PYTHON DONE RIGHT

EPIC WEEK OF CRASH COURSES 20112

ORIGINALLY GIVEN 11/28/11 BY DEREK ERDMANN

TOPICS

- **Python Philosophy**
- **Basics (for you old timers)**
- **Object-Oriented Python**

WHAT IS PYTHON?

- **Interpreted**
- **Dynamically typed**
- **Object-oriented**
- **Can be used for functional or procedural style**
- **Portable**
- **Has a clear, readable syntax**

PYTHON PHILOSOPHY

- **Beautiful is better than ugly**
- **Explicit is better than implicit**
- **Simple is better than complex**
- **Complex is better than complicated**
- **Readability counts**
- **Special cases aren't special enough to break the rules**
  - Although practicality beats purity

PYTHON PHILOSOPHY

- **In the face of ambiguity, refuse the temptation to guess**
- **There should be one - and preferable only one - obvious way to do it**
- **Now is better than never**
  - Although never is often better than *right* now
- **If the implementation is hard to explain, it's a bad idea**
- **If the implementation is easy to explain, it may be a good idea**

ENOUGH META…

PYTHON BASICS

HELLO WORLD

Open IDLE or run `python` in a shell

```
print( “Hello, world!” )
```

Enter this directly into the file `hello.py` and run

`python hello.py` in your shell, and it does the same thing!

FUNCTIONS AND CONDITIONS

```
def getRandomNumber():
        return 4


randint = getRandomNumber()
if 3 < randint < 5:
    print( "Determined by a fair dice roll" )
else:
    print( "Probably isn't random" )
```

LOOPING

```
int i = 0
while i < 10:
    print( "Derek ate " + i + " pies." )
    i += 1
```

This is identical to:

```
    for i in range( 10 ):
        print( "Derek ate " + i + " pies." )
```

LISTS

```
numbers = []
numbers = [ 1, 2, 3, 4 ]
numbers = [ i for i in range( 4 ) ]

for i in range( 4 ):
    print( numbers[i] )

for i in numbers:
    print( i )
```

OBJECT-ORIENTED PYTHON

CLASSES AND OBJECTS

Classes

- **Templates for many objects**
- **Define behaviors and state**

Objects

- **Instances of a class**
- **Can perform actions and hold state**
- **Actual "things" that are built from the templates**

In object-oriented systems, the entire program is a bunch of objects that pass around information and act on each other

PYTHON CLASSES IN CS LAND

```
class Animal:
    __slots__ = ( "name", "age" )

def mkAnimal( name, age )
    animal = Animal()
    animal.name = name
    animal.age = age
    return animal

myanimal = mkAnimal( "George", 42 )
print( myanimal.name )
```

- It works, but isn't sensible

REAL CLASSES

```
class Animal:
      def __init__( self, name, age ):
          self.name = name
          self.age = age

  myanimal = Animal( "George", 42 )
  print( myanimal.name )
```

- `__init__` is the *constructor*
- `self` refers to the current object

MORE ABOUT CLASSES

```
class Animal:
    def __init__( self, name, age ):
        self.name = name
        self.age = age

    def speak( self ):
        print( "My name is " + name + "!" )

myanimal = Animal( "George", 42 )
myanimal.speak()
```

INHERITANCE

A dog is an animal...

```
class Dog( Animal ):
    def __init__( self, name, age, breed ):
        super( Animal, self ).__init__( name, age )
        self.breed = breed

    def speak( self ):
        print( "Woof!" )
```

ACCESSING ATTRIBUTES

- Everything is public in Python
- The convention is that "private" class members are prefixed with "_"

```
class Animal:
    def __init__( self, name, age ):
        self._name = name
        self._age = age
```

- This doesn't actually prevent access to `_name` or `_age`

PROPERTIES

- In Java, private members are accessed with get/set methods
- In Python, use a public property until it needs to be private
- When you need a getter or setter, use decorators

PROPERTIES

```
class Animal:
    def __init__( self, name, age ):
        self._name = name
        self._age = age

    @property
    def name( self ):
        return _name

myanimal = Animal( "George", 42 )
myanimal.name == "George"
```

If you only use `@property`, "name" is read-only

MORE PROPERTIES

```
class Animal:
    def __init__( self, name, age ):
        self._name = name
        self._age = age

    @property
    def name( self ):
        return self._name

    @name.setter
    def name( self, name ):
        self._name = name

myanimal = Animal( "George", 42 )
myanimal.name = "Jeremy"
```

WITH GREAT POWER…

For those of you in CS 242:

- **In class, do it how your professors want you to do it**

During RapDev and everywhere else, do it right

EPIC WEEK OF CRASH COURSES

- 6 PM Tuesday - Beginning Java
- 6 PM Wednesday - Learning to REST
- 6 PM Thursday - Playing with Swings
- 2 PM Friday - Git and GitHub