.activeUsers
   @classmethod
   def newUser(cls, string):
      # data = string.split(",")
      # print(data)
      # return cls(data[0],data[1])
      name, family = string.split(",")
      return cls(name,family)
```

me = User.newUser('mmd,ebi')
print(me.name, me.family)
print(User.newUser('mahdi,ebi').name)
print(User.activeUsers)
->
mmd ebi
mahdi
2
newInstance = User.string("Mahdi","Ebi")
print(newInstance.name , newInstance.family)
point
cls
یا هر اسم دلخواه بر میگردد به خود کلاس
User
حتی میتوان از اسم خود کلاس که همان یوزر هست هم
clsبه جای
استفاده کرد

تغییر دادن رفتار نمایش کلاس به جای آدرس حافظه به متغیر دلخواه
__repr__

```
class User:
   activeUsers = 0

   def __init__(self, name, family):
      self.name = name
      self.family = family
   def __repr__(self):
      return f"{self.name} {self.family}"
```

me = User("mohammad","ordookhani") -> در اینجا به خود کلاس اشاره داریم
print(me) -> mohammad ordookhani -> خود کلاس را پرینت میکنیم
-> اگر از رپر استفاده نمی کردیم آدرس حافظه را نشان میداد
در اصل رپر وقتی خود کلاس پرینت بخواد بشه فعال میشه

تنظیم خروجی پرینت گرفتن از instance

```
class MyClass:
    x = 0
    y = ""

    def __init__(self, anyNumber, anyString):
        self.x = anyNumber
        self.y = anyString

    def __str__ (self):
        return f"My Class x = {str(self.x)} , y= {self.y}"

myObject = MyClass(12345, "Hello")
print(myObject.__str__())
print(myObject)
print(str(myObject))
print(myObject.__repr__())
```

```
Out>
My Class x = 12345 , y= Hello
My Class x = 12345 , y= Hello
My Class x = 12345 , y= Hello
<__main__.MyClass object at 0x000002065E718BB0>
```

تابع مستقل در پایتون

```
staticmethod
class user:
   def __init__(self,name,family):
      self.name = name
      self.family = family

   def get_name(self):
      print(self.name + self.family)
   @staticmethod
   def sum(x,y):
      print(x+y)
me = user('mahdi','ebi')
me.get_name()
me.sum(1,2)
```

در استاتیک متد دیگر نیازی به سلف نیست

نکته ای در مورد init

داخل initفقط میتوانیم از print استفاده کنیم
یعنی نباید از return اسفاده کنیم

```
class Main:
   def __init__(self, name, family):
      return name + family
main = Main('mahdi','ebrahimi')
```

این ارور میدهد، پس نباید از این روش استفاده کنیم

```
class Main:
   def __init__(self, name, family):
      print(name + family)
main = Main('mahdi','ebrahimi')
```

دلیل ??

کلاس ها تو در تو

```
class A:
  def __init__(self):
     print("first")
  class B:
     def __init__(self):
        print("secend")
     def m1(self):
```

```
        print("third")
a=A()
b=a.B()
b.m1()
->
first
secend
third
```

?????????

```
import gc
print(gc.isenabled())
gc.disable()
print(gc.isenabled())
gc.enable()
print(gc.isenabled())
```

ارث بریparent

```
class User:
   activeUsers = 0

   def __init__(self, name, family):
      self.name = name
      self.family = family
class Person(User):
   pass
me = User("Mahdi","Ebi")
you = Person("Ali","behjati")
```

point

این موضوع در مورد کلاس متود های دو کلاس هم صدق میکند

---

point

میخواهیم اگر سن کمتر یا
مساوی با صفر بود
سن را مساوی با ۰ قرار دهد

```
class User:
   def __init__(self, name, family, age):
      self.name = name
      self.family = family
      if age >= 0:
         self.age = age
      else:
         self.age = 0
me = User("mahdi", "Ebrahimi", -23)
print(User.age)
-> 0
me.age = -10
print(me.age)
-> -10
```

اما در این صورت باز هم سن منفی ده است پس اشکال دارد
راه حل این مشکل دو راه است
1) یک فانکشن تعریف کنیم که سن را بگوید

2) property

برداشتن پرانتز باز و بسته در فراخانی فانکشن ها
decorator
@property

```
class User:
   def __init__(self, name, family, age):
      self.name = name
      self.family = family
      if age >= 0:
         self._age = age
      else:
         self._age = 0
   @property
   def age(self):
      return self._age
```

کار پراپرتی این است که
me.age()
را به
me.age
تبدیل می کند
در اصل ما میخواهیم سن را داشته باشیم
و ما نمیخواهیم مانند فانکشن تهش پرانتز باز و بسته باشد
اما اگر ما بخواهیم سن را عوض کنیم به یک ستر نیاز داریم که در پایین میگوییم

با توجه به کد های بالا .ایجاد ستر برای پراپرتی
property
setter
.....

```
   @property
   def age(self):
      return self._age

   @age.setter #هر اسمی نمیتوان
   def age(self, value):6
      if value >= 0:
         self._age = value
      else:
         raise ValueError("age can not be negative! ")

me = User("Mahdi","Ebrahimi", 14)
me.age = 15
```

آوردن اینیشالایزر پرنت در فانکشن فرزند با یک خط
super()

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

class person:
```

```
    def __init__(self, name, family, email):
        ## self.name = name
        ## self.family = family
        # person.__init__(self, name, family)
        super().__init__(name, family)
        self.email = email
```

نکاتی در مورد ارث بری چندگانه
1) در ارث بری چند گانه بهتر است از سوپر استفاده نکرد و از
cls.__init__(self,name,family,...)
چون سوپر اینیت یک پرنت را اجرا میکند
2) ترتیب نام دهی پرنت مهم است چون ترتیب اجرا نیز است یعنی
class clsA(clsB,clsC):
اگر مثلا ما دنبال نام باشیم اول در کلاس بی جست و جو می شود و اگر بود
که تمام اما اگر نداشت در کلاس سی دنبال آن میگردد
3) برای فهمیدن اولویت و ترتیب اجرای کلاس ها میتوانیم از
print(cls.__mro__)
print(cls.mro())
help(cls)
4) برای اینکه متغیر هایی مه داریم در کلاس های مختلف باهم قاطی نشود بهتر است
از نیم منگلینگ اسفاده بکنیم
self.__name -use-> _cls__name

مفهومpolymorphism
چند ریختی
poly -> multi
morph -> form

```
x = [1,2,3]
y = [4,5,6]
len(x)
len(y)
```

یک پولیمورفیسم هست چون برای هر متغیر یک کار ثابتی را میکند len

```
class IAnimal:
    def sound(self): raise NotImplementedError
    def move(self) : raise NotImplementedError
class Dog(IAnimal):
    def sound(self):
        print("hup hup")
class Cat(IAnimal):
    #def sound(self):
    #    print("mio mio")
    pass
```

