

A+ Computer Science

PYTHON CLASSES & METHODS

A Simple Method

Methods allow you to reuse code

```
sayLotsOStuff()  
sayLotsOStuff()
```

```
def sayLotsOStuff():  
    print("Lots Of")  
    print("Stuff")
```

Output

```
Lots Of  
Stuff  
Lots Of  
Stuff
```

A Simple Method

All methods require def, a name, parenthesis and a colon

Each method name in a file must be unique

```
def sayLotsOStuff():  
    print("Lots Of")  
    print("Stuff")
```

A Simple Method

All code in the method needs to be indented the same amount –4 spaces

```
def sayLotsOStuff():  
    print("Lots Of")  
    print("Stuff")
```

Bad Method

```
def sayLotsOStuff():  
print("Lots Of")  
print("Stuff")
```

Bad Method

```
def sayLotsOStuff():  
    print("Lots Of")  
    print("Stuff")
```

pass

pass is a statement that does nothing. It is often used as placeholder for code that needs to be added.

```
def sayNothing():  
    pass
```

Method Parameters

Parameters allow you to pass information to a method

```
def sayThree(words):  
    print(words)  
    print(words)  
    print(words)
```

Method Parameters

Parameters allow you to pass information to a method

```
sayThree("Echo")
```

```
def sayThree(words):  
    print(words)  
    print(words)  
    print(words)
```

Output

```
Echo  
Echo  
Echo
```

Return Methods

Sometimes we want to get information from the method using the keyword return

```
print( cubeIt(4) )    def cubeIt (num):  
                      return num * num * num
```

Output

64

methods.py



Objects & Classes

Sometimes we want to store data together

```
student1Name = "Sarah"
```

```
student1Grade = 12
```

```
student1Class = "Math"
```

```
student2Name = "Tim"
```

```
student2Grade = 11
```

```
student2Class = "Comp Sci"
```

Objects & Classes

Classes make storing that data easier

```
class Student:
```

```
    def __init__(self, theName, theGrade, theClass):
```

```
        self.name = theName
```

```
        self.grade = theGrade
```

```
        self.className = theClass
```

```
=====
```

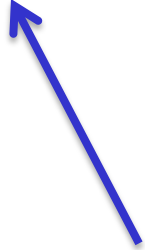
```
student1 = Student("Sarah", 12, "Math")
```

```
student2 = Student("Tim", 11, "Comp Sci")
```

Objects & Classes

Start with the word class, then the name of the class followed by a colon.

class Student:



**Classes start with
a capital letter**

Objects & Classes

Constructors take the data given and stores it together

They always use the method name `__init__`

```
class Student:
```

```
    def __init__(self, theName, theGrade, theClass)  
        self.name = theName  
        self.grade = theGrade  
        self.className = theClass
```

Objects & Classes

The word `init` is has two underscores on each side

```
def init (self, theName, theGrade, theClass)
```

1 2 1 2

Objects & Classes

Constructors are called with the name of the class

```
class Student:
```

```
    def __init__(self, theName, theGrade, theClass)  
        self.name = theName  
        self.grade = theGrade  
        self.className = theClass
```

```
myStudent = Student("Lee", 10, "Science")
```

Objects & Classes

Class methods must be defined in the class and indented

```
class Student:  
    def setName(self, newName)  
        self.name = newName  
  
    def getName(self)  
        return self.name
```


Objects & Classes

All class methods require the parameter `self`.


To use or change the data stored for a class, and use methods in the class you must use the keyword `self`

```
def setName(self, newName)
    self.name = newName
```

Objects & Classes

The object name gets passed to the method as self

```
myStudent.setName("Sam")  
  
def setName(self, newName)  
    self.name = newName
```

A diagram consisting of two blue arrows. The first arrow starts at the word 'myStudent' in the first line of code and points to the parameter 'self' in the second line. The second arrow starts at the string '"Sam"' in the first line and points to the parameter 'newName' in the second line.

Objects & Classes

self.x is not the same as x

```
def setName(self, newName):  
    name = newName
```

```
theStudent = Student("Amber", 12, "Math")  
theStudent.setName("Victoria")  
print theStudent.getName()
```

Output
Amber

Objects & Classes

All class methods require an object to be created.

```
myStudent = Student("Lee", 10, "Science")
```

The object can then call the method

```
myStudent.setName("Sam")
```

Objects & Classes

Class methods are separated in to accessors (which get data) and mutators (which change data)

mutator

```
def setName(self, newName)
    self.name = newName
```

accessor

```
def getName(self)
    return self.name
```

classes.py



Importing Classes

Methods and classes are often stored in other files. To use them we need to import them

```
from student import *
```

```
myStudent = student("Xavier", 9, "History")
```

importclass.py
student.py

Using Libraries

There's already a ton of code written called libraries that we can use by importing. One that we use often is random

```
import random
```

```
print random.randint(2, 22)
```

Using Libraries

**To use the code we have to use
`filename.methodName()`**

```
import random
```

```
print random.randint(2, 22)
```

Random

frequently used methods

Name	Use
<code>random.randint(a, b)</code>	Get a random integer between a and b
<code>random.random()</code>	Get a random number between 0.0 and 1.0

Work on Programs!

Crank

Some Code!

A+ Computer Science

PYTHON CLASSES & METHODS