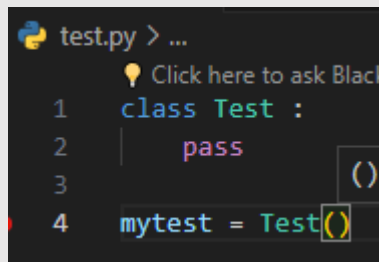


## تمرینات بخش ابتدایی شعر گرای

---

یک مروری بر شعی گرای داشته باشیم و یک موضوع جدید به اسم اسکوپ :

اول بیاییم یه کلاس ساده بسازیم :



```
test.py > ...  
Click here to ask Black  
1 class Test :  
2     pass  
3  
4 mytest = Test()
```

سینتکس کلی یک کلاس به این شکل است :

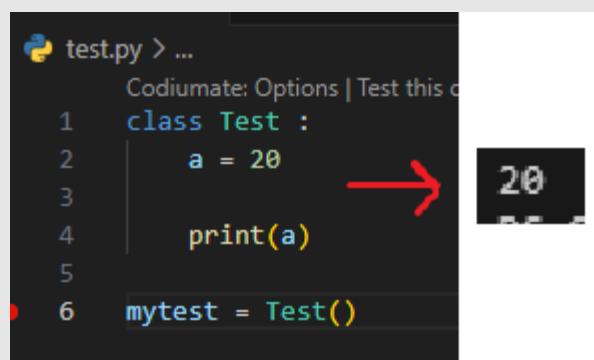
class Classname :

statements

object = Classname()

اول یک اسم برای کلاس تایین میکنیم و آنرا مینویسم و ویژگی میدیم و بعد برای ساختن آبجکت یک متغیر میسازیم و کلاس رو بهش میدیم داخل پرانتز . چرا ؟ چون مثل توابع باید صدا زده شوند.

حالا داخل کلاس ها میتوان از دستورات مختلفی بهره‌مند شد :



```
test.py > ...
Codiumate: Options | Test this c
1 class Test :
2     a = 20
3
4     print(a)
5
6 mytest = Test()
```

A red arrow points from the line `a = 20` in the code to a small black box containing the number `20`, which represents the output of the `print(a)` statement.

اینجا این کلاس شبی تابعی هست که در آن یک مقدار 20 پرینت میشود

---

متغیر های گلوبال و لوکال :

به متغیر هایی که فقط داخل یک بلاک خواصی در دسترس هستند متغیر های لوکال میگویند مثل آی پی های لوکال که فقط با دستگاه های داخلی قابل دسترس هستند

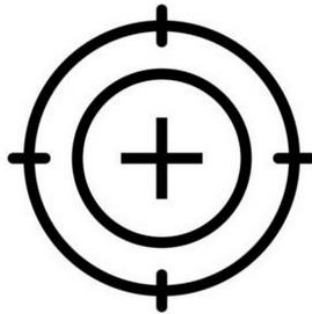
به متغیر هایی که در سراسر برنامه ما قابل دسترس هستند متغیر های گلوبال مینامند

یک متغیر گلوبال :

```
test.py > ...  
  Click here to ask Blackbox to help  
1 my_num = 12  
2 my_string = 'erfan'  
3 my_list = [1,2,3]  
4  
5 if my_num == 12 :  
6     print(my_string)  
7  
8 def show(my_list):  
9     print(my_list)  
10 show(my_list)  
11  
12 while False :  
13     print(my_list)  
14
```

همونطور که میبینید متغیر هایی که تعریف کردیم در هر جا از کد ما و داخل هر بلوکی از کد ها در دسترس هستند

دایره لغات خود را زیاد کنید . scope : محدوده



ولی متغیر هایی که داخل بلاک های کدی نوشته میشوند مانند توابع و کلاس ها فقط در همان اسکوپ ( بازه ) خود همان بلاک در دسترس هستند برای مثال :

```
test.py > ...  
Click here to ask Blackbox to help you code faster  
1 a = 12  
2  
Codiumate: Options | Test this function  
3 def show():  
4     b = 21  
5  
6     print("its from function")  
7     print(a)  
8     print(b)  
9  
10  
11 print("its from out of fuction ")  
12  
13 print(b) "b" is not defined
```

