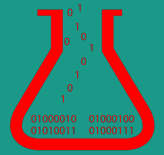
Intro and Setup



Python Basics

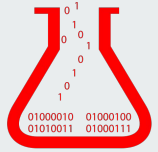
What is Python?

Python is a general-purpose, open-source programming language

Created by Guido van Rossum in 1991

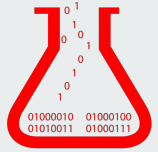
Python 3 (current version)

Python 2(EOL 2015, postponed to 2020)



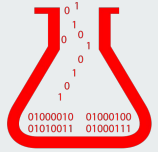
What is Python (con'd)

- Interpreted (scripted, not compiled)
    - - Same code can be used across Operating Systems
- High-Level
    - - Automatically handles low level functionality
- Object-Oriented
- General-Purpose, Multi-Paradigm



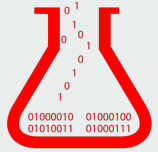
Python Features

- Interpreted language (not compiled)
  - Can be used by multiple Operating Systems
- Dynamically Typed
- Automatic memory management
- Object-Oriented
- A large variety of comprehensive libraries
- Made for readability!



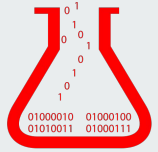
Python Libraries

- Many open-source libraries for analytics:
  - numpy for linear algebra / array mathematics
  - pandas for data manipulation/exploration
  - scikit-learn for machine learning
  - Web-scraping: request, beautifulsoup, Selenium
  - Data Visualization: matplotlib, seaborn, ggplot



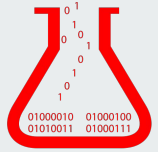
Python - Interpreted language

- A Compiled Language:
  - Needs to run through an additional ‘compile’ step at runtime to make the code machine readable (and checks for errors)
- Python - runs through the interpreter; allowing faster iteration / changes in the code



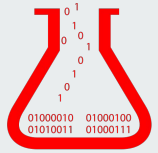
Python - Dynamic Typing

- Dynamic Typing does not link objects to a specific type
- Type checking done at run-time
- Allows variables to change types:
  - number = 4
  - number = ‘four’
  - Number doesn’t have to be an integer or a string, can be both depending on the situation



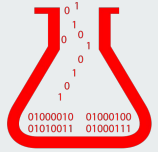
Python - Memory Management

- The user does not need to assign memory blocks to variables or other objects
- Python automatically does this behind the scenes



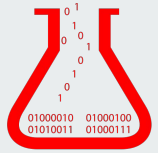
Python - Lazy Evaluation

- Python does not run code line-by-line
- The script is converted to an execution stack
- Values are evaluated when they are needed for execution (e.g., when printed to stdout)
- Values that are never used are never evaluated
- Usually, this is inconsequential, but becomes important when interacting with external devices/objects (e.g., webdrivers when web-scraping)



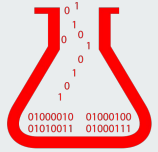
Python - Object Oriented

- Everything is an object in Python
- Therefore almost everything has attributes & methods
- Many libraries are written as objects so it is beneficially to understand the syntax



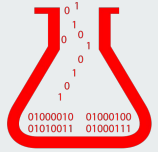
Python - Readability

- Python is written to allow greater code readability
- Whitespace is encouraged between code blocks
  - This whitespace is ignored by Python's interpreter
- Python code uses indents of 4 spaces/1 tab to define code blocks (instead of brackets {})



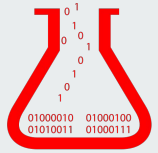
Python 2 and Python 3

- Python 2 was planned to be End-of-Life'd (EOL) and discontinued with the release of Python 3.
- But, the wide-spread use of Python 2 led to its EOL date being pushed back.
- In fact, many of the features of Python 3 were back-ported with the release of Python 2.7.
- The "final" EOL is planned for 2020.