حالا اگر برنامه را اجرا کنیم سگ به درستی اجرا می شود ولی گربه به
ارور میخورد و کاربردش این است که ما میفهمیم کدام را تعریف کرده ایم
این کار ما را موظف می کند که برای گربه صدا را پیاده سازی کنیم
در IAnimal
نشان دهنده ی اینترفیس یا قرار داد است I
که ما در اینحور موارد بهتر است ازش استفاده کنیم

یعنی نمونه سازی Instance
admin = Admin() -> یک نمونه است

---

point
lenبرای ابجکت هایی اجرا می شود که
__ lenرا داشته باشند __
نمونه هایی دیگری نیز داریم مثلا
+ / __add__
* / __mul__
/ / __Truediv__
iter / __iter__

---

iteratorsمفهوم پیمایش
next()
iter()

آیتم هایی که میتوان روی آیتم هایش پیمایش کرد <- iterable
اما هنوز داخل حلقه نرفته و خام است
در اصل آن لیست ایتربل است اما داخل حلقه به ایتریتور تبدیل می شود
یا ابجکتی که میتواند به ایتریتور تبدیل شود
آیتم هایی که میتوان روی آیتم های آن در حلقه فور پیمایش کرد <- iterator
عملیات پیمایش / تک تک پیمایش <- iterate
مانند لیست ها توپل ها و غیره
داخل فور
iterable -> iter() -> iterator
iterator -> next() -> next item

ساخت حلقه فور با کمک
while
iter() -> میشکند
next()

```
a = "Mahdi"
iterName = iter(a)  -> ['M', 'a', 'h', 'd', 'i']
while True:
   try:
      print(next(iterName))
   except:
      break
```

جلوی اکسپت میتوان
StopIteration
هم گذاشت چون اروری است که وقتی اعضا تمام میشود
و ما به بعدی میرویم میبینیم

در این کار می توانیم خیلی مانور دهیم مثلا در
exceptمیتوانیم کار های دیگری انجام دهیم و در
tryهم همین طور

ساخت custom for
اعمال کار های فانکشن روی اعضا

```
def for2(iterable, func):
   iterator = iter(iterable)
   while True:
      try:
```

```
        obj = next(iterator)
      except StopIteration:
        break
      else:
        func(obj)
number = [1, 2, 3, 4, 5, 6]
def squre(num):
  print(num ** 2)
for2(number, squre)
->
1
4
9
16
25
36
```

پیاده سازی کلاس range()

```
class Counter:
  def __init__(self, start, end, step=1):
    self.current = start
    self.end = end
    self.step = step
  def __iter__(self):
    return self
  def __next__(self):
    if self.current < self.end:
      num = self.current
      self.current += self.step
      return num
    raise StopIteration

myCounter = Counter(10, 20)
for num in myCounter:
  print(num)
```

دلیل این اتفاق این است که حلقه فور به ارور استاپ ایتریشن حساس
است و وقتی این ارور را ببیند استوپ میشود و دلیل استفاده نکردن
از حلقه بینهایت این بود که
next داخل فور تا ابد ادامه پیدا میکند
آیا فقط در فور به این شکل هست ؟ /؟

مفهوم generator

```
generator = iterator    iterator != generator
```

پس جنریتور مستقیم یک ایتریتور است و نیازی به
__iter__ کردن ندارد

```
generator function :
def count_up_to(max):
  count = 1
  while count < max:
    yield count
    count += 1
```

فرق جنریتور فانکشن این است که برای بیگ دیتا بسیار مناسب است

و در هر لحظه یک جواب را می گوید و مانند فانکشن های معمولی نیست که کل جواب
را بعد از محاسبه . یکی دیگر از فرق ها این است که مقدار قبلی را در خود
ذخیره میکند مثلا بار اول تا خط ۲۳۰۴ می آید و دفعه بعد از ۲۳۰۵ بعد از
ییلد شروع میکند
yield
مانند
return
است و فرقش این است که بعد از آن هم فانکشن اجرا میشود
و برنامه برک نمیخورد

ایجاد جنریتور در یک خط
myGenerator = (num for num in range(20)) -> generator object
print(next(myGenerator)) -> 0
print(next(myGenerator)) -> 1
print(next(myGenerator)) -> 2

همان کاری را که در لیست کامپرنشن انجام میدادیم در این
انجام می دهیم فقط با پرانتز به جای کروشه

مقایسه جمع اعداد تا یک رقم در یک خط بین جنریتور و لیست کامپرنشن
print(sum([num for num in range(100000000000000)]))
-> MemoryError
print(sum(num for num in range(100000000000000)))
-> after secend print answer
به این دلیل جنریتور در بیگ دیتا استفاده میشود
چون محدودیت مموری ندارد ولی ممکن است طول بکشد

---

point
میتوان فانکشن های زیر هم و تو درتو نیز استفاده کرد
```
def:
   def:
   def:
   def:
      def:
```

---

decorator / دکوریتور ها
```
def My_decorator(func):
   def say_bye():  -> ما به این فانکشن بسته بندی کننده یا Wrapper میگوییم
      print("bye guys")
      func()
   return say_bye
@My_decorator
def Say_hi():
   print("hi guys")
#say = My_decorator(Say_hi)
#sayHelloByDecorator()
->
bye guys
hello guys
```
به جای مدل سازی میتوان از
@funcName استفاده کرد
point
اگر فانکشن مفعول ما

Say_hi
بخواهد آرگومان دریافت کند باید هر آرگومانی برایش میذاریم برای
def say_bye() / func()
بذاریم
اما اگر تعداد فاتکشن های مفعولیمان زیاد شود ما نمیتوانیم
برای همه ی آنها را تنظیم کنیم مثلا یکی دو آرگومان نام و نام خانوادگی دارد
و یکی ممکن است ایمیل هم داشته باشد و اگر تعداد خیلی زیاد شد
نمیتوانیم برای همه ی آنها در دکوریتور تعریف کنیم در نتیجه
def say_bye(*args,**kwargs) / func(*args, **kwargs)

>_
>_

من بخش اصلی پایتون رو بعد حدود ۱۰ ماه یاد گرفتم 99/7/12
که در اصل الان من توی سیزدهم هستم و از ۱۲ شب گذشته
prsa pshgoie / nma suki -> birthday pizza in my guest

>_
>_

نام ها و قطع وابستگی اسمی

```
def My_decorator(func):
   def Wrapper_func():
      print(func.__name__)
      func()
   return Wrapper_func
@My_decorator
def Say_hi():
   print("hi guys")
-> func.__name__ : Wrapper_func
```

اما باید Say_hi باشد

```
from functools import wraps
def My_decorator(func):
   @wraps(func)
   def Wrapper_func():
      print(func.__name__)
      func()
   return Wrapper_func
@My_decorator
def Say_hi():
   print("hi guys")
-> func.__name__ : Say_hi
```

