GENERAL ASSEMBLY

PARSELTONGUE (INTRO TO PYTHON)

Haley Boyan

Data Science Instructional Associate, General Assembly DC

LEARNING OBJECTIVES

- What are front end and back end programming languages?
- What is Python?
- How is it similar to/different from other programming languages?
- Python data types and collections
- Variable assignment
- Language structure (comments, indentation, etc.)
- Basic and conditional operators
- Formatting strings
- Modules and connecting files

INTRO TO PYTHON

PRE-WORK

PRE-WORK REVIEW

- Python course on CodeCademy

PROGRAMMING LANGUAGES

PROGRAMMING LANGUAGES

- Computers know machine language, based on binary
- Programming languages are written based on human language
- Programmers write scripts, or sets of instructions for computers
- Compilers translate human language into machine language

FRONT END VS BACK END LANGUAGES

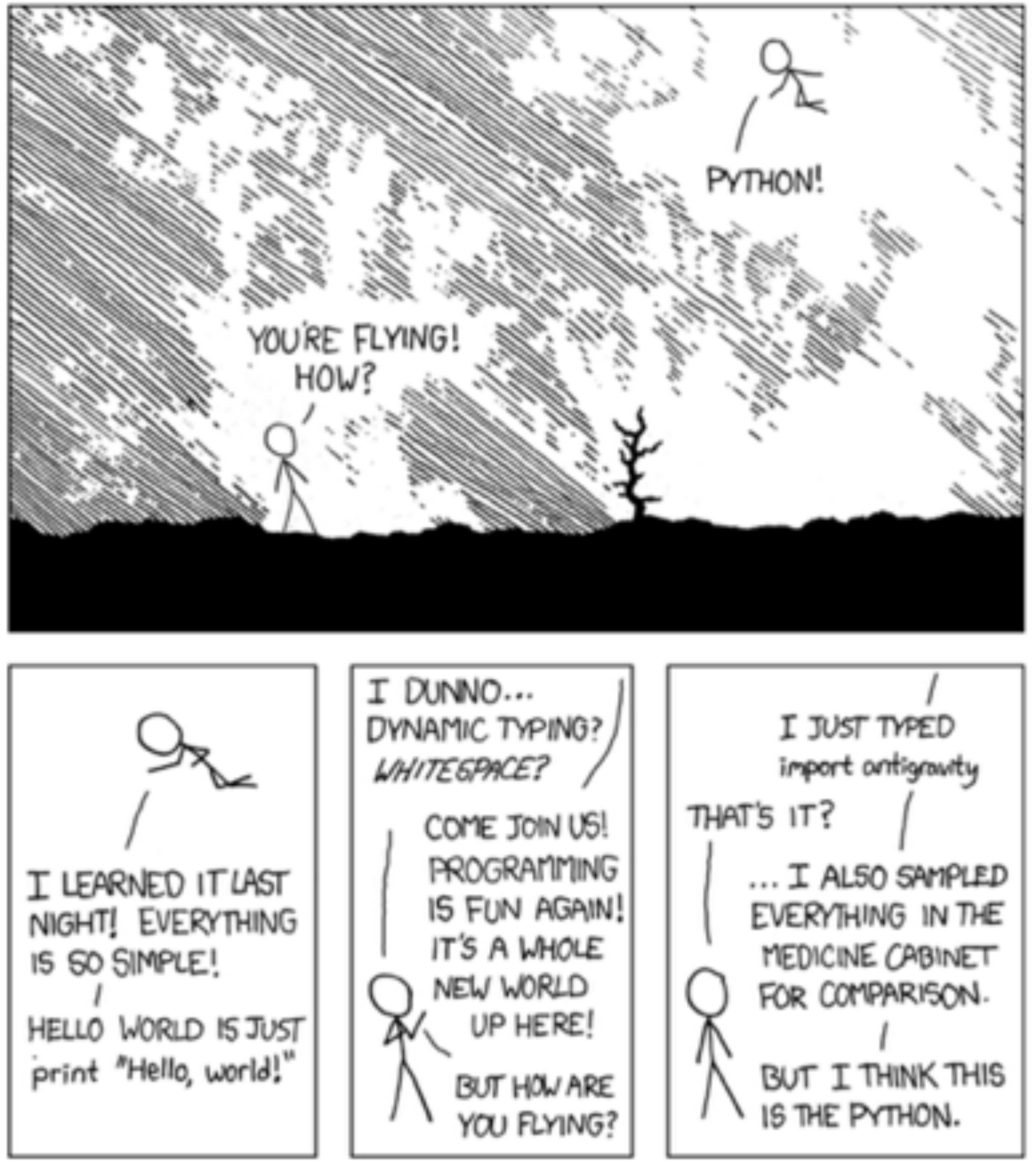
- Front-End: Concerned with user interaction, like website design or user interface
    - E.g. JavaScript, HTML, CSS
- Back-End: Operations side, managing data and information
    - Includes 3 parts:
        - Database: all stored information
        - Application: creates/deletes/changes/renames/etc. items in database
        - Server: computer accessed remotely, runs software to fulfill client requests
    - E.g. Python, Ruby, Java, PHP