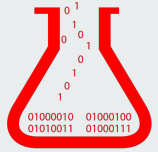
Python 2 vs Python 3

- Python 2.6/earlier and Python 3 have many differences. But, Python 2.7+ and Python 3 are very similar for most applications. The __future__ library even back-ports many Python 3 features and allows for code compatibility. A detailed overview can be found here:
- https://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html



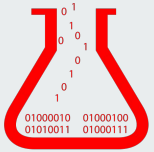
Python 2 vs Python 3

- Most of the differences between Python 2.7+ and Python 3.4+ are very subtle and beyond the scope of this course.
- For this course, we'll be using Python 3.



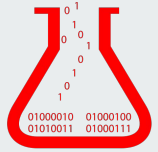
Language Format

Variable Assignments
Spacing
Colons



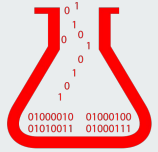
Language Format - Variable Assignments

- Python encourages using descriptive variable names
- Python uses the ‘=’ operator as the variable assignment operator
  - count = 1
  - name = ‘my name’



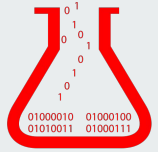
Language Format - Spacing

- Python uses whitespace and indentation to separate blocks of code.
  - Indentation = a Tab (which is converted to 4 spaces)
  - Lines end with a newline character, not a semicolon
    - Semicolons (‘;’) are technically allowed and can be used to make complex statements, but this is considered bad practice.
  - Colons (‘:’) indicate the start of a code block, such as a definition (class/function) or flow-control (if/else/while/for)
  - The block of code after the colon is indented by 1 tab
  - The block ends when the indentation ends



Language Format - Colons

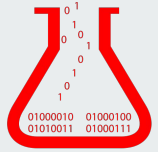
- Colons (‘:’) indicate the start of a block of code
  - Definitions (functions/Classes/methods in a Class)
  - Flow-Control (if/else/while/for)
  - Error Handline (try/except)
- The code block is indented by 1 tab
- The block ends when the indentation ends
- Blocks can be nested (an if block inside an if block, or nested for loops)



Language Format - Example

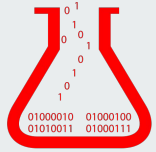
```
counter = 5

while counter > 0:
    print(counter)
    counter -= 1

print("Liftoff!")
```

```
5
4
3
2
1
Liftoff!
```



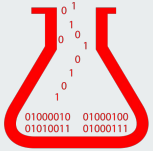
Language Format - Comments

- Comments start with a hashtag (#)
  - Be careful about indenting/un-indenting comments. A comment with indentation that doesn't match its code block can cause an error!
  - Block comments start/end with triple quotes (“““)
- Technically, block quotes are just a string that isn't used or assigned to a variable. It is “stated,” but because it is not stored, its contents (the commented out block of code, as a string), it never makes it on the execution stack, and is ignore.
- Practically speaking, just be aware that block comments use the same triple quotes that strings may use.
- Make sure to close block comments! Editors often insert a closing quote to make coding quicker/easier. When you go to comment out a block of code, keep an eye on the auto-completed quotes!



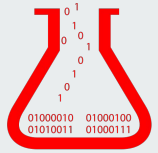
Variable

Variable Assignment
Dynamic Typing
Variable Naming
Mutability



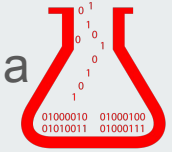
Variables - Overview

- Python uses variables to store information
- Syntax to assign a value to a variable is the equal sign
  - For example: count = 6
  - Assigns the integer value 6 to the variable named count
- Python is dynamically typed:
  - Dynamically typed means that Python doesn’t need to know the type of the variable - the variable name just ‘points’ to that object and the object has a type associated with it



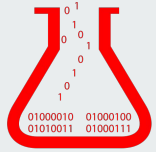
Variables - Dynamic Typing

- Python is dynamically typed:
  - Dynamically typed means that Python doesn't need to know the type of the variable on initialization
  - The variable name just 'points' to that object and the object has a type associated with it
  - For example: count = 6 (count is an integer)
  - And then we do: count = 'six' (count is now a string)
  - Python is ok with this change and won't give an error
- Opposite is strongly typed where you have to define a type for each variable and then that variable name is only associated with that type
  - For example: count: int = 6
  - 'count' is always of type 'integer' and will get an error if assigned to a different type



