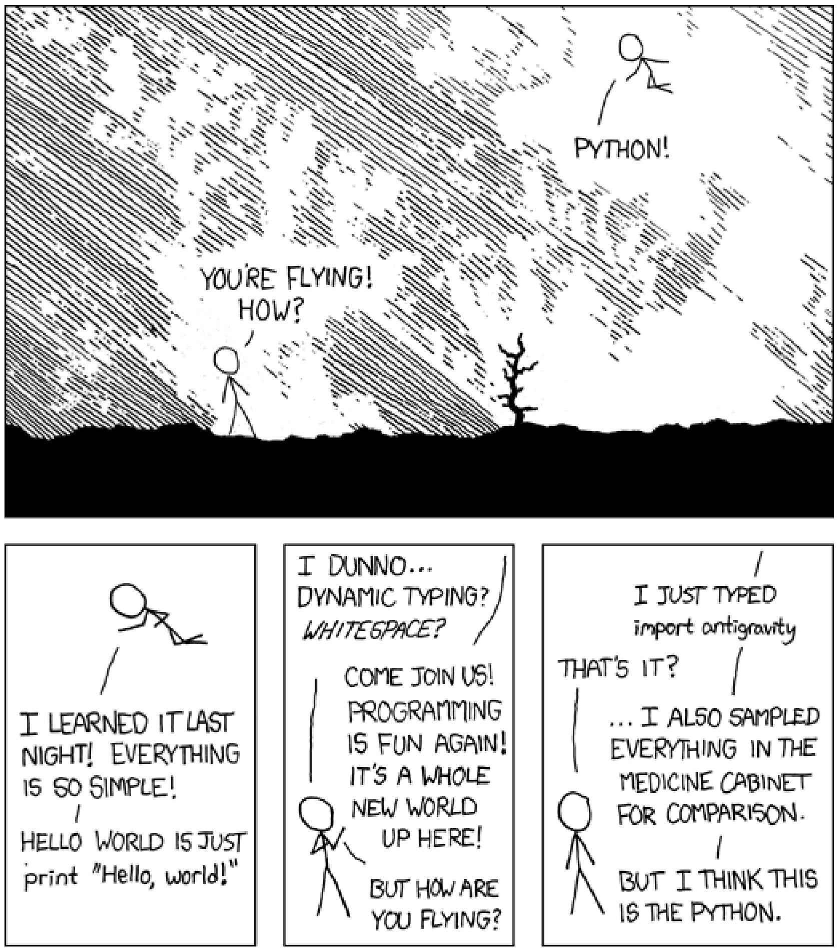
PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Source: http://xkcd.com/353/

3 Basics of Python

Python is a **high-level** language. There are big advantages in working with such languages. The most important is that source code is faster to write and easier to read. You won't have to declare the type of the variables or allocate the memory by yourself!

In these kind of languages the compiling process is absent. You start writing source code and then feed-it into an interpreter which is in charge of executing the program. The interpretation and execution occur alternately. Remember that in the case of a **low-level** language the compiler reads and translates the source code before execution.

Python source code is executed by an interpreter that you can call from the terminal:

```
$python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The prompt `>>>` indicates that the Python interpreter is ready to receive instructions. After each instruction the interpreter displays the result. For instance

```
>>> 1
1
```

or in the case of arithmetic operations

```
>>> 2+2
4
```

You can also interact with the interpreter by writing all the lines that you want to be interpreted into a text file with the extension `.py`. In that case the interpretation and execution can be called from the terminal as follows.