OBJECT ORIENTED PROGRAMMING

- Originally, programs were “logical procedures that take input data, process it, and produce output data”
- Object-oriented programming (OOP) is a programming language model organized around objects rather than "actions" and data rather than logic
- **Object**: a piece of code with a state (attributes) and behavior
  - **State**: stored in fields, known as attributes, NOT callable (e.g. x.size)
  - **Behavior**: stored as methods, which are callable (e.g. x.doThing())
- **Class**: blueprint or prototype from which objects are created
- **Inheritance**: Classes inherit state and behavior from their superclasses

WHAT MAKES PYTHON UNIQUE

- Relies on indentation as a control structure
- Allows you to use variables without declaring them (determines types implicitly)
- Easy to understand, write, and interpret
- Everything is an object (Data types, Collections, Functions, Modules)
- Free


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.

MODULES, PACKAGES, LIBRARIES

- Module:
  - File containing Python definitions and statements
  - File name is the module name with the suffix .py (thing.py)
- Package:
  - Collection of modules under a specific namespace
- Library:
  - When a module/package/something else is "published" people often call it a library
  - Libraries can contain a single module, a single package, or multiple related packages
  - Libraries usually do not provide any specific functionality (you cannot "run a library”)

VERSIONS OF PYTHON

- Python 2.7 vs 3
- 2.7 is stable, 3 is evolving
- Some libraries only work with one version or another
- We will use 2.7
- Check version in Terminal:
    - `python -V`
    - capital V matters!

DATA TYPES IN PYTHON

DATA TYPES

- Every object in Python has a data type
- These hold different types of data, and have different attributes and methods
- Can be manipulated in different ways

STRINGS

- Sequences of unicode characters
- Anything you want to be read literally
- Words, sentences, etc.
- Surrounded by quotes (single or double work)

- Examples:
  - ‘apple’
  - ‘United States of America’
  - “1200”

- Treated as literals, so numbers are just characters (if you add “1” + “1” it will output “11”)

NUMBERS

- Integer: Whole number (1 or 2)
- Float: Decimal number (1.0 or 2.0)
- Long: Numbers with too much information involved, only exist in 2.x

LISTS

- Lists are collections, which store multiple objects
- Surrounded by brackets: [ ]
- Can contain multiple types:
    - list_x = [‘apple’, 1, True]
- Stored in order, called ‘index’
    - Index starts at 0
    - Last index is -1
    - Can call an item at its location with list[i], where i is the place
    - e.g. in the list above, list_x[0] would be ‘apple’, and list_x[1] would be 1
- Mutable (can be changed)

TUPLES

- Immutable lists (cannot add, remove, or change items)
- Indexed and sliced the same
- Surrounded by parentheses: ()
- Computationally less expensive

DICTIONARIES

- Unordered set of key value pairs
- All keys must be unique, value can be same
- Surrounded by squiggly brackets: { }
- Can be modified

OTHERS

- Booleans: True or False
- Datetimes: Stores date and time information
    - Differences between them are “timedeltas”
- Bytes/Byte Arrays: e.g. images

