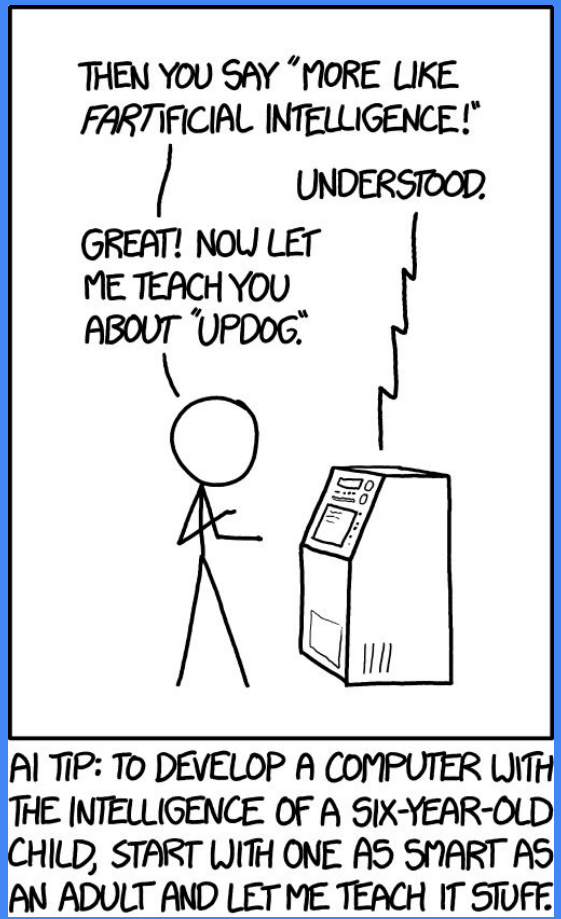
ITSE-1402 Intermediate Python

Class 4: Classes and Functions/Methods

June 13, 2017



Lambda calculus? More like SHAMbda calculus, amirite?

https://xkcd.com/1696/

Methods in Classes

- Methods are functions, but within classes
- There must be a first argument of self in all of method definitions which gets bound to the calling instance

```
class student:
        def __init__(self, x, y):
                self.name = x
                self.age = y
        def get_name(self):
                return self.name
        def get_age(self):
                return self.age
```

Self

- In __init__ self refers to the object currently being created. In other class methods, it refers to the instance whose method was called.
- The first argument of every method is a reference to the current instance of the class.
- Although you must specify self explicitly when defining the method, you don’t include it when calling the method. Python passes it for you automatically.

Instantiating Objects

- __init__ is a constructor for the class and usually is used for defining initial values of variables.
- As seen in the previous example, the arguments passed through the class instantiation are given to __init__.

```
class student:
        def __init__(self, x, y):
                self.name = x
                self.age = y
        def get_name(self):
                return self.name
        def get_age(self):
                return self.age

student1 = student(“Joe”,25)
```

Special Methods

- There are many methods in classes and most have a default action if not defined.
- As an example, the method __repr__ exists for all classes, and you can redefine it. The definition of this method specifies how to turn an instance of the class into a string.

```
class student:
        def __repr__(self):
                return “My name is ” + self.full_name

>>> f = student(“Joe Dirt”, 25)
>>> print f
>>> “My name is Joe Dirt”
```

Special Methods

- There are too many to name them all, but here are a few:

  __init__ : The constructor for the class

  __cmp__ : Define how == works for class

  __len__ : Define how len(obj) works

  __getitem__ : Define how the object is treated when used as a list[1]

- A full list may be found here:

  https://docs.python.org/3/reference/datamodel.html

Special Attributes

- These are some attributes that exist for all classes

__doc__ : Variable for documentation string for class

__class__ : Variable which gives you a reference to the class from any instance of it

__name__ : Variable for class name

__module__ : The name of the module the function was defined in, or None if unavailable.

```
class student:
        """This is a docstring"""

>>> f = student("Joe Dirt", 25)
>>> print f.__doc__
>>> "This is a docstring"
```

Private Data and Methods

- Any attribute/method with 2 leading under-scores in its name (but none at the end) is private and can’t be accessed outside of class
- Any attribute/method with two underscores at the beginning and the end are for built-in methods or attributes for the class
- Note: There is no ‘protected’ status in Python; so, subclasses would be unable to access these private data either.

Data Attributes / Class Attributes

- Data attributes are created and initialized by an __init__() method. These attributes are per instance:

```
class teacher:
        def __init__(self,x):
                self.name = x
        def print_name(self):
                print self.name
```

- Class attributes are created and initialized anywhere in the class. These attributes are per class:

```
class teacher:
        total = 0
        def __init__(self,x):
                teacher.total += 1
                self.name = x
```