

تمرین های تکمیلی

```
class Test : #class
    a = 20
    b = 10
    n = 30

    def __init__(self):
        print(self.a)
        print(self.b)
        print(self.n)

p1 = Test() #instance
p2 = Test()
p3 = Test()
p4 = Test()
```

تنها نکته این کد اینه که : تمام ویژگی های گلوبال در اسکوپ کلاس در سلف ها قابل دسترسی هستند و نیازی به داخل سلف ریختن شون نیست و خود به خود داخل سلف ذخیره میشوند . باید بدونید که این ویژگی های گلوبال در تمامی اینستنس ها در دسترس هستند .

```
Codiumate: Options | Test this class
class Person :
    type1 = 'HUMAN'

Codiumate: Options | Test this method
def __init__(self , a , b) :
    self.name = a
    self.lastname = b

    print("from __init__")
    print(self.name)
    print(self.lastname)

p1 = Person('a' , 'b')
p2 = Person('G' , 'e')

print("from out of class")
print(p1.name)
print(p2.lastname)
print(p2.type1)
print(p1.type1)
```

```
from __init__
a
b
from __init__
G
e
from out of class
a
e
HUMAN
HUMAN
```

در اینجا اول یک ویژگی به اسم type1 به عنوان ویژگی گلوبال داده میشه به تمام نمونه ها بعد داخل تابع داندر اینیت که موقع نمونه گیری اجرا میشه دو ارگومان میگیریم به اسم a و b که چون این تابع موقع نمونه گیری اجرا میشوند پس همونجا هم انتظار میره این 2 ارگومان پاس داده شوند (موقع ساخت p1 p2)

بعد این 2 ارگومان در متغیر های لوکال که فقط مخصوص نمونه ها هستند به اسم self.name و self.lastname ذخیره میشوند و بعد پرینت میشوند

با هر بار نمونه گیری تابع داندر اینیت اجرا میشود و ما چون الان 2 بار نمونه گرفتیم 2 بار انتظار داریم این تابع کد های داخلش اجرا بشه . پایین تر هم اومدیم ویژگی های نمونه ها رو پرینت گرفتیم . که name و lastname که براشون مشخص بودن . type1 رو هم برای جفتشون پرینت گرفتیم تا نشان دهیم که این ویژگی به جفت نمونه ها به صورت یکسان داده شده.

```
class Test :
    def __init__(self , a ):
        self.a = a
    def __str__(self):
        return f'{self.a} is the name'

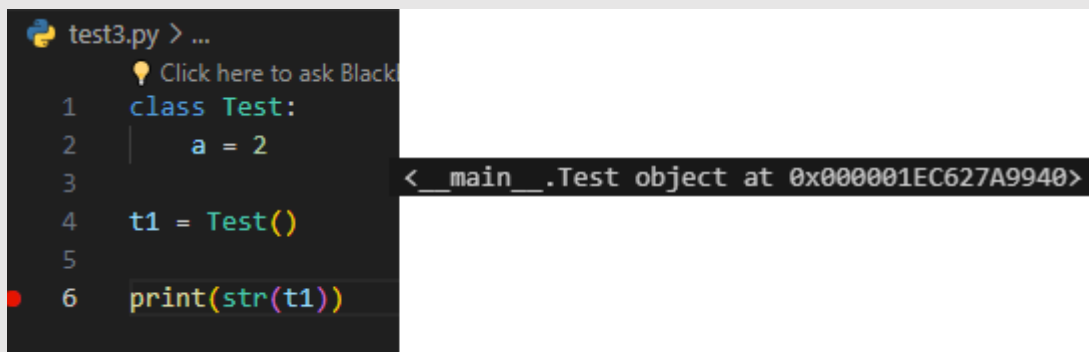
t1 = Test('a')
t2 = Test("b")
print(str(t1))
```

در اینجا اومدیم یک داندر جدید به اسم داندر اس تی ار درست کردیم که همون طور که گفتیم : این داندر ها نشون دهنده رفتار های نمونه ها در مقابل با یکسری تغییرات هستند .

!!! تمام داندر ها باید اون نتیجه رو return کنن !!!

`__str__` : وقتی نمونه ها تبدیل به str شوند کد های داخل ای متود به عنوان رفتار اجرا میشوند .

