PYTHON CRASH COURSE

http://eddiema.ca/docs/pycrashcourse/

LECTURE 2

```
def tell_me_if_i_am_crazy(i_see_a_python):
    if i_see_a_python:
        print 'Quite crazy.'
    else:
        print 'Less crazy.'
```

EDDIE MA

JANUARY 2010

http://eddiema.ca

Agenda

Stuff we want to cover today (30 minutes!)

- Python Scripts
  - Files that you run
  - Modules that you import
- Logical Program Flow
  - Conditionals
  - Loops
- Code Organization
  - Functions
- Collections
  - Lists and Tuples
- Sneak Preview
  - Example GUI Script
- Object Orientation
  - Moved to next week
- Dictionaries and Sets
  - Moved to next week

Windows Users!

When you get home...

Check out this link:

http://showmedo.com/videos/video?name=960000

It shows you how to run python from the console just by typing `python`.

You need to change the “path” “environment variable” to do it.

Python Scripts

things.py

- Python scripts are plain text (ASCII) files that have the extension `.py`
- When they contain no particular structure, the interpreter just reads them top to bottom.
- Try running main.py included for this week.
- Use `python main.py`

Python Scripts

things.py

- Python scripts may contain things that can be imported
  - The file `pointless.py` contains a function `point()` that can be imported
- The `import` keyword offers a few options
  - `import some_module`
    - Try it: in the python console, type “import pointless”
    - Now type pointless.point()
  - `import some_module as some_other_name`
    - Used to give modules short names, type “import pointless as p”
    - Now type p.point()
  - `from some_module import some_things`
    - Used to import components of modules, type “from some_module import *”
    - Now type point()

Fire up your favourite text editor

Let's write a short script

- This script will simulate a six-sided die roll whenever it is called

- Type the following two lines into a new file
  - Call it `dice.py`

```
from random import randint
print randint(1, 6)
```

- Run your script.
  - `python dice.py`
- Again.
- And again.

The `if` Statement

Your die loves the number three and likes even numbers.

- Make your die exclaim excitement whenever it rolls the number three and even numbers.

```
from random import randint
face = randint(1, 6)
~~print randint(1, 6)~~
if face == 3:
    print face, ": awesome"
elif face % 2 == 0:
    #hint-- even numbers
    print face, ": yay"
else:
    print face, ": boo"
```

- Run your script X10

The `while` Statement

Your die won't leave the casino until it rolls a six.

- Make your die stubbornly roll until it gets a six
- The `While` keyword is used for loops
- The loop repeats over and over as long as the condition `face != 6` is true
- `face != 6` means “face is not six”
- The opposite operator is “==” which states equality

```
from random import randint
face = None
while face != 6:
    face = randint(1, 6)
    if face == 3:
        print face, ": awesome"
    elif face % 2 == 0:
        #hint-- even numbers
        print face, ": yay"
    else:
        print face, ": boo"
```

- Run your script X5

Make a function that sums two dice

Yes, beginner's stuff still!

- Create a new file or replace your current one.

```
from random import randint

def rolltwo():

  return randint(1,6) + randint(1, 6)
```

- Run it a few times
- Functions have a few parts you should remember
  - `def` keyword, function name, parentheses for arguments, `return` keyword, return statement

Lists!

Finally! Something new!

- Lists are used to keep sequences of data-- they are similar to arrays
- Their contents and their properties can be altered in memory
- Let's save two thousand paired dice roll's worth of results in a list
- Add the following code somewhere below the rolltwo() function

```
dicerolls = []
#the above line means “create an empty list”
#alternatively, you can say dicerolls = list()
nrolls = 0
while nrolls < 2000:
  dicerolls.append(rolltwo())
  nrolls += 1
#let's look at the list
print dicerolls
```

`for` Loops

`while` loops are great but `for` loops work better for collections

- Try this
  - insert this code somewhere lower in your script...

```
for roll in dicerolls:
    print roll
```

- What happened?
  - The for loop iterates through all of the items in a collection in sequential order
  - Each time it repeats, the variable `roll` is updated (rebound) to be the current item in the list

Tuples

are just immutable lists

- Tuples are just immutable lists
- They are exactly the same as lists, except they can't be modified in memory
- Once they're born, they're the same until they die
- You can use tuples in return statements if you want to tie a couple of values together
- Tuples can specified with a pair of parentheses or the constructor "`tuple()`"

```
exampleTuple = (average, stddev, nsamples)
```

Getting a Specific Element

From an ordered collection

- In Python, all indices start at zero.
- In your dicerolls list, to get the very first result, you would do this:

```
firstResult = dicerolls[0]
```

- To get the tenth result, you would do this:

```
someResult = dicerolls[9]
```

- This is often confusing, so just stick with calling everything by their index-- “dicerolls[9] gets the element at the nineth position”.

Getting a Specific Element

BACKWARDS!

- Python has an additional shorthand of counting from the last index using negative integers.
- Try this code in a new Python console.

```
sentence = "Always brush your teeth."
sentence[-1]
sentence[20]
sentence[18]
sentence[-6]
```

Slicing

Getting subsequences

- Sequentially ordered items can also be sliced to produce subsequences.
- Slicing notation uses the colon notation

```
sequence[x:y]
```

- Where 'x' is the start index, and 'y' is the index AFTER the end index.

Slicing Examples...

Try these in your Python console...

```
sentence = "Always brush your teeth."
print sentence[0:6]
print sentence[7:12]
print sentence[-6:-1]
print sentence[:6]
print sentence[7:]
```