OPERATORS IN PYTHON

ASSIGNMENT

- Defining “variables”
- Variables are items that are defined by you and hold values
- Variables names must start with a letter or underscore
- Cannot be the same as a keyword already in Python
- Python does not make you declare data type, but you can
- Variables can be reassigned

- x = 12
- cat = “yellow”

NUMERICAL OPERATORS

- + (add)
- - (subtract)
- / (divide)
- * (multiply)
- % (modulus/remainder)
- ** (exponent)

- If you add the equal sign after an operator, it means do the operation and reassign the value of the original
    - x = 1
    - x + 1 (returns 2)
    - x += 1 (now x equals 2, returns nothing)

CONDITIONAL OPERATORS

- == (is equal to)
- != (does not equal)
- < (less than)
- > (greater than)
- <= (less than or equal to)
- >= (greater than or equal to)
- a

BOOLEAN OPERATORS

- & (AND)
- | (OR)
- ^ (XOR - only in one or the other)
- ~ (NOT)
- in
- not in

FORMATTING STRINGS

- % can also be used to hold a space for a variable in a string
- “I am %d years old” %(21)
  - returns “I am 21 years old”

BASIC PYTHON COMMANDS

BASIC PYTHON COMMANDS

- print: Displays the content
- return: Assigns the output of a function, but does not display it
- import: Connect and bring in the contents of a file or library
- input/raw_input: Assigns user input as value of a variable
- type(x): Displays the data type of variable x
- range(x, y): All the integers between x and y
    - If you only enter one variable, x defaults to 0
- length(x): Number of characters/items in variable x

COMMENTS

- Pseudocode: Essentially a written explanation of your code
    - Usually written before actually putting in the code content
    - Can help others understand what your code does
    - Good reference point for yourself
- One line comments: #
- Multi-line comments: ‘’’ (at start and end)

DATA TYPE SPECIFIC METHODS

- All data types have their own specific methods that can be used
- There are methods in the class:
    - string.digits = 0123456789
    - string.uppercase = ABCDEFGHIJKLMNOPQRSTUVWXYZ
- There are methods that can be called on a variable of each data type:
    - string.replace(inputVariable, old, new)
- There are methods that only work on that data type:
    - length(inputVariable)

GUIDED PRACTICE: PYTHON BASICS

INTEGERS AND FLOATS

EXERCISE

DIRECTIONS

1. In Terminal, type Python
2. You should see >>> (if not, please tell me)
3. Try typing and returning each of the following:

```
22
-44
45-19
```

4. Try typing and returning each of the following:

```
4.0
18.7
45.0 - 19.0
```

STRINGS

EXERCISE

DIRECTIONS

1. Try typing and returning each of the following:

```
“Hello world”
x
x = “Hello world”
x
print(x)
x[1]
x[1:5]
x[:4]
x[4:]
```

LISTS AND TUPLES

EXERCISE

DIRECTIONS

1. Try typing and returning each of the following:

```
x = [‘red’,’yellow’,’green’]
x[0]
x[1] = ‘blue’
print(x)
```

2. Try typing and returning each of the following:

```
y = (‘red’,’yellow’,’green’)
y[0]
y[1] = ‘blue’
print(y)
```

DICTIONARIES

EXERCISE

DIRECTIONS

1. Try typing and returning each of the following:

```
x = {}
x[‘apple’] = ‘red’
x[‘lime’] = ‘green’
print x
x = {‘apple’:’red’, ‘apple’:’green’}
x = {‘apple1’:’red’, ‘apple2’:’green’}
x[‘apple1’]
```

OPERATIONS

EXERCISE

DIRECTIONS

1. Try typing and returning each of the following:

```
x = 1
y = 7
x + y
z = ‘Let’s’
q = ‘go’
z + q
z + ‘ ‘ + q
q * 8
test == z
test = z
```

BEFORE NEXT CLASS

BEFORE NEXT CLASS

DUE DATE

- Homework:
- http://campus.codeschool.com/courses/try-python/contents
- Complete Course

TITLE

CREDITS

TITLE

CITATIONS

TITLE

Q & A