Year. Month. Date. Subject ۰۳-۱۲

اینجا یه کلاس می‌نویسیم خیلی ساده بنویسید.

```
class Life:
    def __init__(self, name='unknown'):
        print('hello : ' + name)
        self.name = name
    def live(self):
        print(self.name)
    def __del__(self):
        print('good bye : ' + self.name)
```

این کلاس سه تا تابع داره. این تابعا هرکدوم چیکار می‌کنن؟

یعنی اگه کاربر اسم وارد نکنه اسمش میذاره unknown اگه ob = life('sara') اگه اینو اجرا کنم توی خروجی چی چاپ میشه؟ باید ببینید توی تابع سازنده init ، print هست باید اول اسم چاپ بشه چون تابع (سازنده اجرا میشه)

init به صورت بدیهی اجرا میشه وقتی که صدا زده میشه پس hello:sara خروجی این چیه؟

```
ob.live()
```

↓ alireza

```
ob2 = Life()
```

↓ hello : alireza

اگه این یکی رو ورودی بهش ندم چی چاپ میشه؟ هر اسمی که اینجا بذاریم برامون میذاره چون اینجا مقدار ندادم توی ورودی اتوماتیک برامون unknown ست میشه.

```
class C:
    def __init__(self, a):
        self.a = a
    def f(self, x, y):
        return (self.a + x + y)

ob = C(1)
print(ob.f(2,3))
```

↓ 6 بالایی جای self مقدار ندادیم

اگه توی تابع init ، print بذاریم چیزی برنمی‌گردونه! اینجا چیزی برمی‌گردونیم موقع صدا زدن میشه.

یعنی ... self که اینجا صدا زده شده یعنی با اسم کلاس تابع فراخوانی میشه.

```
print(C.f(ob,2,3))
```

↓ 6 باید جای self مقدار بدیم یعنی ob

تفاوت این دو تا دستور خروجی 6 ، بالایی با اسم شی پایینی با اسم کلاس

Year. Month. Date. Subject

```
life.live(ob)
      ↓
     sava
```

اگه بخوایم ob.live هم مثل سوال دیگه با کلاس صدا بزنیم میشه :

```
ob.live() → جای self خالی
```

کلاس A ، فقط def اول رو داره و def دومی داخل تورفتگی نیست و یه تابع جداست . حالا سوال main() چه خروجی ای داره؟

```
def add(obj, K)
    obj.L += 1
    K += 1
class A:
  def __init__(self):
      self.t = 1
def main():
    ob = A()
    K = 0
    add(ob, K)
    add(ob, K)
    print(ob.t)
    print(K)
main()
```

مربوط به مبحث دامنه متغیرها : متغیر سراسری ، هر وقت بیرون از تابع تعریف می‌کنیم .

اینجا K داخل تابع main اعتبار داره به علاوه print ها اما اینجا K رو به تابع ad دادیم ولی توی تابع اولی ورودی K داریم .

مثال متغیر سراسری :

```
x = "global"  → متغیر سراسری
def foo():
  print("x inside :", x)
  y = "ash"  → متغیر محلی
foo()
print("x outside :", x)
print(y) → خطا میده چون خارج هست
```

هرجایی که تعریف شده همون جا هم دیده میشه ، داخل تورفتگی باشه بیرونش دیده نمیشه .

بیرون تابع تعریف شده → global

تمرین :

```
x = "global"
def foo():
    x = x * 2
    print(x)
Foo()
Print(y) → خطا میده چون محلی تعریف شده
```

• از بیرون تابع قابلیت دسترسی به محلی ها رو نداریم .

```
x = "global"
def foo():
    global x        → گفتیم متغیر تابع از نوع سراسری باشه
    y = "local"
    x = x * 2
    print(x)
    print(y)
```

↓ خروجی

```
globalglobal
local
```

```
x = 20      # global
def foo()
    x = 5   # local
    x = x * 2
    print(x)
foo()
```

```
x = 5
def foo():
    x = 10
    print("local x:", x)
foo()
```

↓

10 چون foo محلی رو میده

print (n) → بیرون تابع (فراخوانی)

خروجی : 5

---

property : خروجی 6 میشه ، و تابع area از کلاس Rect ایجاد شده ، و اینجا print(s.area) چون از Rect ارث بری کرده پس تابعش رو داره . اینجا (۲۰ area) پرانتز نداره و به شکل تابع باهاش برخورد نشده بلکه یک دیتیه ، چون property داره ، دسترسی بهش از طریق gett و sett اتفاق میفته .

Year. Month. Date. Subject

```
class Point:
  def __init__(self, x, y)
      self.x = x
      self.y = y
  def dist(self, pt)
      a = pt.x-self.x    ← فاصله بین دو نقطه در محور مختصات    3-2=1
      b = pt.y-self.y                                            3-3=0
      return math.sqrt(a**2+b**2)
p1 = Point(2,3)
p2 = Point(3,3)  یا (6,7) با این میشه → 5.65
print(p1.dist(p2))
    ↓
   1.0
```

اینجا p2 میشه pt ام بوده و همینطور p1
و شئ ای که تابع dist فراخوانی کرده p1

```
class B:
  def __init__(self,a,b,c):
     self.a = a
     self._b = b
     self.__c = c
 def f(self):
     print(self.a)
     print(self._b)   → protected
     print(self.__c)  → private
ob = B(1,2,3)
      → ob.f()   خروجی 123
print(ob.a) → 1
            → print(ob._b) → 2          print(ob.__c) → خطا میده
print(ob._B__c) → 3  → با اسم کلاس فراخوانی شد
ob = B(1,2,3)
print(ob.a) → 1
print(ob._b) → 2
print(ob._B__c) → 3
```

Honar

```
class Rect:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    @property
    def area(self):
        return self.x * self.y
class Square(Rect):
    def __init__(self, z):
        super().__init__(x=z, y=z)

r= Rect(2,3)
print(r.area)
s= square(5)
print(s.area)
```

↓
6
25

نکته : متغیرهای که کسی نزدیکش نیست جدا زده میشه .

```
class A:
    def __init__(self):
        self.__x=1
        self.y=2
    def p(self):
        print(self.__x, self.y)
class B(A):
    def __init__(self):
        super().__init__()
        self.__x= 3
        self.y= 4

b=B()
b.p()
print(b._B__x)
```

↓ 1 4

مثال‌ها

print(b._A__x) → 1

print(b._A__y) → خطا

print(b.y) → 4

برای B رو حساب میکنه

b.p() → 1 3

print(b.y) → 4

print(b._B__y) → x خصوصی نوشتن

Honar

```
class A:
    def __init__(self):
        print('A')
        self.x = 5
    def func(self):
        self.x = 2
        print(self.x)
class B(A):
    def func(self):
        self.x += 7
        return self.x
b = B()
print(b.func())
```

A
6