```
$ python mycode.py
```

In this class we are interested in covering the very basics of python. The recommended reference is this on-line text:

- Think Python: How to Think Like a Computer Scientist, Allen B. Downey. `http://www.greenteapress.com/thinkpython/html/index.html`

Another good reference based on the MIT course Introduction to Computer Science and Engineering

- Introduction to Computation and Programming Using Python, John V. Guttag, MIT Press, 2013.

After knowing the basics a very useful repository for useful recipes in Python is here:

- Python Grimoire: `https://taoofmac.com/media/dev/Python/Grimoire/grimoire.html`

In what follows I will go through basic concepts in python illustrated by source code.

3.0.1 My first Python program

Save the following code in a file `hello.py`

```
print "Hello World"
```

Then execute it as:

```
$python hello.py
```

Compare it to the C source code. Smile.

3.1 Variables and Statements

You don't need to declare the type of the variable before using them.

```
# Some variables are declared

a = 1
b = 10.0
c = "computer"

# its type is printed to screen
print type(a)
print type(b)
print type(c)
```

You can also perform simple arithmetic operations

```
#examples of simple arithmetic operations
#division behaves the same way as in C

a = 10.0
b = 9.0

c = 1
d = 10

print "a =",a
print "b =",b
print "c =",c
print "d =",d
print "product a*b", a*b

print "a/b", a/b
print "c/b", c/b
print "c/d", c/d
```

Advanced functions can should be used in the following way

3.2 Conditionals

The most common way to control the flow with the program is with the `if`, `else`, `elif` and `while` statements

```
#controling the program flow with if, else, elif, while


#simple if statement and boolean variable
value = True #boolean variable
if(value):
    print 'The boolean value was true'

# zero is equivalent to False
if(0):
    print 'This will never be printed to screen'


#simple example of if-else
a = 10
b = 4
if(a>b):
    print 'a>b: a=', a, ',b=', b
else:
    print 'a<=b, a=', a, ',b=', b


#simple example of if-elif-else
month='July'

if(month=='January'):
    month_number=1
elif(month=='February'):
    month_number=2
else:
    month_number=-1
print month, 'corresponds to month_number', month_number

if(month_number==-1):
    print 'That means that I don\'t have', month, 'in my list'

#simple example of while statement

print 'Countdown starts'
n=10
while n > 0:
    print n
    n = n-1
print 'Blastoff!'
```

3.3 Iteration

Iteration comes in a flavor of `for` loops:

```
PI = 3.14159
n_points = 12
for i in range(n_points):
    radius = 1.0 * i
    surface = 4.0 * PI * radius**2
    volume = (4.0/3.0) * PI * radius**3
    print radius, surface, volume
```

3.4 Lists

One of the most basic and useful structures in python are lists. They can include any kind of variable.

A list can be defined as:

```
my_list = ["Apple", 3.40, 1, "Hello world"]
```

the following code would print all the items in the previous list

```
print my_list[0]
print my_list[1]
print my_list[2]
print my_list[3]
```

If you want to print the first item to the third

```
print my_list[0:3]
```

Or if you want to iterate over each item in the list and do something with them:

```
for item in list:
    new_item = 2 * item
    print new_item
```

3.5 Reading files

```
# read the file
infile = open('contacto_de_JaimeForero.txt', 'r')

#load the full text by lines
text = infile.readlines()
```

```
#each line is a part of the message
name = text[0]
date = text[1]
place = text[2]
message  = text[3]

print name
print date
print place
print message

#clean and split
name = text[0].rstrip('\n')
date = text[1].split()
place = text[2].split(',')
message = text[3].rstrip('\n')

print name
print date
print place
print message

infile.close()
```

3.6 Functions

The definition of functions follows the generic form:

```
def function_name(inputs):
    statements
```

The indentations of the statements is **very important**. This is the way the interpreter recognizes the end and the beginning of your function. Here is an example of two functions

```
#How to define a function
# *Pay attention to the 4 spaces of indentation*

#Here is a function that doesn't return any value
def print_message(message):
    print 'I am here to tell you this:', message

#Here is a function that returns some number
def volume(radius):
    vol = (4.0/3.0) * 3.14159 * radius**3
    return vol
```

```
#now these two functions are used
print_message('I will be back in 5 minutes')
print volume(5.0)
```