الان دیگه اسم به خود فانکشن بر می گردد

مقدار دهی به دکوریتور با مقدار بولیان

```
def show_decorator(is_show):
   def inner_decorator(func):
      @wraps(func)
      def wrapper():
         if is_show:
```

```
            func()
        else:
            return "you dont have permission"
    return wrapper
  return inner_decorator

            True
@show_decorator(False)
def adminPage():
  print("you are in admin page")

adminPage()
T -> you are in admin page
F -> you dont have permission
```

گرفتن مقدار از دکوریتور و یک مثال:
اگر تعداد اعضای نام از عدد داده شده بیشتر باشد اررو دهد

```
from functools import wraps
def check_string_length(characterCount):
  def inner_decorator(func):
    @wraps(func)
    def wrapper(name):
      if len(name) > characterCount:
        print("an error occured")
      else:
        func(name)
      return func
    return wrapper
  return inner_decorator

@check_string_length(5)
def show_name(name):
  print(name)
show_name("Mahdi") -> Mahdi
show_name("mohammad Kazam") -> an error occured
```

<- decorator factoryبه فانکشن پرنت که یک ورودی میگیرد

IO fileخواندن فایل های متنی
open
read()
txt -> "test.txt"
hello
python is here

_____
py ->

```
text = open("test.txt")
  print(text.read())
```

point
Curcer -> خط چشمک زنی است که موقع تایپ کردن میبینیم
موقع خواندن فایل متنی این خط از اول تا آخر متن را
ایندکس به ایندکس طی میکند به این دلیل ما نمیتوانیم یک
فایل متنی را دوبار بخوانیم چون حرفی جلوی آن نیست

مگر اینکه کرسر را به ایندکس صفر بیاوریم این کار را با
text.seek(index)
به غیر از خواندن حرف به حرف میتوانیم خط به خط
هم فایل را بخوانیم
text.readline()
ولی باز هم مشکل بالا را داریم ولی
با این فرق که کرسر از اول خط به آخر خط میآید
و ما باید کل خطوط را دریافت کنیم
و کل خطوط را یکجا بخوانیم (\n)
text.readlines()

بستن فایل به دلیل استفاده از رم
اگر ما یک فایلی را میخوانیم یک کانکشن بین فایل پایتون
با فایل متنی برقرار میشود که این باعث مصرف شدن رم
میشود که ما میتوانیم بعد از عملیات هایمان
این کانکشن را با دستور زیر ببندیم
text.close()

باز کردن و بستن فایل در یک دستور
```
with open("text.txt") as File:
   print(File.read())
   #File.seek(0)
   #print(File.read())
```

تغییر مود فایل متنی
به صورت پیشفرض مود روی
r -> read
فقط خواندن است
mode 'w' -> write
```
with open("text.txt", mode='w') as File:
   File.write('edited with python')
```
متن قبلی پاک میکند و متن جدید را جایگزین میکند

---

mode 'a' -> append
متن جدید را به متن فایل اضافه میکند
و هر چند بار اجرا بگیریم به همون اندازه اضافه میشود

---

r: Opens the file in read-only mode. Starts reading from the beginning of the file and is the default mode for the open() function.
rb: Opens the file as read-only in binary format and starts reading from the beginning of the file. While binary format can be used for different purposes, it is usually used when dealing with things like images, videos, etc.
r+: Opens a file for reading and writing, placing the pointer at the beginning of the file.
w: Opens in write-only mode. The pointer is placed at the beginning of the file and this will overwrite any existing file with the same name. It will create a new file if one with the same name doesn't exist.
wb: Opens a write-only file in binary mode.
w+: Opens a file for writing and reading.
wb+: Opens a file for writing and reading in binary mode.
a: Opens a file for appending new information to it. The pointer is placed at the end of the file. A new file is created if one with the same name doesn't exist.
ab: Opens a file for appending in binary mode.
a+: Opens a file for both appending and reading.

ab+: Opens a file for both appending and reading in binary mode.

پاک کردن فایل یا ترمینال
.flush()
https://www.geeksforgeeks.org/file-flush-method-in-python/

---

requestsنکاتی در مورد
در فایل ریکوئست نوشته شده

API
تبادل اطلاعات بین دو برنامه و پلتفرم
در اصل بر نامه ها با هم صحبت میکنند
api ها
به سه دسته تقسیم میشوند
1- private -> فقط داخل شرکت
2- partner -> بین دو شرکت
3- piblic -> عمومی برای همه
یکی از معروف ترین نوع هایش
web api
هست که برنامه اطلاعاتیرو از وب میگیره که پابلیک است
به عنوان مثال برنامه ی هواشناسی میگه یوزر
تهرانه حالا هوا چیه (از ای پی ای که سرویسش را خریده میپرسه)
در اصل برنامه هواشناسی هست ولی داده از خودش نیست

Data base -> محل ذخیره ی اطلاعات
Client -> کاربر
Server -> محل پردازش اطلاعات
Application -> رابط بین پایگاه داده و کلاینت ولی به همین ختم نمیشود

اسطلاحات دیتا بیس
Container -> Table -> Data
Table -> تعدادی از دیتا های کوچک / جدول
Diagram -> پیش نمایش دیتا ها به صورت درختی
Container -> ظرف / فایل اصلی و بزرگ دیتا بیس که به جدول ها بخش بندی میشود
????....

برای پایگاه داده در پایتون از
SQLite اس کیو لایت استفاده میکنیم + نصب
نصب:
https://www.sqlite.org/download.html
باید دو فایل را دانلود کنیم
1- 32/ 64 bit
2- a bundle of command line

___

هر دو فایل را استخراج میکنیم و بعد محتوات پوشه را در کنار
فایل هایdll , def
قرار میدهیم و در پوشه ای با نام دخواه
که بهتر است در درایو ویندوز قرار دهیم

___

this pc -> properties -> Advanced system settings -> Advanced -> Environment variables
درpath قسمت
دابل کلیک میکنیم و آدرس فولدری که ساختیم را اضافه میکنیم

فایل کامل شده
برای استفاده در پایتون کافی است
import sqlite3
که در کنار خود پایتون موجود است

وصل کردن فایل پایتون به دیتابیس
import sqlite3
connection = sqlite3.connect("./my-database.db") -> ./ به فولدر اصلی بر میگردد
#فایل اتوماتیک ایجاد می شود

