Python Workshop

Gabriel Gaudreault
Concordia University

February 12, 2016

Today

Hello

- Intro to Programming
- Syntax of Python
- Demos
- Particularities of the Language
- Extra Resources

Why Python?

- Easy to Learn
- More Natural Syntax: no ";", fewer "{", "}"
- No Pointers
- Recent
- Dynamically Typed
- Automatic Memory Management
- Lots of Libraries
- Multi-Paradigm: Imperative, Object-Oriented, Functional, Scripting, Web
- High-Level
- Big Community

Why Not Python?

See previous slide

Python

Interactive	Interpreted	Script
`Gabriel$ python`	`Gabriel$ vi code.py`	`Gabriel$ vi code.py`
`>>> fav_number = 3`	`Gabriel$ python code.py`	`chmod +x code.py`
`>>> 2 + fav_number`		`Gabriel$ ./code.py`
`5`		
`>>>`		

For script mode, include `#!/usr/bin/python` at top of the code

“Main” function in Python :

```
if __name__ == "__main__":
     starting_function()
```

First Program

Printing salutations

DEMO: HELLO + INTERACTIVE

Standard Data Types

- *Boolean
- Numbers (integers, float, long, complex)
- Strings
- List
- Tuple
- Dictionary

Variables

Unlike other languages, Python is dynamically typed, which means that in a lot of cases types do not have to be explicitly stated, for example in variable instantiation or when defining a function

DEMO VARIABLES

Numbers

Basic math operators in Python:

$$+ \ - \ / \ * \ \% \ \backslash \ < \ > \ <= \ >= \ // \ abs \ **$$

Different types:

Integers

Long

Float

Complex

INTERACTIVE DEMO

Truth

Special keywords: True, False

Most types can be evaluated to a truth value...
True: any non-zero integer, characters, non-null strings, non-empty lists, non-empty dictionaries, ...
False: 0, [], “”, ,...

Logic Operators:

$$or\ \ and\ \ not\ \ != \ == \ <= \ >=$$

Truth

BE CAREFUL:

```
True and 1
True == 'a'
True = 43
True == 1
True and "elephant"
```

DEMO

Characters and Strings

- Characters are enclosed within single quotes, e.g. 'a'
- Strings use double quotes, e.g. "hola"
- Combine strings with "+", e.g. "Super" + "man" = "Superman"
- Access single characters using index, e.g. cat[2] = "t"

Note: Just as in most things in computer science, indexing starts at 0 not 1!

Characters and Strings

Useful functions on strings:

$$append, +, *, [i], [i:j]$$

INTERACTIVE DEMO

Back to Print

- Add "," at end of print statement to keep cursor on same line
- Can combine values using ",", "+", or inline format characters
- Special characters: $\backslash n, \backslash$ ", $\backslash\backslash, \backslash t, \backslash *$

INTERACTIVE DEMO

Input

- Get input from console using `raw_input("text")`
- The argument for the function gets printed out and the output is whatever gets written by the user

DEMO

Lists

- Python also supports Lists, e.g. $["a", "b"], [[[1], 2], 3], ["Edward", "Paul", "Suzie", "NotNicole"]$
- Lists are mutable
- Unlike other languages, Python lists are “untyped”
- Useful operations on strings: pop,del, len, in, append, index, insert, remove(obj), reverse, sort

INTERACTIVE DEMO

Dictionaries

- Python also supports Dictionaries by default, e.g. $me = "name" : "gabriel" , "hands" : "2" , "t - shirt" : "blue"$
- Dictionaries are also mutable
- Access values through keys, e.g. $me["name"] = "gabriel"$
- Useful operations: del, clear(), len(), has_key(), items(), keys(), values(), update()

INTERACTIVE DEMO

Tuples

- Tuples $(a1, ..., an)$ are also supported in Python
- Tuples are IMMUTABLE
- You can still access the data, just not modify it

INTERACTIVE DEMO

Functions

```
def my_function(inputs):
    ....
    do stuff
    ....
    return output
```

- Functions are blocks of code with input and output
- They are reusable structures
- Functions are not run when encountered, have to be called
- Functions have to already have been seen by the interpreter before being called

DEMO

Note: In Python, tabs/spaces are super important to structure the code, as opposed to {...} or ";" in other languages

Back to Types

Not having explicit types can be fun and make code less heavy to read, but can be problematic as code

INTERACTIVE DEMO

Memory

Some types in Python are more basic than others:
- Strings $\equiv$ Lists of characters
- Characters and Numbers passed by value instead of by Reference
- Dictionaries, Strings, Lists built recursively on other types

INTERACTIVE DEMO

Files

- Open files with `variable = open(filename, flag)`
- Read the files with: read(), readlines()
- Write with `write()`
- Close file with `close()`

INTERACTIVE DEMO

Conditionals

```
if CONDITION:
    do stuff
elif CONDITION:
    do stuff
else:
    do stuff
```

- Only `if` is obligatory
- Interpreter tests a branch then either runs the inside code or skips to new branch

INTERACTIVE DEMO
WRITE IFF ESLSJFIE ELIF ILS CODE

Loops

2 different kinds of loops in Python

- *For* loops: Runs what is in the body once for each value in A

  ```
  for i in A:
          do stuff
  ```

- *While* loops: Runs what is in the body as long as the condition is true

  ```
  while CONDITION:
          do stuff
  ```

Use `break` to force exit from the loop

INTERACTIVE DEMO

Regular Expressions

Super useful when dealing with text and words
Useful functions:

- `re.match(pattern, text, flags)`: searches the text for the pattern from the beginning of the text
- `re.search(pattern, text, flags)`: searches the text for the pattern, anywhere in the text
- `re.sub(pattern_to_match, pattern_to_sub, text)`: replace the first pattern by the second

The pattern in these cases has to be either of the form `r'pattern'` or you can use the compile function to turn a string into a pattern.

TREE DEMO

Lambda

Python also supports anonymous functions of the style $\lambda x.f(x)$ written as `lambda x:  f(x)`

SEMANTICS DEMO

Resources

http://learnpythonthehardway.org/book/
https://web.stanford.edu/class/linguist278/
http://www.tutorialspoint.com/python/
https://docs.python.org/2/tutorial/