در حالت عادی وقتی نمونه هارو تبدیل به رشته کنیم این اتفاق میوفته :



```
test3.py > ...
Click here to ask Black
1 class Test:
2     a = 2
3
4 t1 = Test()
5
6 print(str(t1))
```


<__main__.Test object at 0x000001EC627A9940>

رفتاری رو داریم میبینیم که مورد انتظار ما نیست . ما میخاهیم وقتی از نمونه ها یک رشته ساخته میشوند یکسری رفتار دلخواه مارو از خودشون نشون بدن

درکد اولیه : در `__str__` مشخص کردیم که اگر از روی نمونه ها رشته ساخته شد بیا و برگردون : یک رشته رو که توش `self.a` رو هم داره (کد اول سوال)

داخل این ها هم کد های دیگری هم میشه نوشت :

```
test3.py > Test > __str__
Codiumate: Options | Test this class | Click here to ask Black
1 class Test:
2     a = 2
3
4     Codiumate: Options | Test this method
5     def __str__(self):
6         print("opening __str__")
7         print(self.a)
8         return f'{self.a} is returned'
9
10 t1 = Test()
11
12 print(str(t1))
```



```
opening __str__
2
2 is returned
```

اینجا موقعی که یک رشته از نمونه کلاس میگیریم , تابع داند اس تی ار اجرا میشه و داخلش اول پرینت میشه `opening __Str__` و بعد پرینت میکنه متغیر `a` رو و بعد ریترن میکنه یک رشته ای رو که `self.a is returned` هست .

!!!! اون چیزی که ریترن میشه ریخته میشه توی `str(t1)` و بعد پرینت میشه (خط آخر)

مثال آرایش گری :

```
class client:
    type_now = 'barber'
    language = 'farsi'

    Codiumate: Options | Test this method
    def __init__(self , a , b , c):
        self.name = a
        self.lastname = b
        self.charge = c
        print(f"welcome to our barber shop {self.name} with {self.charge}")

    Codiumate: Options | Test this method
    def purchase(self):
        if self.charge < 1000 :
            return False
        else :
            return True

    Codiumate: Options | Test this method
    def reserve(self , t):
        if self.purchase():
            print(f"you are reserved for {t}")
        elif self.purchase != True :
            print("you dont hav eenough charge in your wallet")
        else :
            print("a problem accured , pleaes try again later")

    def exit(self):
        print("client has exited the barber shop")

client1 = client('erfan' , 'safarali' , 500)
client1.reserve(14)
```

```
welcome to our barber shop erfan with 500
you dont hav eenough charge in your wallet
```

اول 2 ویژگی به اسم type_now و language به صورت گلوبال داده میشه .

داخل __init__ 3 تا ورودی میاد که داخل name و lastname و charge ریخته میشود . آخرشم یک پرینت داریم که وقتی نمونه ساخته شد و ارگومان ها داده شدن اطلاعات اون نمونه رو چاپ بکنه

یک تابع به اسم purchase داریم که چک میکنه اگر charge کمتر از 1000 بود برگردونه False و اگر بزرگتر بود برگردونه True

این تابع در تابع بعدی کاربرد داره : تابع رزرو : یک ارگومان میگیره به اسم t (فقط زمان صدا زده شدن از کاربر میخادش و چون نیازی بهش نداریم جایی ذخیرش نکردیم) چک میکنه اگر اون چیزی که تابع purchas برمیگردونه True بود چاپ کنه که وقت برای شما رزرو شده در t (همون ورودی تابع) و اگر نبود پرینت کنه که اعتبار شما کافی نیست

یک تابع exit هم داریم که فقط چاپ میکنه که کاربر خارج شد

حالا برای اجرا : یک نمونه گرفته میشه به اسم client1 که عرفان داخل a و صفرعلی داخل b و 500 داخل c ذخیره میشه که اینا به ترتیب در متغیر های لوکال name lastname و charge قرار میگیرند . بعدشم پرینت میکنه این اطلاعات رو بخوابر پرینتی که داخل __init__ گذاشتیم

بعد متد reverse رو روی اون نمونه اعمال میکنیم و 14 رو بهش میدیم تا داخل ارگومان t قرار بگیره . حالا تابع purchase اجرا میشه چون گفتیم : if self.purchase() یعنی اگر مقدار return شده این تابع True بود چون من مقدار 500 رو داده بودم پس تابع Purchase فالس برمیگردونه پس اجرا نمیشه شرط اول داخل reverse و شرط دومی اجرا میشه که میگه شما اعتبار کافی ندارید

