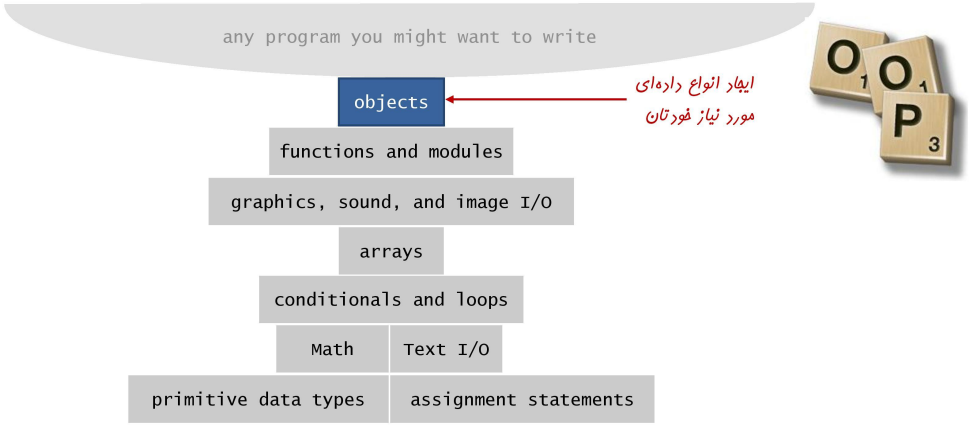
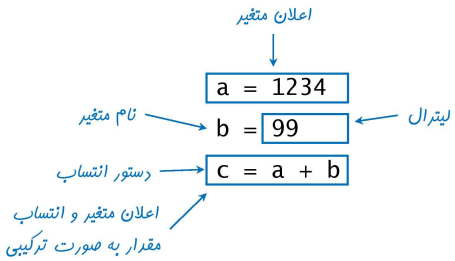
اجزای برنامه‌نویسی



تعاریف اولیه

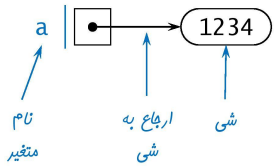
- اشیا. تمام مقادیر داده‌ای در یک برنامه پایتون به وسیله اشیا و روابط میان اشیا بازنمایی می‌شوند.
  - یک شی، یک بازنمایی درون-حافظه‌ای از یک مقدار از یک نوع داده‌ای خاص است.
    - مکان در حافظه
    - نوع داده‌ای: بیانگر رفتار شی (مقادیر و عملیات)
    - مقدار



تعاریف اولیه

- متغیر. نامی برای ارجاع به یک شی.
- دستور انتساب. مقید کردن متغیر سمت چپ به شی ایجاد شده در سمت راست عملگر انتساب.

```
a = 1234
```

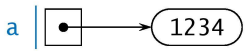


متغیر به یک شی ارجاع می‌کند

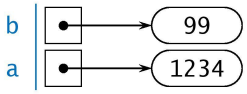
ردیابی (در سطح شی)

- ردیابی در سطح شی. دنبال کردن اشیا و ارجاع‌ها برای یک درک بهتر.

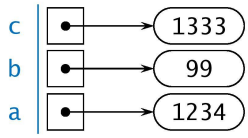
```
a = 1234
```




```
b = 99
```




```
c = a + b
```



انواع داده‌ای

- نوع داده‌ای. یک مجموعه از مقادیر و عملیات قابل انجام بر روی آنها.
- انواع اولیه. عملیات آنها مستقیماً به دستورالعمل‌های ماشین ترجمه می‌شوند.

Data Type	*Values*	*Operations*
boolean	True, False	not, and, or, xor
int	-2^{31} to $2^{31} - 1$	add, subtract, multiply
float	2^{32} possible real values	add, subtract, multiply

- برنامه‌نویسی شی‌گرا. می‌خواهیم برنامه‌هایی بنویسیم که بتوانند هر نوع از داده‌ها را پردازش کنند.
  - رنگ‌ها، تصاویر، رشته‌ها، جریان‌های ورودی، ...
  - اعداد مختلط، بردارها، ماتریس‌ها، چندجمله‌ای‌ها، ...
  - نقطه‌ها، چندضلعی‌ها، ذرات باردار، اجرام آسمانی، ...