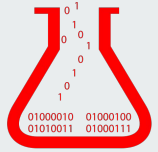
Variable Naming

- Case matters (mystr is a different variable than MyStr)
- Cannot start with a number
- Usually variable names are in all lowercase
- Can use underscores to make them more readable
    - E.g.: word_dict or my_list
- Keep variable names short (you might have to write them a lot!)
- ‘Counter’ variables are often a single letter like: i,j,k
- Try to name variables something that is easy to read for you and other programmers (i.e., avoid lowercase “L” as it looks like uppercase “I” and pipe: l, I, |)



= VS ==

- = is the assignment operator
  - That is, ‘=‘ assigns the values on the right to the variable name on the left
- == is the equality operator
  - == checks if the value on the left equals the value on right
    - Returns True if the values are equal
    - Returns False if the value are not equal
- Examples:
  - 3 == 3.0 (True)
  - True == 1 (True)
  - 4 == ‘4’ (False)



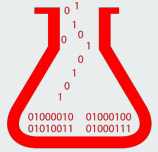
Mutability

Mutable = the item can be changed after created

Immutable = the item cannot be changed after created (but can be over-written with a new value)

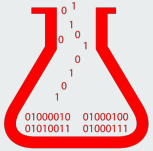
All primitives are immutable

Some iterables are mutable (i.e., lists, dictionaries), others are immutable (i.e., tuples, strings)



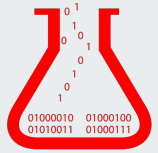
Basic Variable Types

Variable Assignment
Dynamic Typing
Variable Naming
Variable Types



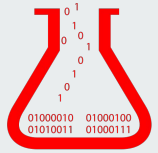
Python's Basic Variable Types

- Atomic Types ('Primitives')
- Data Structures ('Iterables')



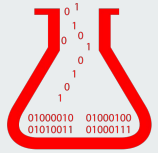
Primitive (Atomic) Types

- Integer (int)
- Float (float)
- Boolean (bool)
- NoneType
- String (str)



Iterable Types(Data Structure)

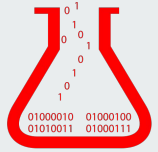
- List
- Dictionary (dict)
- Tuple
- Set
- Strings
  - Though technically a primitive type, strings act like iterables in many ways, and are iterable in loops.



Basic Built-in Functions

Introducing a couple built in functions (more in a later section):

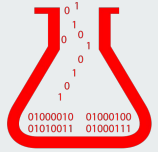
- len() <- returns the length of that object
  - len('mystring') → 8
- type() <- returns the type of that object
  - type(2.0) → float
- print() <- prints to stdout (the screen or command line by default)
  - print('mystring') -> prints: mystring



A Look Under the Hood

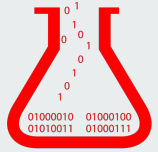
Everything in Python is an object!

- Each object has various methods - some of these are ‘named’ methods that will work on many different object (any object with that method defined)
- For example:
  - len() calls the object method named object.__len__()
- This can be extended to your own objects by defining that __len__() method in them



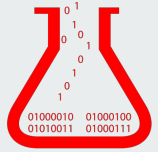
Where to Look for Questions/Answers

- **Python.org**: official Python site, has all official documentation
- **Stackoverflow.com**: StackOverflow is a forum for programming questions in all languages, and has a HUGE Python section. Most basic questions probably been asked, or a similar question has been asked, and there are a TON of answer out there
- **Google**: Be aware if the answer is for Python 2 vs 3. If in doubt, use ‘python3’ (no space) in your search query to narrow your results.



PEP8: The Official Rulebook

- https://www.python.org/dev/peps/pep-0008/
- PEP8 is the official style guide for Python programming
- It was one of the first documents written
- Its author, Guido van Rossum, is the creator of Python!
- PEP is short for Python Enhancement Proposal
- PEP8 is the industry-wide standard for Python formatting and style, and a great way to learn some of the nuances of the language
- (In practice, most programmers follow most of the rules)



Questions?

Contact:

Denis Vrdoljak
denis@bds.group