مثال فروشگاه موبایل :

```
class Object :
    from_country = 'iran'
    licence = 'iranian standard organization'

    Codumate: Options | Test this method
    def __init__(self , input_name , input_price , input_type):

        self.name = input_name
        self.price = input_price
        self.type = input_type

        print(f"your object is successffully regisrered , its name is {self.name} with the price of {self.price}")

    Codumate: Options | Test this method
    def purchase(self , user_wallet):
        print("stating purchase")
        success_list = [ ]
        print("checking your wallet ...")

        if user_wallet > self.price :
            success_list.append(self.name)
        else:
            print("you dont have enough money")
            print("are you elligible to use you mortgage or credit card ? ")

        return success_list

    def logout(self):
        print("you are successfully logged out")

    def charge(self , input_charge):
        print("charging your account")

    Codumate: Options | Test this method
    def __str__(self) :
        print("entering __str__ method")
        return self.name.upper()

xiaomi_redmi_note_10_pro_max = Object('note10' , 14000000 , 'mobile')
print(xiaomi_redmi_note_10_pro_max.purchase(1000000000))

xiaomi_redmi_note_10_pro_max.logout()
print(str(xiaomi_redmi_note_10_pro_max))
|
if str(xiaomi_redmi_note_10_pro_max ) == 'note10':
    print((str(xiaomi_redmi_note_10_pro_max).upper()))
else :
    print("error")
```

your object is successffully regisrered , its name is note10 with the price of 14000000
stating purchase
checking your wallet ...
['note10']
you are successfully logged out
entering __str__ method
NOTE10
entering __str__ method
error

اول 2 مقدار گلوبال رو وارد کلاس میکنیم که به تمام ابجکت ها داده بشه .

بعد داخل متود داندر اینیت 3 تا ارگومان میگیریم و اونهارو به داخل 3 مقدار name و price و type و آخر هم توی یه پیام پرینت میکنیم که محصول تون با موفقیت ثبت نام شد و نامش این هست و قیمتش هم این

یک متود ساختیم به اسم purchase که مثل مثال قبلی یک ارگومان موقع صدا شدنش میخاد به اسم user_wallet ولی جایی سیو نمیشه . بعد پرینت میکنه شروع به خرید بعد یک لیست از محصولات خریده شده میسازیم . پرینت میکنیم که چک کردن اعتبار شما . بعد شرط میزنیم که اگر اون ارگومان ورودی اون تابع یعنی user_wallet بزرگتر از

self.price بود بیاد اون محصول رو به لیست اضافه کنه اسمش رو وگرنه چاپ کنه که قیمت محصول از جیب شما بیشتره ایا میخایی وام بگیری یا از کارت اعتباری استفاده کنی؟

بعد هم در آخر اون لیست رو return میکنه تا یه جای دیگه ذخیرش کنیم یا چاپش کنیم

2 تا تابع logout و charge هم داریم که ساده هستن

یک تابع __str__ هم داریم تا اگر از ابجکت هامون یک رشته ساختن رفتار که میخاییم برامون اجرا بشه یعنی اول چاپ کنه که وارد داندر استرینگ شدیم و بعد برگردونه self.name.upper() چرا؟ میخام اسم اون ابجکت رو که رشته هست رو بیاره و متد upper() که برای رشته ها بود و حروف رو بزرگ میکرد رو براشون چاپ کنه

حالا یک ابجت میسازیم به اسم یک گوشی و بعد بهش 3 مقدار ارگومان های داندر اینیت رو میدیم

بعد هم تابع purchase رو بهش اون ارگومان خودشو میدیم و میریزیمش توی پرینت . چرا؟ چون گفتیم قراره اون لیست محصولات خریداری شده رو return کنه .

بعد متد لاگ اوت رو صداش زدیم

بعد هم از روی اون ابجکت یک رشته ساختیم تا تابع __str__ اجرا بشه

حالا پایین تر گفتم اگر مقدار رشته اون باجکت برابر بود با note10 بیا و یکاری رو بکن اگر نبود برگردن ارور

حالا چرا ؟ اگر خروجی رو دقت کنید میبینید که 2 بار چاپ شده `entering __str__` چرا ؟ چون بیار اون بالا پرینت کردیمش و یکبار اینجا براش شرط زدیم پس 2 بار اجرا شد . اگر شرط درست باشه میشه 3 بار اجرا بشه چون گفتم اگر درست بود بیا همونو بیار و `upper` رو روش اجرا کن