The Abdus Salam International Centre for Theoretical Physics

HOW PYTHON DOES OO

Ali Farnudi

CLASSES

CLASSES

```
class Person:

    def __init__(self, first, last):
        self.firstname = first
        self.lastname = last

    def name(self):
        return self.firstname + " " + self.lastname
```

initialisation (constructor)

CLASSES

```
class Person:

    def __init__(self, first, last):
        self.firstname = first
        self.lastname = last

    def name(self):
        return self.firstname + " " + self.lastname
```

member variables (attributes)

CLASSES

```
class Person:

    def __init__(self, first, last):
        self.firstname = first
        self.lastname = last

    def name(self):
        return self.firstname + " " + self.lastname
```

member function (method)

CLASSES

```
class Person:

    def __init__(self, first, last, age=0):
        self.firstname = first
        self.lastname = last
        self.age = age

    def name(self):
        return self.firstname + " " + self.lastname

    def birthday(self):
        self.age += 1
```

CLASSES

```
class Person:

    def __init__(self, first, last, age=0):
        self.firstname = first
        self.lastname = last
        self.age = age

    def name(self):
        return self.firstname + " " + self.lastname

    def birthday(self):
        self.age += 1
```

```
per_1=Person('Ali','Farnudi')

print(per_1.name())
```

```
Ali Farnudi
```

CLASSES

```
class Person:

    def __init__(self, first, last, age=0):
        self.firstname = first
        self.lastname = last
        self.age = age

    def name(self):
        return f“{self.firstname} {self.lastname}”

    def birthday(self):
        self.age += 1
```

CLASSES - OVERLOADING

```
class Person:

    def __init__(self, first, last, age=0):
        self.firstname = first
        self.lastname = last
        self.age = age

    def __str__(self):
        return f“{self.firstname} {self.lastname}”

    def birthday(self):
        self.age += 1

per_1=Person(‘Ali’,'Farnudi')

print(per_1)
```

```
Ali Farnudi
```

CLASSES - OVERLOADING

Operator	Expression	Internally
The string representation	str	__str__(self)
The number of elements	len	__len__(self)
Check membership	in	__contains__(self, value)
Index operator	[index]	__getitem__(self, index)
Addition	+	__add__(self, value)
Subtraction	-	__sub__(self, value)
Multiplication	*	__mul__(self, value)
Power	**	__pow__(self, value)
Equal to	==	__eq__(self, value)
Greater than	>	__gt__(self, value)
Bitwise Right Shift	>>	__rshift__(self, value)
Bitwise NOT	~	__invert__(self)

CLASSES

```
class Person:

    def __init__(self, first, last, age=0):
        self.firstname = first
        self.lastname = last
        self.age = age

    def __str__(self):
        return f“{self.firstname} {self.lastname}, age={self.age}”

    def birthday(self):
        self.age += 1
```

```
per_1=Person(‘Ali’,'Farnudi')

per_1.birthday()
per_1.birthday()

print(per_1)
```

```
Ali Farnudi, age=2
```

CLASSES

```
class Person:

    def __init__(self, first, last, age=0):
        self.firstname = first
        self.lastname = last
        self.age = age

    def __str__(self):
        return f"{self.firstname} {self.lastname}, age={self.age}"

    def birthday(self):
        self.age += 1
```

```
per_1=Person('Ali','Farnudi', 29)

per_1.birthday()
per_1.birthday()

print(per_1)
```

```
Ali Farnudi, age=31
```

CLASSES - ACCESSOR METHODS

```
class Point:

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
```

```
>>> point_1=Point(2,3)
>>> point_1.x, point_1.y
2, 3
```

CLASSES - ACCESSOR METHODS

```
from math import sort, atan2
class Point:

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
        self.r = sqrt(x**2+y**2)
        self.phi = atan2(y,x)
```

```
>>> point_1=Point(3,4)
>>> point_1.x, point_1.y
3, 4
>>> point_1.r, point_1.phi
5.0, 0.9272952
```

warning

```
>>> point_1.r=10
```

CLASSES - ACCESSOR METHODS

```
from math import sort, atan2
class Point:

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y



    def r(self):
        return sqrt(x**2+y**2)


    def phi(self):
        return atan2(y,x)
```

```
>>> point_1=Point(3,4)
>>> point_1.x, point_1.y
3, 4
>>> point_1.r(), point_1.phi()
5.0, 0.9272952
```

warning

```
>>> point_1.r=30
```

CLASSES - ACCESSOR METHODS

- property decorators

```
from math import sort, atan2
class Point:

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y


   @property
    def r(self):
        return sqrt(x**2+y**2)

   @property
    def phi(self):
        return atan2(y,x)
```

```
>>> point_1=Point(3,4)
>>> point_1.x, point_1.y
3, 4
>>> point_1.r, point_1.phi
5.0, 0.9272952
```

```
>>> point_1.r=10
```

```
Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

AttributeError: can't set attribute
```

CLASSES - ACCESSOR METHODS

- property decorators with assignment

```
from math import sort, atan2
class Point:

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y


   @property
    def r(self):
        return sqrt(x**2+y**2)

   @r.setter
   def r(self, r_new):
       r_old = self.r
       scale = r_new/r_old
       self.x *= scale
       self.y *= scale

   @property
    def phi(self):
        return atan2(y,x)
```

```
>>> point_1=Point(3,4)
>>> point_1.x, point_1.y
3, 4
>>> point_1.r, point_1.phi
5.0, 0.9272952

>>> point_1.r=10
>>> point_1.r, point_1.phi
10.0, 0.9272952
>>> point_1.x, point_1.y
6.0, 8.0
```

The Abdus Salam International Centre for Theoretical Physics

Inheritance

```
class Foo(object):
    def hello(self):
        print “Hello! Foo here.”

    def bye(self):
        print “Bye bye from Foo!”

class Bar(Foo):
    def hello(self):
        print “Hello! Bar here.”
```

```
>>> f = Foo()
>>> f.hello()
Hello! Foo here.
>>> f.bye()
Bye bye from Foo!
>>>
>>> b = Bar()
>>> b.hello()
Hello! Bar here.
>>> b.bye()
Bye bye from Foo!
```

THE END