## Python 101

@sam

# History 101

- Created by Dutch programmer Guido Van Rossum, as a hobby project in 89.
- Named after the british comedy group, Monty Python
- First official release was in 1990
- Most recent release was in June 18.
- Free and Open Source Free as in Freedom

# Installing Python

- Pre-installed on OS-X and linux.
- Windows binaries from <a href="http://python.org/">http://python.org/</a>
- Open the terminal on your computer and type python to test if the install is working
- Exit the python interpreter with Ctrl-D

#### Example

x = 34 - 23 # A comment.

y = "Hello" # Another one.

z = 3.45

```
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + " World" # String concat.
    print x
    print y
```

## Numbers in python

Integers - No fractional part, example 3

Float - Fractional part, example 2.1

What about 3.0?

#### Mathematics

- Five operators, +(addition), (subtraction) \* (multiplication), / (division) and % (modulo)
- The first four have their usual meaning and applicable for integers and float
- a%b is gives the remainder when a is divided by b. Only for integers
- Open python on your terminal and try
  - o **5+6**
  - o **31-11**
  - o **5%2**
  - o **7/3**

# Strings

- Strings are simply text data.
- Denoted using ' 'or " ".
- Example, "hello world"
- On your terminal try **print "Hello world"**
- Also try print "Hello" + " " + 'world'

### Boolean

- Data types that store a boolean value (True or False)
- They are often the result of a comparison
- Used with comparison operators, > , < , >=, <= and == (sic)
- The comparison operators return a True or a False value
- Examples , try on your terminal
  - 5 > 6 ----> False
  - 6.3 <= 100.0 ----> True
  - 6.3 == 100.0 ----> False
  - "A" == 'A" ----> True

# Logical Operators

- Three logical operators **and**, **or**, **not**
- They work on Boolean
- **and** takes two parameters and returns True if both of them are True
- **or** takes two parameters and returns False if both of them are False
- **not** takes one arguments and inverts that.
- Examples
  - True and True
  - 4 > 5 or 7 < 2
  - not False

### Brackets

- Denote order of computation in case you are in doubt
- Example -
  - 3\*5 1 will give 14
  - 3 \* (5 1) will give 12
- What will be the output of (11 > 2 and 5 > 1.1) or (3 > 1)?

#### Variables

- Named entities that store a particular value.
- The values can be altered.
- Assignment is done using = sign.
- Variables can be assigned any value in python, not bound to a data-type
- Variables can be re-assigned the values any time
- question?

```
X = 42
X = "The answer to life, universe and everything"
print X
```

#### Spaces and Indentation

- Unlike other languages spaces and indentation matter in python
- No semicolon at the end.
- Newline denotes the end of line
- No braces
- The first line with less indentation is outside of the block.
- The first line with more indentation starts a nested block

### **Conditional Statements**

- Very often there is a need to make decisions in code and branch out
- For example, trying to write a program to achieve this
  - If marks are more than 40 student passes
  - Else the student fails
- Python provides three conditional statements for this
  - $\circ$  if
  - elif
  - Else
- They are often combined with logical operators **and, or**, **not**

Translating the previous scenario to code,

if marks >= 42:

print "Pass"

else:

print "fail"

if marks > 90:

print "A"

elif markes > 75:

print "B"

elif marks > 60:

print "C"

elif marks > 40:

print "D"

else:

print "F"

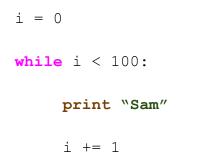
# Lists in python

- Used to store a list of values
- Example a = [1, '1', 2, '2']
- A list can be indexed to fetch individual elements, example,
  - a[0] will give first element
  - a[1] will give the second element
  - A[length 1] will give the last element
- A list can also be indexed from the end, example,
  - a[-1] will give the last element.
  - a[-2] will give the second element from last.
  - How will you get the first element using negative index?

- Strings are lists of characters
- Try on your terminal,
  - 'Hello'[-1]
- Lists can be dynamically appended at runtime
- The syntax is using a .append
- Example :
  - a = [1,2,3]
  - a.append(4)
  - o print a

### Loops

- Loops are programming constructs that enable you to repeat a set of instructions
- Assume you need to print your name, 100 times. You can write the print statement and copy it hundred times, or use a loop
- To achieve the above you can try



- The previous example creates a variable i with a value 0.
- It then starts a "loop", while i is less than 100,(i.e. i < 100 is True) the two statements in the block will be executed.
- The first statement is what we need to execute, print the name.
- The second statement increases the value of i by 1.
- This ensures that the loop is stopped sometime in the future. If you don't increase the value of i, i < 100 will never be False
- Try executing the above code, without the last statement.

## Another example

i = 0
<b>while</b> i < 100
print i
i = i*2
i = 1
<b>while</b> i < 100
print i
i = i * 2

### For Loops

- Another Kind of loop is a for loop.
- Syntax is something like for <a> in <list b>:
- Example

A = [1,2,3,4] for i in A: print i

• It is used to iterate over a list



# Future Reading

Beginners

- <u>https://wiki.python.org/moin/BeginnersGuide/Programmers</u>
- Coursera https://www.coursera.org/learn/python
- Books <u>Learn Python the Hard Way</u>
- <u>https://www.codecademy.com/learn/learn-python</u>
- Different language programmers <u>http://tdc-www.harvard.edu/Python.pdf</u>

Intermediate

• Python Wiki - <u>https://wiki.python.org/moin/BeginnersGuide/Programmers</u>