ایجادtable
و ساخت دیتا بیس
import sqlite3
connection = sqlite3.connect("./my-database.db")
curcer = connection.curcer() -> برای اینکه اعمال روی دیتا بیس اتجام بشه

```
sql = """
   CREATE TABLE IF NOT EXISTS user (
      userId  INTEGER,
      name    VARCHAR (60),
      family    VARCHAR (60),

   );
"""
```

curcer.execute(sql) -> "sql" add to "curcer", curcer ---> connection
connection.commit() -> انجام و اجرای کار / حتما باید باشه تا نباشه انگار کاری انجام نشده
connection.close() -> بستن

point
, بعد از هر دستور
; بیرون از پرانتز ساخت جدول
""" """ برای متن های بلند

___
"""ایجاد کن یک جدول اگر وجود نداشت"""
اسم جدول
داخل پرانتز نام ستون ردیف (مثلا یوزرنیم)
تایپ ردیف
types-> INTEGER / VARCHAR
مثل قبلی و ۶۰ داخل پرانتز هم یعنی بیشتر از ۶۰ کاراکتر را نگیرد

___
یکبار که جدول بسازیم و اجرا کنیم دیگر نیازی به کد ها سازنده ی اون جدول نیست چون ساخته شده است و دیگه به برنامه متصل نیست

حالا دیتا بیس را میتوان در پای چارم از قسمت سمت راست دید database

چون قرار است کد سازنده را روی فایل دیتا بیس بنویسد از کرسر استفاده میکنیم

دیتابیس یک کان تینر یا همان ظرف است که اطلاعات درونس قرار میگیرد

عملیاتCRUD در دیتا بیس
Create Update Read Delete

Create:
```
sql = """
   INSERT INTO user VALUSE (1,"mahdi" ,"ebrahimi")
"""
```
point
اگر خواستیم چند آیتم را در لحه به جدول اضافه کنیم
دیگر نمیتوانیم از دستور
.execute(sql)
استفاده کنیم و مجبوری از دستور
.executescripts(sql)

_____
....
```
many_user = [
            (1,'mahdi','ebrahimi')
            (2,'mohammad','ebrahimi')
            (3,'jafat','goragbar')

          ]
Curcer.executemany("INSERT INTO user VALUSE (?,?,?)", many_user )
```
....

ادامهCRUD
Read:
```
import sqlite3
connection = sqlite3.connect("./my-database.db")
Curcer = connection.curcer()
sql = """
   SELECT * FROM user WHERE userId = 1  -> آنهایی را میارد که یوزر آی دیشان ۱ باشد اگر نبود هیچ
"""
curcer.execute(sql) -
for i in curcer:
   print(i)
connection.commit()
connection.close()
```

point
""همه چیز رو بخون
""آنهایی که ... هستند را بده
اطلاعاتش درون کرسر ذخیره میشود
برای دسترسی به آن از حلقه فور استفاده میکنیم

دستوری است برای اینکه بگوییم
""این داخلش باشه
```
SELECT * FROM user WHERE family LIKE "%ebrahimi%"
```
درصد عقب یعنی اول مهم نیست چی باشه
در صد آخر یعنی آخرش مهم نیست چی باشه
درصد عقب جلو باهم یعنی مهم نیست قبل و بعدش چی باشه و هر جای جمله بود اوکیه
در نتیجه اگر بدون درصد باشه یعنی فقط این داخل جدول باشه
%_% _% %_ _

ادامهCRUD
Update:
```
connection = sqlite3.connect("./my-database.db")
Curcer = connection.curcer()
sql = """
   UPDATE user SET name= "mohammad" WHERE userId = 1
"""
curcer.execute(sql)
#curcer.execute(UPDATE user SET name= "mohammad" WHERE userId = 1 )
connection.commit()
connection.close()
```

اگرWHERE
رو نذاریم تمام نام های جدول تغییر می کند

ادامهCRUD
Delete:
```
connection = sqlite3.connect("./my-database.db")
Curcer = connection.curcer()
sql = """
   DELETE FROM user WHERE userId= 1
"""
curcer.execute(sql)
connection.commit()
connection.close()
```

نحوه خرجی گرفتن از برنامه
pip install pyinstaller

___
بنویس IDEداخل ترمینال
pyinstaller --onefile --windowed mainFile.py
onefile -> بدون فایل های اضافهexeفقط یک فایل
windowed -> داخل یک صفحه باز شود نه داخل ترمینال

اضافه کردن یک فایل پایتونی به یک فایل پایتونی دیگر

ممکن است یک کد بنویسید که کاربری باشد و بخواهید اون رو داخل
یک فایل پایتونی دیگر استفاده کنید، اگر اسکریپت کاربردی پایتونی داخل یک پوشه با
فایل پایتونیتون باشه، میتونید اون رو به همون صورت که کتاب خانه هارو import میکنید
اون رو هم import کنید

: مثلا به این صورت
```
import MyScript
```

__name__ :
اگر اسکریپت اصلی باشد__main__ بر میگرداند
اگر فایلی باشد که اسکریپت رو درونش Import کردیم اسم فایل اسکریپت اصلی رو بر میگرداند

مثلا ما یک فایل داریم به نام

A
و میخواهیم از این اسکریپت برای یک فایل به نام
B
یعنی باید A رو در B اضافه کنیم
یک فایل هم به نام Cداریم که فایل B داخلش import شده

files:

A:
print(__name__)

B:
import A

C:
import B

Outputs:

A output>
__main__

B output>
"A"

C output>
"A"

point

اگر فایلی که داریم داخلش اسپیس باشه نمیتونیم از روش نرمال استفاده کنیم، مثلا
اسم فایلمون "Myfile 2.py" دیگر نمیتوانیم به صورت زیر واردش کنیم
import Myfile 2
ولی برای وارد کردنش میتوانیم از روش زیر استفاده کنیم
__import__("Myfile 2")
در import ساده چیزی داشتیم که میتوانستیم as بدیم بهش تا به
اسم دلخواهمون صداش کنیم، اینجا هم همین رو داریم
MyFunc = __import__("Myfile 2")
print(MF2.MyVariable)
یا به جای این روش
MyFunc = importlib.import_module("Myfile 2")

____
پایان بخش اصلی
____
____
____
____
_>
رصد کردن فایل های ویندوز با پایتون
آموزش os در نکات

_>

---

unit test
https://hamruyesh.com/product/how-unit-test-function-class-python/

تبدیل دیکشنری استرینگ شده به دیکشنری

```
string = "{'server1':'value','server2':'value'}"
s = string.replace("{" ,"")
finalstring = s.replace("}" , "")
list = finalstring.split(",")
dictionary ={}
for i in list:
   keyvalue = i.split(":")
   m= keyvalue[0].strip('\'')
   m = m.replace("\"", "")
   dictionary[m] = keyvalue[1].strip('"\'')
print(dictionary)
```

نکات تکمیلی چسباندن اعضای لیست

.join()

```
def namelist(names):
   nameList = [elem['name'] for elem in names]
   return ' & '.join(', '.join(nameList).rsplit(', ', 1))

namelist([{'name': 'Bart'},{'name': 'Lisa'},{'name': 'Maggie'},{'name': 'Homer'},{'name': 'Marge'}])
#Bart, Lisa, Maggie, Homer & Marge
```