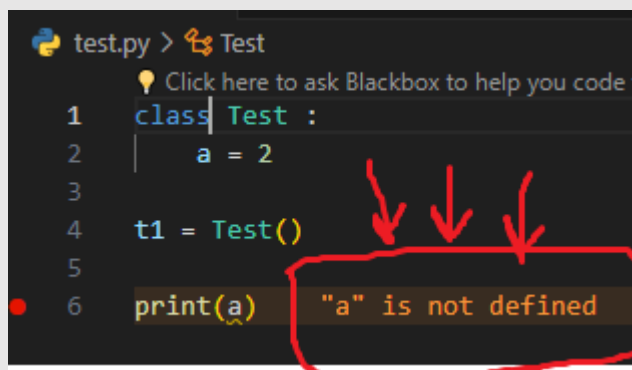
```
Traceback (most recent call last):  
File "c:\Users\hamid\Desktop\New folder (2)\test\test.py", line 13, in <module>  
    print(b)  
    ^  
NameError: name 'b' is not defined
```

همانطور که میبینید متغیر b چون داخل تابع ساخته شده و فقط در اسکوپ خود تابع در دسترس هست وقتی خارج از تابع صدا زده میشود ارور وجود نداشتن متغیر را دریافت میکنیم

باز می‌گم فقط موقعی که **تشکیل** شون داخل یک اسکوپ بلاکی باشد در جاهای دیگه قابل دسترس نیستند مثلاً یک متغیر خارج از تابع تعریف میشه و به تابع هم پاس داده میشه و خارج از تابع هم منتقل میشه چون خارج از اسکوپ تابع و در اسکوپ گلوبال کل پروژه تعریف شده

حالا اینارو گفتیم تا برسیم به متغیرهای گلوبال اسکوپ و لوکال اسکوپ کلاس‌ها :

وقتی متغیر ای همینجوری لُختی تعریف میشه وسط کلاس یک متغیر گلوبال ( در اسکوپ کلاس ) خونده میشه ولی در اسکوپ کل پروژه گلوبال نیست چرا ؟ چون خارج از کلاس به صورت مستقیم در دسترس نیست و نسبت به اسکوپ و زاویه دید داخل کلاس گلوبال محسوب میشه



```
test.py > Test
Click here to ask Blackbox to help you code f
1 class Test :
2     a = 2
3
4 t1 = Test()
5
6 print(a) "a" is not defined
```

The screenshot shows a Python IDE with a file named 'test.py'. The code defines a class 'Test' with a class attribute 'a' set to 2. It then creates an instance 't1' of the 'Test' class. Finally, it attempts to print 'a'. A red box highlights the error message '"a" is not defined' which appears because the print statement is outside the class scope and 'a' is not a global variable.

همونطور که میبینید متغیر a از خارج از کلاس در دسترس نیست پس این یک متغیر لوکال با دید از کل پروژه نسبت به کلاس و یک متغیر گلوبال با دید داخلی کلاس هست

