Lecture 2
Modules & Functions

Recap last lecture

- Debugging (and types of errors)
- Variables (types and values)
- Expressions
- Questions?

Morning Routines

Alarm clock goes off
Hit snooze
Alarm clock goes off
Hit snooze
Alarm clock goes off
Get out of bed
Do yoga
Take a shower
Get dressed
Have breakfast
Cycle to the uni

Morning Routines

Alarm clock goes off
Hit snooze
Alarm clock goes off
Hit snooze
Alarm clock goes off
Get out of bed
Do yoga
Take a shower
Get dressed
Have breakfast
Cycle to the uni

Do Yoga

Do this 4 times:

- Downward dog
- Plank
- Cobra

Do this two times:

- Warrior 2
- triangle
- Warrior 2
- Warrior 2 reversed

Cobra

Lay on your belly

Put your hands next to your shoulders palms down

Press the back of your feet in the ground

Lift your upper body without using your hands

Lift it a bit higher using your hands

Make sure you keep your elbows close to your body

Modules & Functions

Modules

Modules

`import`

  Import a module

`from` *`module`* **`import`** *`function`*

  Imports a specific function from a specific module (use with care)

`dir()`

  Returns a list of all functions in a given module

Some modules

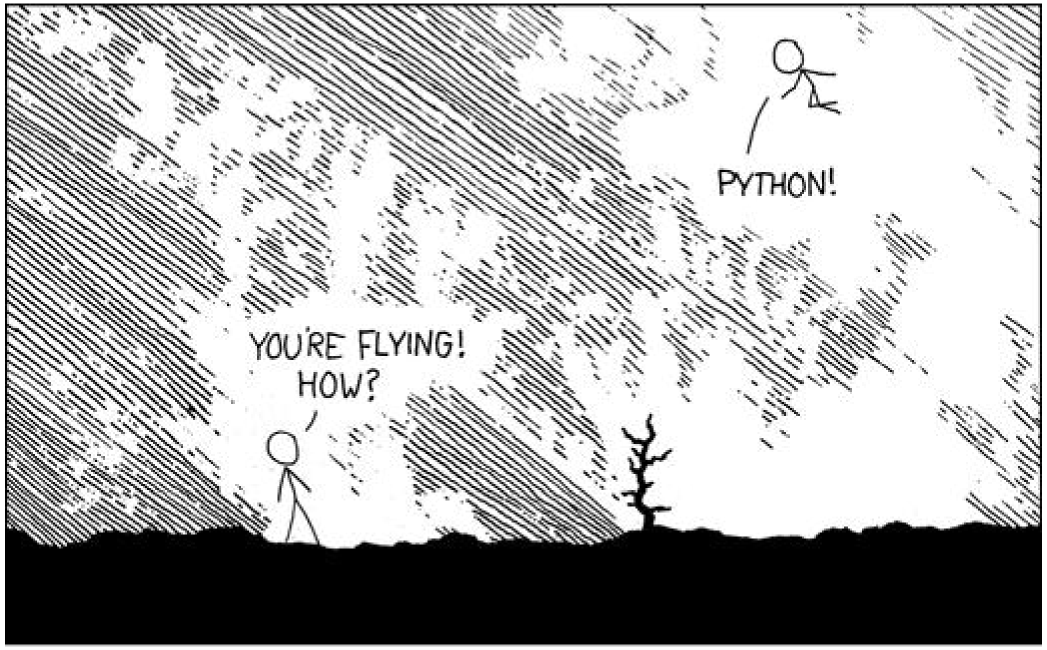
math

```
    .pi, .e, .sqrt(), .sin(), .radians()
```

random

```
    .random() returns random float (0,1)
    .randrange() returns random int within specified range
    .shuffle() puts a list in random order
```

import antigravity



Turtles

Turtles

```
import turtle

wn = turtle.Screen()

donatello = turtle.Turtle()

donatello.forward(150)

donatello.left(90)

donnatello.forward(75)
```

Terminology

```
donatello = turtle.Turtle()
```

`donatello`: **object** (variable)

`turtle`: **module**

`Turtle()`: **class** (type of object)