ورودی های دیگر پرینت

دیگر پارامتر های پرینتmore print parameter

```
print('mahdiebi.exe','gmail.com', sep='@')  #-> mahdiebi.exe@gmail.com
print('mahdiebi.exe','gmail.com') #-> mahdiebi.exe gmail.com
print('mahdiebi.exe','gmail.com', sep='')  #-> mahdiebi.exegmail.com
print('mahdiebi.exe','gmail', sep='@', end='.com', )  #-> mahdiebi.exe@gmail.com

sample = open('samplefile.txt', 'w')
print('Text', file = sample)
sample.close()
```

ستاره ها در لیست هاasterisks

دسترسی به اعضای لیست با کمک ستاره

```
numbers = [2, 1, 3, 4, 7]
more_numbers = [*numbers, 11, 18]


print(*numbers)  #-> 2 1 3 4 7
print({*numbers})  #-> {1, 2, 4, 5, 7}  #class set
print(numbers)  #-> [2, 1, 3, 4, 7]
print(more_numbers)  #-> [2, 1, 3, 4, 7, 11, 18]
print(*more_numbers)  #-> 2 1 3 4 7 11 18
```

```
print(*more_numbers, sep=', ')  #-> 2, 1, 3, 4, 7, 11, 18

fruits = ['lemon', 'pear', 'watermelon', 'tomato']
print(fruits[0], fruits[1], fruits[2], fruits[3]) #-> lemon pear watermelon tomato
print(*fruits)  #-> lemon pear watermelon tomato
```

بزرگ کردن حرف اول استرینگ

```
.title()
print('hello world'.title())
->
Hello World
```

ریستارت و اجرای دوباره برنامه restart

```
def restart():
    import sys
    print("argv was",sys.argv)
    print("sys.executable was", sys.executable)
    print("restart now")
    import os
    os.execv(sys.executable, ['python'] + sys.argv)

restart()
```

کپی کردن copy to clipboard

```
import pyperclip
pyperclip.copy('The text to be copied to the clipboard.')
spam = pyperclip.paste()
```

پاک کردن اسپیس های اضافه سمت راست و چسباندن خط ها به هم

```
mahdi
mmd
kazem
~.rsplit()
mahdi
mmd
kazem
```

درصدی loading

```
import time, sys
def loading():
   print("Loading...")
   for i in range(0, 100):
      time.sleep(0.0001)
      sys.stdout.write(u"\u001b[1000D")
      sys.stdout.flush()
```

```
        time.sleep(1)
        sys.stdout.write(str(i + 1) + "%")
        sys.stdout.flush()
    print

loading()
```
->
1% -> ادامه پیدا میکند

لودینگ خطی و پیشرفتی
```
import time, sys
def loading():
    print "Loading..."
    for i in range(0, 100):
        time.sleep(0.1)
        width = (i + 1) / 4
        bar = "[" + "#" * width + " " * (25 - width) + "]"
        sys.stdout.write(u"\u001b[1000D" +  bar)
        sys.stdout.flush()
    print

loading()
```
->

https://b2n.ir/845795

پرینت کردن عدد ۱ تا صد داخل یک خط
```
stdout.write(string)
stdout.flush()
```

اضافه کردن چیزی به آخر رشته

```
txt = 'barbary'
x = txt.ijust(3,'zende')
```
->
barbaryzendezendezende

رمز نگاری با کتابخانه
pyaes
https://github.com/ricmoo/pyaes
کار با هش بهتر است #

وایرشارک شبکه

کتابخانه
ctypes
c استفاده از برخی توابع زبان

آپلود فایل
```
from google.colab import files
```

uploaded = files.upload()

تاریخ امروز
import datetime
datetime_object = str(datetime.datetime.now())[:10]
nowDate = str(datetime.date.today())

تفاوت list / array
list -> Different Data type
array -> Numeric Data type
array -> حجم کمتر

تک تک اجزا به توان ۲ <- array ** 2
ارور. چون باید داخل فور این کار ار بکنیم <- list ** 2

تک تک اجزا با هم جمع میشود <- array + array
list + list -> error

تبدیل لیست با آرایه
np.array([1,2,3,4])

تفاوت range / arange
arange اینت تولید میکنه و arrayتولید میکنه
rangeاینم اینت تولید میکنه ولی باید برای استفاده باید تبدلی به لیست کنیم

توان رساندن
np.power(2, 3) -> 8
np.power(np.arange(1,4) 3) -> [1 8 24]

منفی کردن اجزای آرایه
np.negative([1, 2, 3]) -> [-1, -2, -3]

---

np.exp([1 2 3])

لگاریتم
np.log(ARRAY)

سینوس
np.sin(ARRAY)

---

multi- dimensional Array
shape -> 1D / 2D / 3D
type -> int64 / تایپ های تمام اجزا باید برابر باشند

x = np.arange(3)
y = np.arange(3)
z = np.arange(3)
multi_array = np.array([x, y, z], #dtype= np.int32#)
print(multi_array)
print(multi_array.shape) -> (satr,soton)

print(multi_array.stype)

___

اعداد بین دو عدد
w = np.linespace(1, 10, 100) -> یک تا ده رو صد قسمت کن
print(w)

___

چند بعدی کردن داده آرایه یه بعدی
print(np.arange(9).reshape(3,3))
print(np.arange(9).reshape(3,-1)) # ستون اتوماتیک
print(np.arange(9).reshape(-1,3)) # سطر اتوماتیک
print(np.arange(18).reshape(2,3,3)) # 2 بعدی

___

___

___

END SOME NUMPY

___

___

___

بدست آوردن باینری عدد
bin(NUMBER)

___

عوض کردن ورژن مورد استفاده پایتون در CMD
python -V python 3.8

___

موارد استفاده از سیمیکالن ;
semicolon

___

تعریف تغیر
a = 2; a*=2
print(a)
-> 4

___

ایمپورت کردن کتابخانه
import pdb; pdb.set_trace()

___

کوتاه نویسی در شرط
if x < y < z: print x; print y; print z

___

چاپ اعداد مبنای ۱۶ به صورت مبنای ۱۰
اعداد مبنای ۱۶ رو میتوانیم همین جوری به پرینت بدهیم و عدد مبنای ۱۰ را چاپ میکند
0x000001D834C63580
??
0x
برا اعداد است یا برای خانه ی حافطه
برای نشان دادن خانه ی حافظه ای که
بعضی از موقع ها پایتون برمیگرداند
باید چیکار کنیم
??
>>>

___

تابع ها و سینتکس های حرفه ای و ریز تر
>>>

___

تبدیل استرینگ به کد
x = 1
print(eval('x + 1'))
-> 2

_____
from math import *
names = {'square_root': sqrt, 'power': pow}
print(eval('dir()', names))
# Using square_root in Expression
print(eval('square_root(9)', names))
->
['__builtins__', 'power', 'square_root']
3.0

_____
from math import *
a = 169
print(eval('sqrt(a)', {'__builtins__': None}, {'a': a, 'sqrt': sqrt}))
->
13.0

https://www.programiz.com/python-programming/methods/built-in/eval

---

کد اسکی حرف
اسکی به حرف <-
chr()
حرف به کد اسکی <-
ord()

print(ord('a'))
-> 97
print(chr(97))
-> 'a'
print(chr(ord('a')))
-> 'a'
https://www.programiz.com/python-programming/methods/built-in/chr

---

نوشتن اعضای یک لیست کنار شمارنده
The enumerate() method adds counter to an iterable and returns it (the enumerate object).

