In [ ]:
```
# inhertiance
```

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

    def show(self):
        print('name:' , self.name)



class Student(Person):
    def __init__(self, name, score):
        #Person.__init__(self, name)
        super().__init__(name)
        self.score = score

    def welcom(self):
        print('welcom', self.name)

stu = Student('ali',20)
stu.welcom()            # welcom ali
stu.show()              # name: ali
```

In [ ]:
```
# parent class
class Person:
    def __init__(self, name, id):
        self.name = name
        self.id = id

    def display(self):
        print(self.name)


# child class
class Emplyee(Person):
    def __init__(self, name , id, salary, post):
        Person.__init__(self, name, id)
        self.salary = salary
        self.post = post

emp1 = Emplyee('sara', 1234, 5000000 , 'modir')
emp2 = Emplyee('omid', 1678, 2000000 , 'secreter')

emp1.display()       # sara
emp2.display()       # omid
```

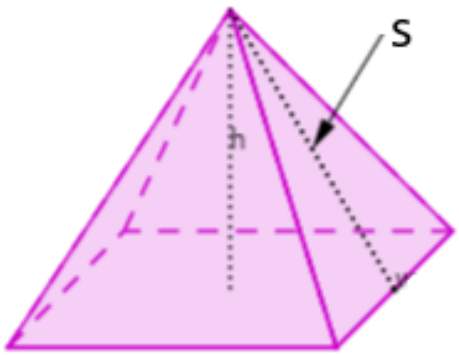
In [ ]:

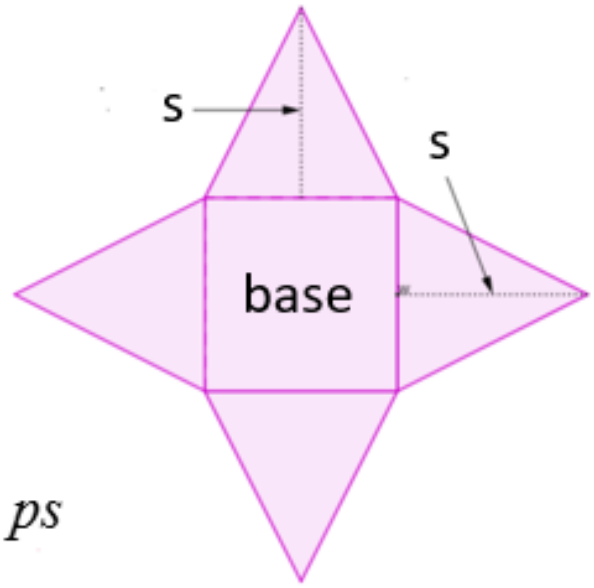
```
class Rect:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def area(self):
        return self.x * self.y


class Square(Rect):
    def __init__(self, z):
        #Rect.__init__(self, x=z,y=z)
        super().__init__(x=z,y=z)

r = Rect(2, 3)
print(r.area())        # 6

s = Square(5)
print(s.area())       # 25


# __mro__   :  Method resolution order

print(Square.__mro__)
# (<class '__main__.Square'>, <class '__main__.Rect'>, <class 'object'>)

print(Rect.__mro__)
# (<class '__main__.Rect'>, <class 'object'>)
```

In [ ]:

```
# multiple inheritance

class B1:
    def __init__(self, x):
        self.x = x
        print(x)

class B2:
    def __init__(self, y):
        self.y = y
        print(y)

class D(B1, B2):
    def __init__(self, z):
        self.z = z
        print(z)
        B1.__init__(self, 2)
        B2.__init__(self, 3)

