

PYTHON: CLASSES / VARIABLE SCOPE

9.10.2018

ANACONDA INSTALL?!?



TYPES

- * Everything in python has a **type**

- *

```
>>> type(["w","h","a","t"])  
<class 'list'>
```

- *

```
>>> type(5)  
<class 'int'>
```

- *

```
>>> type("lol")  
<class 'str'>
```

- *

```
>>> def func(): pass  
>>> type(func)  
<class 'function'>
```

TYPES

- * A **type** is a blueprint for an object
- * It can have functions (aka methods)
(*like .split() with strings*)
- * It can have variables (aka class attributes)
- * Each **instance** of a type can also have its own variables (aka data attributes)

TYPES

- * A type can be called like a function:
`>>> some_instance = some_type()`
- * This is called **instantiation**, since it creates a new **instance** of the type
- * *You've already seen this! (dict(...))*

CLASSES

- * You can create your own types!
- * But when you do, they're called **classes** for some reason!
- * What's the difference between classes and types? There is none! There are just two names to slightly confuse and infuriate you!

CLASSES

- * You may never need to write a class
- * But you will use classes other people have written ***ALL THE TIME***
- * So you need to learn about classes

CLASSES

```
* class Line(object):  
    def __init__(self, m, b):  
        self.m = m  
        self.b = b  
  
    def compute(self, x):  
        return self.m * x + self.b
```


CLASSES

class name

superclass

```
* class Line(object):  
    def __init__(self, m, b):  
        self.m = m  
        self.b = b  
  
    def compute(self, x):  
        return self.m * x + self.b
```

initializer

method

CLASSES

```
* >>> my_line = Line(3.5, 0)
>>> print(my_line.compute(12))
>>> print(my_line.m)
```

INHERITANCE

```
* class StupidLine(Line):  
    def compute(self, x):  
        return self.b * x + self.m  
  
* >>> my_stupid_line = StupidLine(3.5, 0)  
  >>> my_stupid_line.compute(12)
```

VARIABLE SCOPE

```
* s = "some dumb string"
  def func():
      print(s)
  func()
```

```
* def func():
    s = "some dumb string"
  func()
  print(s)
```

VARIABLE SCOPE

```
* s = "some dumb string"
def func():
    print(s)
func()
```

ACCEPTABLE

```
* def func():
    s = "some dumb string"
func()
print(s)
```

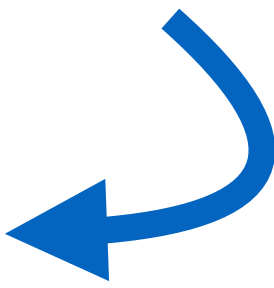
GARBAGE

VARIABLE SCOPE

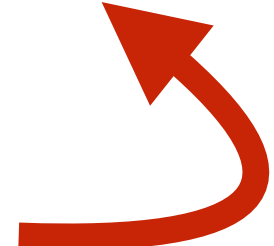
- * The **scope** of a variable is the set of places in your code where that variable is available
- * Scope propagates “inward”, not “outward”:
 - * Vars defined outside a function are available inside, but
 - * Vars defined inside a function are NOT available outside

VARIABLE SCOPE

```
* s = "some dumb string"
  def func():
      print(s)
  func()
```



```
* def func():
    s = "some dumb string"
  func()
  print(s)
```



VARIABLE SCOPE

- * What defines a scope?
 - * Modules (files)
 - * Functions
 - * Classes (kinda but not totally!)

VARIABLE SCOPE

- * How do we get variables from an “inner” scope to an “outer” one?
- * **Modules & Classes:** using the dot .
(*e.g.* blah.stuff)
- * **Functions:** return
 - * *what if i didn't return it? lol it's gone*

BYE