Terminology

`donatello.forward()`

`donatello: object`

`forward():` **`method`**

Terminology

`donatello.color ()`

`donatello: object`

`color:` **`attribute`**

`color(): method`

OO Exercise

- Think of an object
- Think of two different attributes and methods

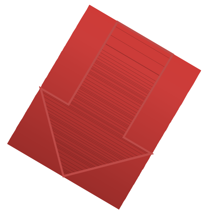
Instances

Every turtle object that you create is called an *instance* of that class

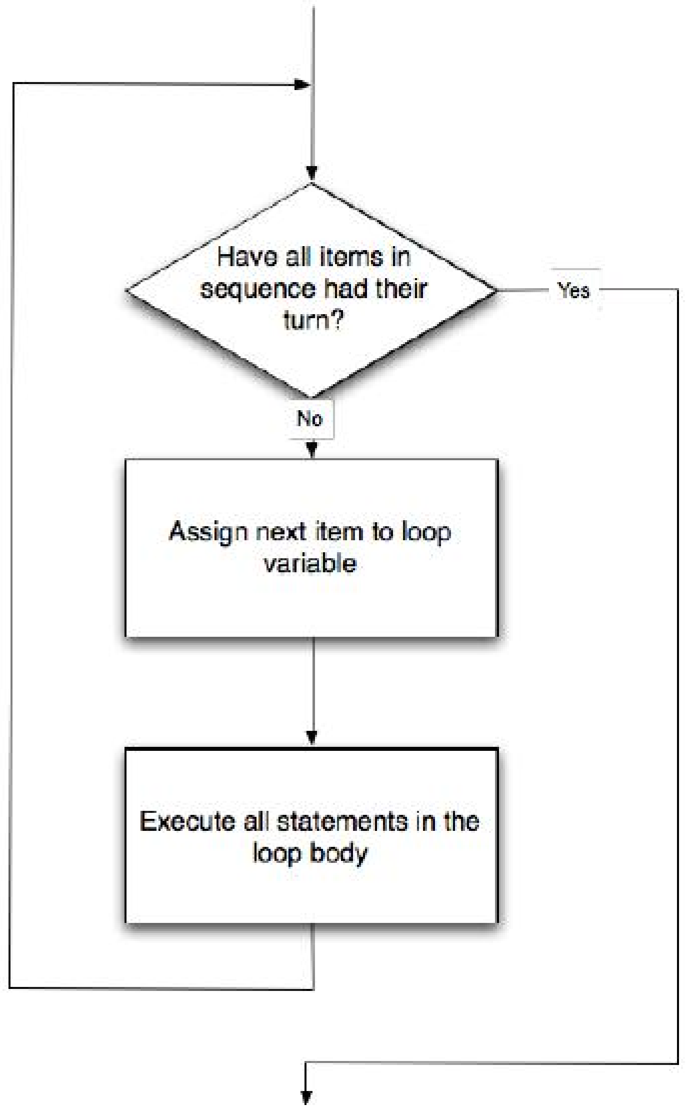
Instances

```
import turtle

donatello = turtle.Turtle()
michelangelo = turtle.Turtle()
raphael = turtle.Turtle()
leonardo = turtle.Turtle()

donatello.forward(100)
raphael.left(90)
raphael.forward(100)
```

Intermezzo:
The for loop



```
for thing in listOfThings :
    doSomethingWith(thing)
```


Have all items in sequence had their turn?
Yes
No
Assign next item to loop variable
Execute all statements in the loop body

Basic Lists

```
[“item1”, “item2”, “item3”]

[1, 2, 3]

[1, “item2”, 3.0]
```

Morning Routine

```
import morning

allTeeth = [‘Molars’, ‘Pre-molars’,
     ‘Canines’, ‘Incisors’]

for tooth in allTeeth :
    morning.brush(tooth)
```

range()

- parameter: integer
- returns: list counting from 0

```
raphael = turtle.Turtle()

for i in [0,1,2,3]:
    raphael.forward(50)
    raphael.left(90)
```

```
donatello = turtle.Turtle()

for i in range(4):
    donatello.forward(50)
    donatello.left(90)
```

range()

- Parameters:
  - Start (integer)
  - Stop (integer)
  - Step (integer)
- Returns: list

Some more turtle methods

```
.heading()

.penup()     .up()

.pendown()   .down()

.shape(‘turtle’)

.speed(10)

.stamp()
```

Do Yoga

Do this 4 times:

- Downward dog
- Plank
- Cobra

Do this two times:

- Warrior 2
- triangle
- Warrior 2
- Warrior 2 reversed

Do Yoga

```
import yoga

thijs = yoga.Yogi()

for i in range(4):
      thijs.downwardDog()
      thijs.plank()
      thijs.cobra()

for i in range(2):
      thijs.warrior2(1)
      thijs.triangle()
      thijs.warrior2(1)
      thijs.warrior2(-1)
```

Modules

- Can contain:
  - Values
  - Functions
  - Classes, which have...
    - Methods
    - Attributes

Functions

Function syntax

```
def functionName(parameters) :
    statements
    return (value)
```

Function Example

```
import turtle

def drawSquare(t, size) :
    for i in range(4) :
        t.forward(size)
        t.left(90)

wn = turtle.Screen()
michelangelo = turtle.Turtle()
drawSquare(michelangelo, 50)
wn.exitonclick()
```

Functions that return values (fruitful)

- Seen examples previously from `math` and `random` modules
- Let’s try and make a `powerof` function, that takes two values `x` and `p` and return x^p

Local vs global scope

- Variables inside function are local and temporary
- Variables outside function are global and persistent
- Python looks first for local variables, then global variables

Functions can call other functions

```
def square(x) :
    y = x * x
    return y

def sum_of_squares(x, y, z) :
    a = square(x)
    b = square(y)
    c = square(z)
    return a + b + c
```

Recap

- Modules
- Object Oriented programming
- Iteration (`for` – loop)
- Functions
- Scope

This week’s homework

- Read e-book: Debugging interlude + Chapters Turtles, Modules, & Functions
- Solve these problems:
  - CH Turtles: 1, 3 – 12 + Turtle Race
  - CH Functions: 1 – 5
- Bring these solutions to lab sessions (hard copy):
  - CH Turtles: 4, 6, 10
  - CH Functions: 2, 4