_____
enumerate(iterable, start=0)

_____
grocery = ['bread', 'milk', 'butter']
enumerateGrocery = enumerate(grocery)
print(type(enumerateGrocery))
-> <class 'enumerate'>
print(list(enumerateGrocery))
-> [(0, 'bread'), (1, 'milk'), (2, 'butter')]
enumerateGrocery = enumerate(grocery, 10)
print(list(enumerateGrocery))
->
[(10, 'bread'), (11, 'milk'), (12, 'butter')]

_____

```
grocery = ['bread', 'milk', 'butter']
for count, item in enumerate(grocery, 100):
   print(count, item)
->
100 bread
101 milk
102 butter
```

_____

خروجی این مانند zip است، پس میتوانیم آن را unzipکنیم

```
grocery = ['bread', 'milk', 'butter']
enumerateGrocery = enumerate(grocery)
a = list(enumerateGrocery)
print(list(zip(*a)))
->
[(0, 1, 2), ('bread', 'milk', 'butter')]
```

point

اگر counterبرایش نذاریم اون اعداد در اصل ایندکس میشود

https://www.programiz.com/python-programming/methods/built-in/enumerate

_____

asci()

??

The ascii() method returns a string containing a printable representation of an object. It escapes the non-ASCII characters in the string using \x, \u or \U escapes.

```
normalText = 'Python is interesting'
print(ascii(normalText))
->
'Python is interesting'

otherText = 'Pythön is interesting'
print(ascii(otherText))
->
'Pyth\xf6n is interesting'
print('Pyth\xf6n is interesting')
->
Pythön is interesting
```

_____

```
randomList = ['Python', 'Pythön', 5]
print(ascii(randomList))
->
['Python', 'Pyth\xf6n', 5]
```

https://www.programiz.com/python-programming/methods/built-in/ascii

_____

تبدیل استرینگ به بایت

byte()

```
string = "Python is interesting."
# string with encoding 'utf-8'
arr = bytes(string, 'utf-8')
```

```
print(arr)
->
b'Python is interesting.'
```

_____

ایجاد یک بایت به اندازه مورد نظر ??

```
size = 5
arr = bytes(size)
print(arr)
->
b'\x00\x00\x00\x00\x00'
```

_____

تبدیل لیست به بایت

```
rList = [1, 2, 3, 4, 5]
arr = bytes(rList)
print(arr)
-> b'\x01\x02\x03\x04\x05'
```

چیز مشابهی هم هست به نام bytearray

تفاوتشان ??

https://www.programiz.com/python-programming/methods/built-in/bytes
https://www.programiz.com/python-programming/methods/built-in/bytearray

---

فهمیدن کد مخصوص به یک آبجکت

این کد مانند هویت یک آبجکت است

```
print('id of 5 =',id(5))
-> id of 5 = 140472391630016
a = 5
print('id of a =',id(a))
-> id of a = 140472391630016
b = a
print('id of b =',id(b))
-> id of b = 140472391630016
c = 5.0
print('id of c =',id(c))
-> id of c = 140472372786520
```

https://www.programiz.com/python-programming/methods/built-in/id

---

copmlex()

??

https://www.programiz.com/python-programming/methods/built-in/complex

---

compile() دستور

??

https://www.geeksforgeeks.org/python-compile-function/
https://www.programiz.com/python-programming/methods/built-in/compile

---

دستور repr

این فانکشن یک ورودی میگیرد که اون ورودی میتواند هزجیزی باشد که قابل پرینت باشد

```
x = 'mmd is here'

print(repr(x))
print(eval(repr(x)))
-> 'foo'
```

فهمیدن این که یک آبجکت قابل فراخوانی است یا نه
callable()

```
x = 5
print(callable(x))    ->    False

def testfunc():
        print("Test")

y = testfunc
print(callable(y))    ->    True
```

---

dir()

فهمیدن اتریبوب های یک کلاس

https://www.programiz.com/python-programming/methods/built-in/compile

??

---

divmod()

https://www.programiz.com/python-programming/methods/built-in/divmod

??

---

delattr()

https://www.programiz.com/python-programming/methods/built-in/delattr

??

فهمیدن اینکه یک ابجکت داخل کلاس وجود دارد
hasattr()

```
class Person:
```

```
        age = 15
        name = 'mahdi'

person = Person()

print('Person has age ?', hasattr(person, 'age'))
print('Person has salary?', hasattr(person, 'email'))
->
        Person has age? True
        Person has email? False
```

دیدن اطلاعات کامل مربوط به فایل و کد

```
globals()

age = 15

globals()['age'] = age + 1
print('the age is : ', age)
-> the age is : 16


__

print(globals()['__name__'])
-> __main__
```

https://www.programiz.com/python-programming/methods/built-in/globals

---

دیدن متد های درون تابع

```
locals()

def localsNotPresent():
   return locals()
def localsPresent():
   present = True
   return locals()
print('localsNotPresent:', localsNotPresent())
print('localsPresent:', localsPresent())

->
        localsNotPresent: {}
        localsPresent: {'present': True}
```

https://www.programiz.com/python-programming/methods/built-in/locals

---

استراکچر فروزن ست

چیزی است مانند ست ولی تفاوت هایی دارد

```
A = frozenset([1, 2, 3, 4])
B = frozenset([3, 4, 5, 6])
```

کپی کردن فروزن ست
C = A.copy() # Output: frozenset({1, 2, 3, 4})
print(C)
-> frozenset({1, 2, 3, 4})

اجتماع دو فروزن ست
print(A.union(B)) # Output: frozenset({1, 2, 3, 4, 5, 6})
-> frozenset({1, 2, 3, 4, 5, 6})

اشتراک دو فروزن ست
print(A.intersection(B)) # Output: frozenset({3, 4})
-> frozenset({3, 4})

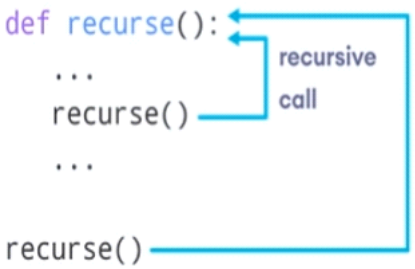
اجزایی که در A هست ولی در B نیست
print(A.difference(B)) # Output: frozenset({1, 2})