d = D(1)    # 1  2  3

print(D.__mro__)
```

Surface Area of Pyramid





Surface Area = $A + \frac{1}{2}ps$

A = Area of base

p = perimeter of base

s = slant height

In [ ]:

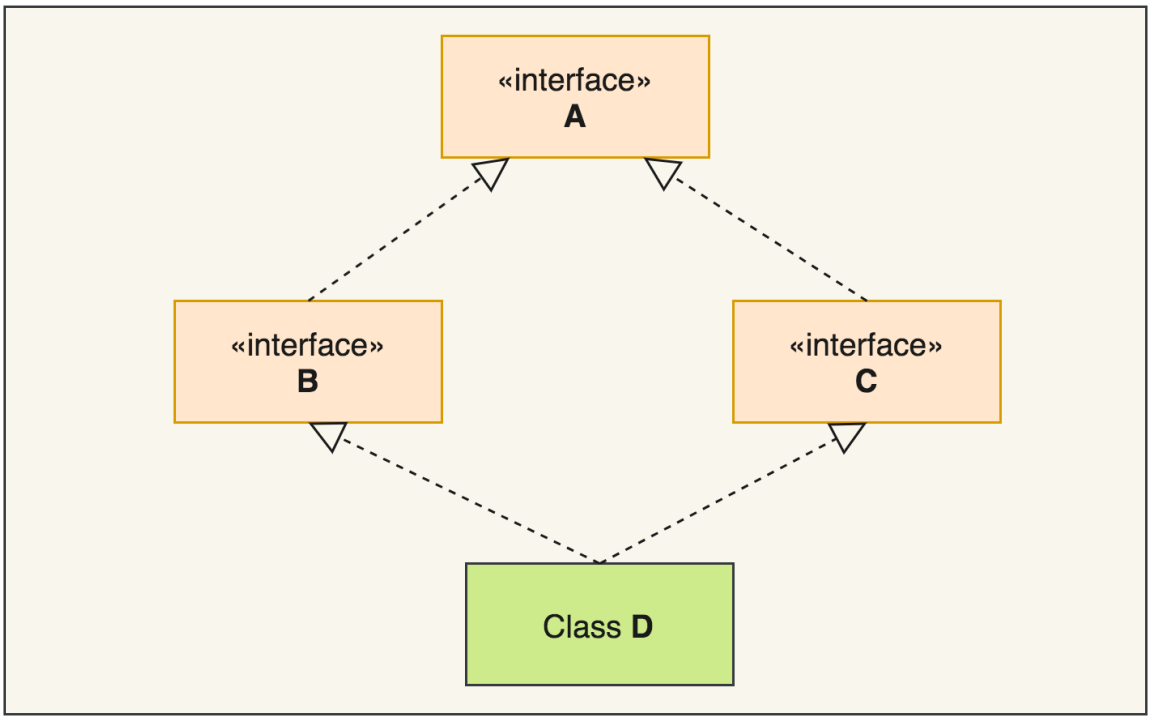
```
class Square:
    def __init__(self, x):
        self.x = x

    def area(self):
        return self.x * self.x

    def perimeter(self):
         return 4 * self.x

class Triangle:
     def __init__(self, y, z):
         self.y = y
         self.z = z

     def area(self):
         return 0.5 * self.y * self.z



class Pyramid(Square, Triangle ):
    def __init__(self, b, s ):
        self.b = b
        self.s = s
        Square.__init__(self,x=b)
        Triangle.__init__(self,y=b,z=s)

    def area(self):
        a = Square.area(self)
        p = Square.perimeter(self)
        return a + 0.5 * p * self.s

t = Triangle(3, 4)
print(t.area())    # 6.0

p = Pyramid(2, 5)
print(p.area())    # 24.0    :  2*2 + 0.5 * 8 * 5
```

In [ ]:

```
# multilevel inheritance
class A:
    def __init__(self, name):
        self.name= name

    def getname(self):
        return self.name

class B(A):
    def __init__(self, name, age):
        A.__init__(self,name)
        self.age = age

    def getage(self):
        return self.age

class C(B):
    def __init__(self,name, age, score):
        B.__init__(self,name, age)
        self.score = score

    def getscore(self):
        return self.score

ob = C('ali', 30, 18)
print(ob.name)        # ali
print(ob.getname())   # ali
print(ob.getscore())  # 18
```

In [ ]:

```
class Employee:
      def __init__(self, id, name):
          self.id = id
          self.name = name


class HE(Employee):
      def __init__(self, id, name, hw, hr):
          super().__init__(id, name)
          self.hw = hw
          self.hr = hr

      def h(self):
          return self.hw * self.hr

class SE(Employee):
      def __init__(self, id, name, s):
          super().__init__(id, name)
          self.s = s

      def h(self):
          return self.s

class CE(SE):
      def __init__(self, id, name, s, c):
          super().__init__(id, name , s)
          self.c = c

      def h(self):
          return self.s + self.c

class P:
    def payroll(self,lst):
        for i in lst:
            print(f'{i.id}:{i.name} = {i.h()}')

ob1 = HE(1,'sara', 4, 100000)
ob2 = SE(2,'ali',5000000)
ob3 = CE(3,'taha',3000000,500000)

ob = P()
ob.payroll([ob1, ob2, ob3])
'''
1:sara = 400000
2:ali = 5000000
3:taha = 3500000
'''
```



In [ ]:

```
# dimaond problem

class A:
   def f(self):
       print('A')

class B(A):
   def f(self):
       print('B')

class C(A):
   def f(self):
       print('C')

class D(B, C):
    pass

d = D()
d.f()   # B
```

In [ ]:
```
class A:
   def f(self):
       print('A')

class B(A):
   def f(self):
       print('B')

class C(A):
   def f(self):
       print('C')

class D(C , B):
    pass

d = D()
d.f()   # C
```

In [ ]:
```
class B:
    def __init__(self, x, y):
        self._a = x      # protected
        self.__b = y     # private

    def f(self):
        print(self._a)
        print(self.__b)

class D(B):
    def h(self):
        print(self._a)
        #print(self.__b)  error

d = D(1, 2)
d.h()   # 1
d.f()   # 1 2
```

In [1]:
```
class B:
    def __f(self):
        return 'A'

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

class D(B):
    def __f(self):
        return 'B'

d = D()
d.g()    # A
```

```
A
```

In [ ]:

```python
class B:
    def _f(self):
        return 'A'

    def g(self):
        print(self._f())

class D(B):
    def _f(self):
        return 'B'

d = D()
d.g()    # B
```

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-MNP2hd7iIxaRbeALkkA)