مقایسه متدها با توابع

- متد. تابع مرتبط با یک شی خاص.

	متد	تابع
نمونه فراخوانی	x.bit_length()	stdio.writeln(bits)
فراخوانی با	نام متغیر	نام ماجول
پارامترها	یک ارجاع به شی و آرگومان‌ها	آرگومان‌ها
هدف اصلی	دستکاری مقدار شی	محاسبه مقدار برگشتی

In [ ]:
```
'''
   OOP :
       class
       object
       __init__

'''
```

In [ ]:

```
class Test:
    ''' class for add  , mul '''
    def set_var(self,a,b):
        self.a = a
        self.b = b

    def add(self):
        return self.a + self.b

    def mul(self):
        return self.a * self.b

ob1 = Test()

ob1.set_var(2,3)
print(ob1.a)     # 2
print(ob1.b)     # 3

print(ob1.add()) # 2+3=5
print(ob1.mul()) # 2*3=6

print(ob1.__dict__)   # {'a': 2, 'b': 3}

print(ob1.__doc__)    #  class for add  , mul

ob2 = Test()
ob2.set_var(4,7)
print(ob2.a)     # 4
print(ob2.b)     # 7
print(ob2.add()) # 11
print(ob2.mul()) # 28

del ob1
# print(ob1.a)     # name 'ob1' is not defined

del ob2.b
print(ob2.__dict__)   # {'a': 4}
```

In [ ]:
```
print('---- init ----')

class Test:
    def __init__(self,a,b):
        self.a = a
        self.b = b

    def add(self):
        return self.a + self.b

    def mul(self):
        return self.a * self.b

ob1 = Test(2, 3)

print(ob1.a)     # 2
print(ob1.b)     # 3

print(ob1.add()) # 5
print(ob1.mul()) # 6

ob2 = Test(6,1)
x = ob2.add()
print(x)    # 7
```

In [ ]:
```
class Book:
    def __init__(self,author, title):
        self.author = author
        self.title = title

    def info(self):
        print(self.title + ':' + self.author)

x = Book('Golzari','C++')
x.info()                    # C++:Golzari
```

In [ ]:
```
class Student:
    def __init__(self, name, score=None):
        self.d = {}
        self.d['name'] = name
        self.d['score'] = score

    def get_stu(self):
         return self.d

lst = []

ob1 = Student('mahsa',20)
ob2 = Student('ali',17)

lst.append(ob1.get_stu())
lst.append(ob2.get_stu())

print(lst) # [{'name': 'mahsa', 'score': 20}, {'name': 'ali', 'score': 17}]

print(lst[0]) # {'name': 'mahsa', 'score': 20}
```

In [ ]:
```
class Circle:
    def __init__(self, r):
        self.r = r

    def area(self):
        return self.r ** 2 * 3.14

x = Circle(8)
print(x.area())   # 200.96

print(isinstance(x, Circle))  # True
```

In [ ]:
```
d = dict()
print(type(d))  # <class 'dict'>

a = 2
print(type(a)) # <class 'int'>
```

In [ ]:
```
class Counter:
    def __init__(self, x):
        self.x = x

    def up(self):
        self.x += 1

    def down(self):
        self.x -= 1

ob = Counter(4)
print(ob.x)    # 4
ob.up()
ob.up()
print(ob.x)   # 6
ob.down()
print(ob.x)   # 5
```

In [ ]:
```
class C():
    def __init__(self):
        self.s = ''

    def get_string(self):
        self.s = input('input: ')

    def show(self):
        print(self.s.upper())

#ob = C()
#ob.get_string()
#ob.show()
```

In [ ]:
```
class Complex:
    def __init__(self, r, i):
        self.r = r
        self.i = i

x = Complex(2,4)
print(x.r)        # 2
print(x.i)        # 4
```