اجتماع منهای اشتراک
print(A.symmetric_difference(B)) # Output: frozenset({1, 2, 5, 6})

توابع بازگشتی

توابع باگشتی، توابعی هستند که طی یک حل مسائله و الگوریتم خاص، خود تابع در خود تابع اسفاده میشود

سرعت این توابع کند تر از حلقه ها است، چون حلقه ها از پشتیبانی داخلی پردازنده برخوردارند
و البته توبع بارگشتی در پیاده سازی خیلی ممکن است آن الگوریتم خاص غیر ممکن یا سخت باشد



در مسائل توابع بازگشتی سه چیز داریم
1) Decomposition
2) Composition
3) Base Case

Decomposition : ??
Composition : ??
Base Case : جایی که الگوریتم متوقف میشود، میتوان گفت جایی که دیگر تابع خودشو فراخونی نمیکنه

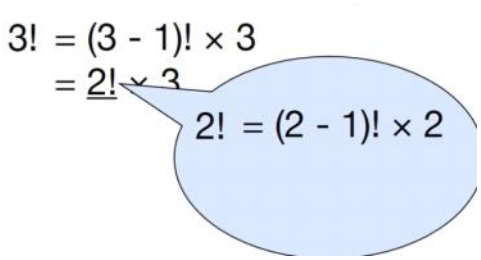
Data
structure...

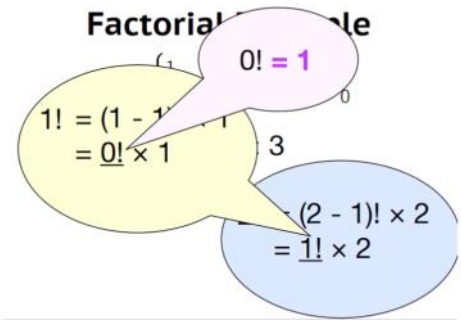
چند پیاده سازی با توابع بازگشتی

✓ Factorial

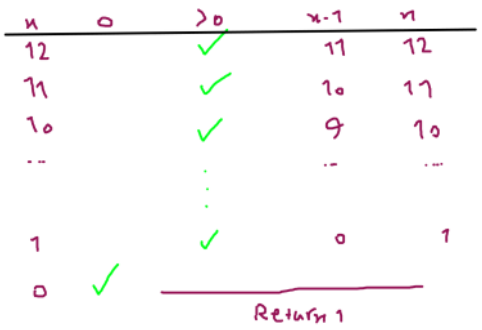
```
def fact(n):
    if n == 1:
        return 1
    else:
        return fact(n-1) * n
```

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ (n-1)! \times n & \text{if } n > 0 \end{cases}$$





- $N! = (N-1)! * N$ [for $N > 1$]
- $1! = 1$
- $3!$
  - $= 2! * 3$
  - $= (1! * 2) * 3$
  - $= 1 * 2 * 3$
- Recursive design:
  - Decomposition: $(N-1)!$
  - Composition: $* N$
  - Base case: $1!$



✓ Count Down

یعنی از اون عددی که به فانکشن ورودی میدهیم تا ۱ بیاید
مثلا ۳ بدهیم
۳
۲
۱

```
def Countdown(n):
    if n == 0 :
        return 1
    else:
        print(n)
        return Countdown(n - 1)
```

✓ Count Down Up

یعنی ما دو عدد بدیم و از عدد اول به سمت عدد دوم برود مثلا
10, 3
3, 2, 1, 0, 1, 2, . . . , 7, 8, 9, 10

```
def Count_Down_Up(N, n):
    if abs(n) > N:
        return
    else:
        print(abs(n))
        Count_Down_Up(N, n-1)
```

البته این یک مشکل دارد که در آخر یک None هم میدهد

✓ GCD or HCF

GCD(Greatest Common Divisor) or HCF(Highest Common Factor)
مقسوم علیه مشترک
بزرگ ترین مقسوم علیه ای که بر هر دو بخش پذیر باشد
24 = 1, 2, 3, 4, 6, 8, 12, 24
18 = 1, 2, 3, 6, 9, 18

GCD = 6

GCD(68, 119) = GCD(68, 51)
= GCD(17, 51)
= GCD(17, 34)
= **GCD(17, 17)**

در اینجا الگوریتم ریکرشن ما به این صورت هست که
مقسوم تفاضل بزرگ تر از کوچک تر فرق با خود اعداد ندارد
تا جایی این را ادامه میدهیم تا هر دو برابر شوند، سپس یکی از
آن اعداد مقسوم علیه مشترک ما میشوند

البته این اگوریتم برای اعداد بزرگ خیلی کار ساز نیست
و به ارور موری برخورد میکنیم

```
def GCD(a, b):
    if a < b :
        return GCD(a, b - a)

    elif b < a :
        return GCD(a - b, b)
    else:
        return a
```

Iterative Euclidean Algorithm

```
def gcd(a, b):
    while a != b:
        if a < b:
            b = b - a
        elif b < a:
            a = a - b
    return a

print gcd(68, 119)
```

point
همیشه میتوان، یک ریکرسیو رو به ایتریتیو (دارای حلقه) تبدیل کرد

✓ محاسبه ی باقی مانده ی دو عدد

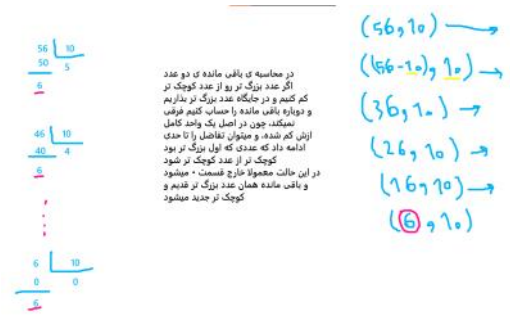
(56, 10) → (46, 10) → (36, 10) →
(26, 10) → (16, 10) → (**6**, 10)

56 % 10 = 6

ما اینجا، به جای اینکه اعداد رو از هم کم کنیم، میایم و مد آنها
یا همان باقی مانده اش را میگیریم

در محاسبه ی باقی مانده ی دو عدد
اگر عدد بزرگ تر رو از عدد کوچک تر
کم کنیم و در جایگاه عدد بزرگ تر بذاریم

و دوباره باقی مانده را حساب کنیم فرقی
نمیکند، چون در اصل یک واحد کامل
ازش کم شده، و میتوان تفاضل را تا حدی
ادامه داد که عددی که اول بزرگ تر بود
کوچک تر از عدد کوچک تر شود
در این حالت معمولا خارج قسمت ۰ میشود
و باقی مانده همان عدد بزرگ تر قدیم و
کوچک تر جدید میشود

