Getting started with Python pt 1

Irene Siragusa, PhD

Python

- Python is an interpreted, general-purpose, object-oriented language that is widely used nowadays for data analysis and machine learning applications.
- Useful resources
  - Google's Python Class [https://developers.google.com/edu/python]
  - Google's Python Class exercises [https://developers.google.com/edu/python/google-python-exercises.zip]
  - W3Schoolds Python [https://www.w3schools.com/python/]
  - Python's official documentation [https://docs.python.org/3/]

Python

- Create folder big data
- Terminal
  - Test via termial per python
    - Py windows installed via url
    - Python windows from store
    - Python/Python3 mac

  - https://docs.python.org/3/library/venv.html
  - Create venv

```
python3 -m venv /Users/irene/big-data-env
```

  - Activate venv

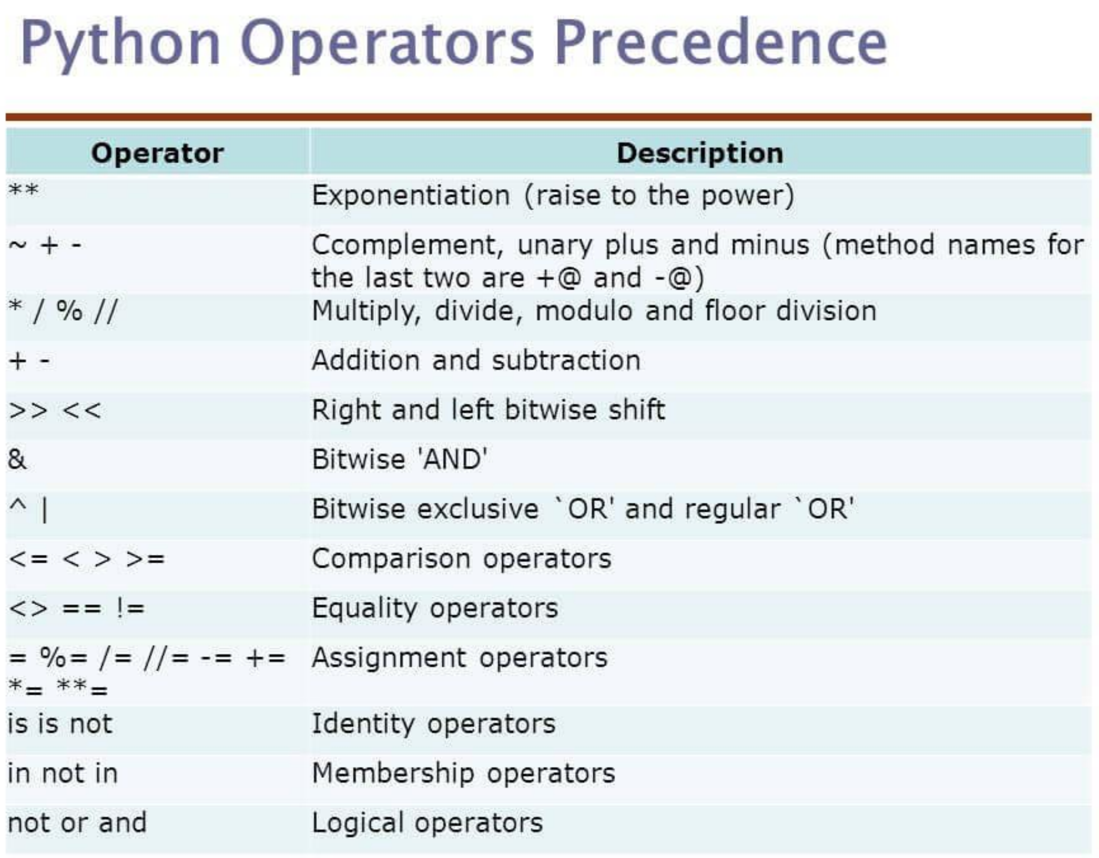
```
source /Users/irene/big-data-env/bin/activate
<venv>\Scripts\activate.bat
```

  - Open folder big data in vs code
  - Install library numpy
  - Test w/ test.py and test.ipynb

Python

- Further information about Python's build-in functions [https://docs.python.org/3/library/functions.html]
- Python documentation

[https://docs.python.org/3/]



Python Operators Precedence

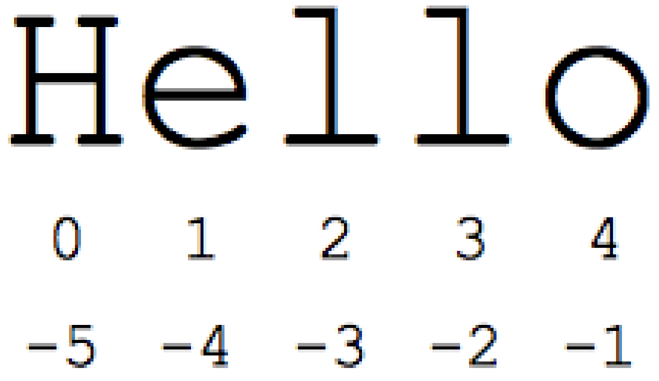
Operator	Description
**	Exponentiation (raise to the power)
~ + -	Ccomplement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Data types