In [ ]:
```
class Email:
    def f(self):
        self.sent = False

    def g(self):
        self.sent = True

ob = Email()
ob.f()
print(ob.sent)  # False
ob.g()
print(ob.sent)  # True
```

In [ ]:
```
class Machine:
    model = 'peugeot'

    def __init__(self, t):
        self.t = t

m1 = Machine('206')
m2 = Machine('207')

print(m1.model)  # peugeot
print(m2.model)  # peugeot
print(m1.t)      # 206
print(m2.t)      # 207
```

In [ ]:
```
class C:
    lst = []

    def __init__(self, name):
        self.name = name

    def f(self, x):
        self.lst.append(x)

ob1 = C('A')
ob2 = C('B')

ob1.f(1)
ob1.f(2)
ob1.f(3)

print(ob1.lst)   # [1, 2, 3]
print(ob2.lst)   # [1, 2, 3]

ob2.f(4)
print(ob1.lst)   # [1, 2, 3, 4]
print(ob2.lst)   # [1, 2, 3, 4]
```

In [ ]:
```
class C:
    def __init__(self, name):
        self.name = name
        self.lst = []

    def f(self, x):
        self.lst.append(x)

ob1 = C('A')
ob2 = C('B')

ob1.f(1)
ob1.f(2)
ob1.f(3)

print(ob1.lst)   # [1, 2, 3]
print(ob2.lst)   # []

ob2.f(4)
print(ob1.lst)   # [1, 2, 3]
print(ob2.lst)   # [4]
```

In [ ]:
```
class Student:
    stream = 'cse'   # class variable

    def __init__(self, name , score):
        self.name = name      # instance variable
        self.score = score    # instance variable

# objects of student class
s1 = Student('ali', 19)
s2 = Student('sara', 18)

print(s1.name)    # ali
print(s2.name)    # sara

print(s1.stream)  #cse
print(s2.stream)  #cse

print(Student.stream)   # cse
# print(Student.name)   # AttributeError
```

In [ ]:
```
class Dog:
    animal = 'dog'

    def __init__(self, b , c):
        self.b = b
        self.c = c

d1 = Dog('pug' , 'brown')
d2 = Dog('bulldog' , 'black')

print(d1.animal)   # dog
print(d2.animal)   # dog

print(d1.c)   # brown
print(d2.c)   # black
```

In [ ]:
```
# private

class Test:
    def __init__(self, a, b):
        self.a = a     # public
        self.__b = b   # private

    def f(self):
        self.__b += 1
        print(self.__b)

ob = Test(1, 2)
print(ob.a)          # 1
#print(ob.__b)       # AttributeError

ob.f()               # 3

print(ob.__dict__)   # {'a': 1, '_Test__b': 3}
print(ob._Test__b)   # 3     name mangling

ob._Test__b  = 8
print(ob._Test__b)   # 8
```

In [ ]:
```
class S:
    def __init__(self, x):
        self.__a = x

    def f(self):
        print(self.__a, end=' ')
        self.__a += 1
        return(self.__a)

    def g(self, m):
       print( m + self.f())

ob = S(1)
print(ob.__dict__)   # {'_S__a': 1}
print(ob.f())        # 1 2
ob.g(5)              # 2 8
```

In [ ]:
```
class AB:
    __x = 3

    def show(self):
        print(self.__x)


ob = AB()
ob.show()        # 3
print(ob._AB__x) # 3
```

In [ ]:
```
# private method

class ABC:
    def __f(self):
        print(1)

    def g(self):
        return (self.__f())

ob = ABC()
ob.g()      # 1
# ob.__f()  # AttributeError

ob._ABC__f() # 1
```

دانشگاه شهید مدنی آذربایجان
برنامه نویسی پیشرفته با پایتون
امین گلزاری اسکوئی
۱۴۰۰-۱۴۰۱

Codes and Projects (click here) (https://github.com/Amin-Golzari-Oskouei/Python-Programming-Course-Advanced-2021) slides and videos (click here) (https://drive.google.com/drive/folders/1Dx3v7fD1QBWL-MNP2hd7ilxaRbeALkkA)