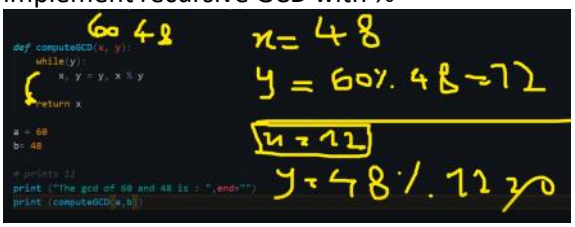


56 % 11 = 45 % 11 = 34 % 11 = 23 % 11 = 12 % 11 = 1 % 11 = 1

```
def mode(a, b):
    if a > b :
        return mode(a-b, b)

    elif a < b:
        return a
```

با ترکیب GCD و این روش مود یا باقی مانده به یک الگوریتم جدید برای GCD میرسیم که البته ریکرسیو نیست و حلقه دارد

implement recursive GCD with %



حالا اگر جای دو متغیر رو عوض کنیم

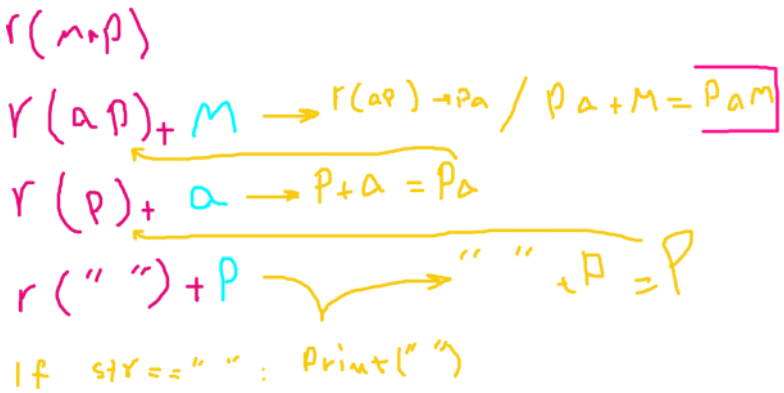
میبینیم اگر مراحل را برویم جلو بعد از چند مرحله اضافه به شروع مدل قبلی
که متغیر ها جابجا بودند میرسیم و قاعدتا نتیجه هم فرقی نمیکند

✓ String Reversal

در اصل میخواهیم حروف یا اعداد رو بر عکس کنیم
mahdi -> idham

```
def Reverse(string):
```

```
    if string == "":
        return string       # OR : return ""
    else:
        return Reverse(string[1:]) + string[0]
```



✓ sum digits

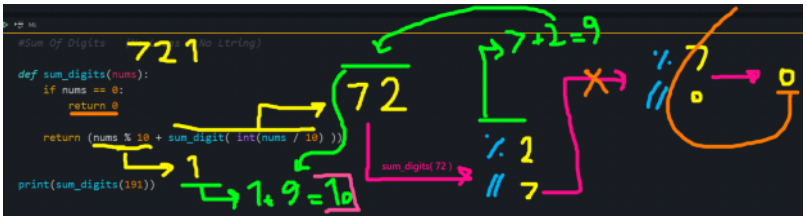
میخواهیم یک رشته اعداد ورودی دهیم و مجموع ارقام را بدهد، بدین صورت :
sum_digits(190000) → 1 + 9 = 10
sum_digits(198253) → 1 + 9 + 8 + 2 + 5 + 3 = 28

Recursive Algorithm

- **Sum of digits of 0 is 0.**
- **Sum of digits of N > 0:**
  Find last digit + sum of digits except last.
  N % 10 N // 10

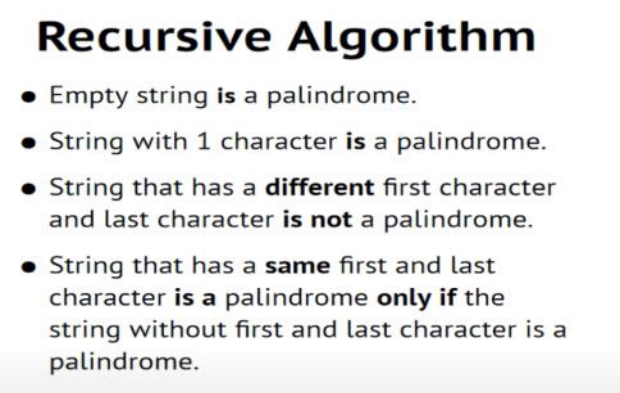
توضیح الگوریتم ریکرشن :
اعداد یک قابلیتی دارند که بدین صورت است
128 % 10 = 8
128 // 10 = 12
12 , 8 : 128
در اصل باقی مانده ی یک عدد به ۱۰ برابر با یکانش است
و خارج قسمت یک عدد به ۱۰ برابر با کل عدد به جز یکانش است
که این در اصل به خاصیت عدد ۱۰ و بر مبنای ۱۰ بودن اعداد ما بر میگردد

```
def sum_digits(nums):
    if nums == 0:
        return 0
    return (nums % 10 + sum_digit( int(nums / 10) ))
```



✓ Palindrome

پالیندرم، در اصل اعداد یا حروفی هستند که قبل و بعد وارونگی (مقلوب) فرق نمیکنند، مثلا:
121, 101, 0, level , civic, madam, malayalam, radar, reviver, rotator, terret, . . .




```
def palindrome(string):
    string = string.lower()
    if len(string) == 0 or len(string) == 1:
        return True

    elif string[0] != string[-1]:
        return False

    elif string[0] == string[-1]:
        return palindrome(string[1 : -1])
```

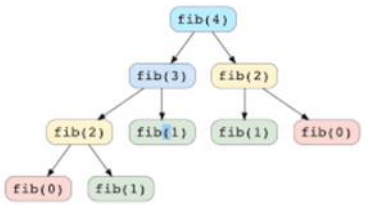
کد بهینه تر :

```
def palindrome(string):
    string = string.lower()

    if len(string) < 2:
        return True

    if string[0] != s[-1]:
        return False

    return palindrome(string[1 : -1])
```

✓ Fibonacci

پیاده سازی دنباله فیبوناچی
0, 1, 1, 2, 3, 5, 8, 13, . . .

fib(0)= 0
fib(1)= 1
fib(n) = fib(n-1) + fib(n-2)




```
def fib(n):
   if n <= 1:
       return n
   else:
       return(fib(n-1) + fib(n-2))
```

Recursive

این فایل شامل تمام پیاده سازی های ریکرسیو فانکشن های گفته شده در پایتون میباشد
به فرمت .ipynb که باید در jupyter اجرا شود

نصب کتابخانه های مورد نیاز به صورت دسته جمعی

یک فایل متنی میسازیم و کتابخانه های مورد نیاز رو توش لیست میکنیم
requirements.txt →
redis
requests
opencv-python

>>> pip install -r requirements.txt