- Integers
  - Represented with 32-bits, can be converted from others formats with `int()`
- Reals
  - Can be obtained via built-in function `float()` with range [-10^{308} , 10^{308}] a number can be printed as -3.456 or in scientific notation -3.456e-7
- Booleans
  - Related to logical operations and to relational operator and can have only True or False values
- Strings
  - Can be obtained via `str()` or within single or double apex notation;
  - Can be accessed via array notation and their length can be obtained via `len()` function

Data types

- The following are some escape characters `\n, \t, \b, \\, \', \”` , and strings can be concatenated via + operator
- Strings, list and tuples can be accessed with the following notation `[start:stop:step]`
- Note that stop position is excluded (you arrive at stop-1)
- You can use a variant of the standard indexing



Data types – Strings functions

- Strings cannot be modified, if manipulation is needed a new object is built.
- `s.lower()` and `s.upper()` return string s in lowr/upped case
- `s.strip()` removes any white spaces at the beginning and at the end of the string
- `s.isalpha()`, `s.isdigit()`, `s.isspace()` returns `True` if ALL chatacters in a string are alphanumeric, numeric, spaces
- `s.startswith('other')` and `s.endswith('other')` check if string `s` starts/ends with string `other`
- `s.find('other')` search other in s and returns the starting index of the first occurrence, otherwise -1
- `s.replace('old', 'new')` returns a string where all the occurrences of `old` are substituted with `new`
- `s.split(delim')` return a list of sub-strings divided by the given delimitator. Defaul delimitator is a single space.
- `s.join(list)` concats element of a given `list` in a string with `s` as delimitator.

Print function

- To output/display a python object use `print()` function
- To print well-formatted strings, use `print(f’’)`
  - `pi = 3.14159265358979`
  - `print(f’Pi is {pi:1.5f}’) -> Pi is 3.14159`

Lists

- List can be defined as `a = [element, element, …]`
- Can contain different types of elements and can be accessed via array notation and slicing (as strings)
- Can be modified
- `len(a)` return length of a given list
- `a.pop(0)` removes first element
- `a.append('Monica')` appends given element at the end of a list
- `a.insert(1, 'Joey')` insert given element at given position
- `list.extend(['Rachel', 'Phoebe'])` extends the given list with another
- `print(list.index('Joey'))` returns index of a given element
- `list.remove('Joey')` if the element is found, is removed from a list
- `print(list[1:-1])` prints list from index 1 to second-to-last element

Tuples

- `a = (element, element, …)`
- Cannot be changed (elements can be modified)
- Generally used when writing functions

Control functions – `if elif else`

```
if condition:
        istruction
elif condition:
        istruction
       ...
else:
       istruction
```

Control functions – `for`

```
for iter in iterable:
    istruction
```

Control functions – `while`

```
while condition:
     istruction
```

Sorting

- Sorting of an iterable object can be obtained using the following function

```
sorted(iterable, key=None, reverse=False)
```

- It return a list of ordered element in ascending order.
  - `key` refers to a function that works as ordering criteria
  - `reverse=True` enables descending ordering.
- Lists have the built-in method `sort()` for in place ascending sorting and returns `None`

Hands on exercises!

E1. Statistical measures

- Calculate mean, variance and standard deviations by formulas and double check with native functions in statistics module

E2. Degrees to Radians

- Implement a radians function that returns the radian equivalent of a degree. Use the following formula:

```
rad = degrees * pi/180
```

- Use this function to print a chart showing the radian equivalent of all degrees ranging from 1° to 180°. Use two digits of precision for the results. Print the outputs in a neat tabular format.

E3. Duplicate Elimination

- In organizations, a list of email addresses is often compiled for marketing purposes. However, duplicate email addresses need to be removed from this list. Write a function that receives a list and returns a list containing only unique values. Test your function with a list of email addresses.

E4. Counting Votes

- Write a script that uses a dictionary to determine the number of votes received by candidates in an election. The votes are concatenated in a string where each vote is separated from the next by a comma.
- [Hint: split can take in an argument for the specific delimiter you wish to use in a string.]

E5. Class Average – TXT

- Write code that enables you to store any number of grades into a grades.txt plain text file.
- Write code that reads the grades from the grades.txt file you created in the previous exercise.
- Display the individual grades and their total, count and average.

E6. Class Average – CSV

- An instructor teaches a class in which each student takes three exams. The instructor would like to store this information in a file named grades.csv. Write code that enables an instructor to enter each student's first name and last name as strings and the student's three exam grades as integers.
- Use the csv module to write each record into the grades.csv file. Each record should be a single line of text in the following CSV format:

```
firstname,lastname,exam1grade,exam2grade,exam3grade
```

- Read the grades.csv and display the data in tabular format.

E7. Class Average – JSON

- Use the json module to write the student information to the file in JSON format, create a dictionary of student data in the following format:

```
gradebook_dict = {'students': [student1dictionary,
                student2dictionary, ...]}
```

- Each dictionary in the list represents one student and contains the keys 'first_name', 'last_name', 'exam1', 'exam2' and 'exam3', which map to the values representing each student's first name (string), last name (string) and three exam scores (integers).

- Output the gradebook_dict in JSON format to the file grades.json.

- Read the grades.json and display the data in tabular format.