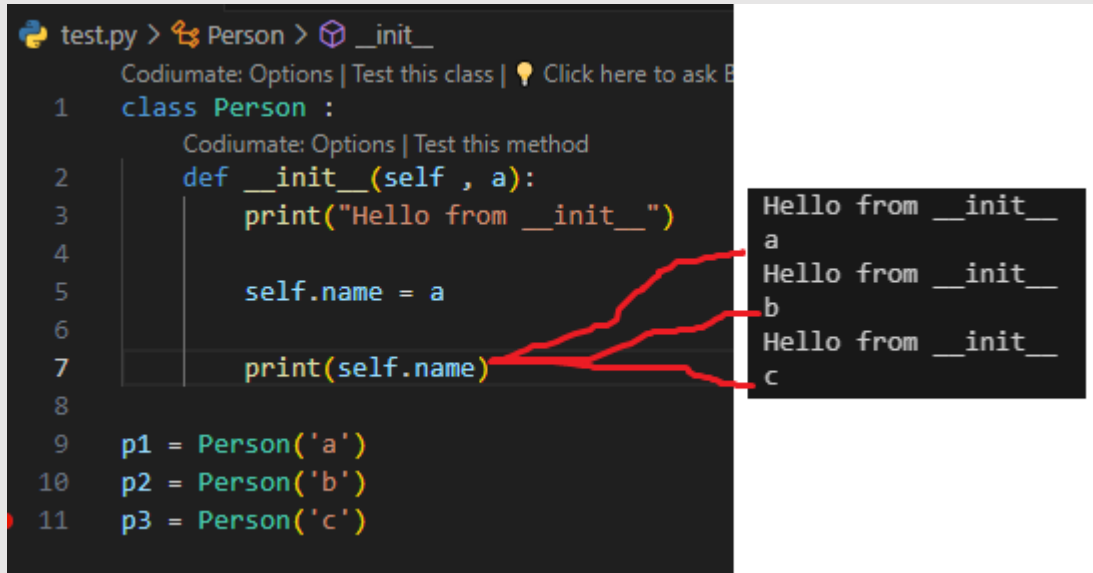
---

دایره لغات خود را افزایش دهید : method : به توابعی که داخل کلاس ها نوشته میشوند  
متود میگویند :

```
test.py > Test > hi
Codiumate: Options | Test this class | Click H
1 class Test :
2
3     Codiumate: Options | Test this method
4     def show():
5
6         print("hello")
7         Codiumate: Options | Test this method
8     def write():
9
10        print("write")
11        Codiumate: Options | Test this method
12    def hi():
13
14        print("hi")
```

به هر یک از این توابع یک متود میگویند . هوش مصنوعی هم این رو تایید میکنه (:

تابع `__init__` : قبلاً گفته شده ولی بخاطر اهمیت این موضوع باز می‌گم : تابعی هست که موقع نمونه سازی از کلاس اجرا میشود :



```
test.py > Person > __init__
Codiumate: Options | Test this class | Click here to ask B
1 class Person :
  Codiumate: Options | Test this method
2   def __init__(self , a):
3       print("Hello from __init__")
4
5       self.name = a
6
7       print(self.name)
8
9   p1 = Person('a')
10  p2 = Person('b')
11  p3 = Person('c')
```

Output:

```
Hello from __init__
a
Hello from __init__
b
Hello from __init__
c
```

3 تا آبجکت گرفتیم که با هربار درست شدن ابجکت یکبار تابع داند اینیت اجرا میشود ( تمام محتویات آن ) آن ورودی ها هم به `a` می روند در ارگومان تابع داند اینیت و داخل متغیر لوکال از اسکوپ کلاس به نام `name` که در نماینده خود یعنی `self` ذخیره شدند می‌رود.

برای صدا زدن متد های معمولی از تابع میتوان از ابجکت ها استفاده میکنیم . چرا ؟ چون متد ها ویژگی هایی هستند که به تمام ابجکت ها اختصاص داده میشوند مثل لیست ها در پایتون . متد `copy` در لیست ها به تمام لیست هایی که میسازیم اختصاص داده میشود

```
test.py  X
test.py > ...
Codiumate: Options | Test this class | Click here to ask
1 class Test :
2     def __init__(self):
3         print("hello to my class")
4
5     def t1(self):
6         print('hello from t1')
7
8     def t2(self):
9         print("hello from t2")
10
11    def t3(self):
12        print("hello from t3")
13
14    myt1 = Test()
15    myt2 = Test()
16
17    myt1.t1()
18    myt1.t2()
19    myt2.t2()
20    myt2.t3()
```

```
hello to my class
hello to my class
hello from t1
hello from t2
hello from t2
hello from t3
```

همونطور که میبینید ما از داخل ابجکت های myt1 myt2 اومدیم و متد هارو ( تابع هارو  
( صدا زدیم

دقیقا مثل کاری که با رشته ها و .. میکریم :



```
test.py > ...  
  Click here to ask Blackbox to h  
1  mystr = 'hello'  
2  
3  mystr.capitalize()  
4  mystr.center()
```

وقتی ما یک رشته تعریف میکنیم در اصل یک ابجکت از کلاس رشته رو درست کردیم که تمام ویژگی هایی تاییین شده در کلاس رشته رو دارد مثل متد های `capitalize` یا ویژگی تیکه تیکه شدن (`slicing`) یا ویژگی قابل شمارش بودن و ...

---