# Python

Official Documentation - <https://docs.python.org/3/>

# Data Types and Operator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Structure** | **Ordered** | **Mutable** | **Constructor** | **Example** |
| int | NA | NA | int() | 5 |
| float | NA | NA | float() | 6.5 |
| string | Yes | No | ' ' or " " or str() | "this is a string" |
| bool | NA | NA | NA | True or False |
| list | Yes | Yes | [ ] or list() | [5, 'yes', 5.7] |
| tuple | Yes | No | ( ) or tuple() | (5, 'yes', 5.7) |
| set | No | Yes | { } or set() | {5, 'yes', 5.7} |
| dictionary | No | Keys: No | { } or dict() | {'Jun':75, 'Jul':89} |

# Data type and operator

* Str.spilit() – list of words
  + Str.lower()
* Dict.get() – return key value in dict

## dict

* Dict.items()
* Dict.keys()
* Dict.values()

# Function

* Example –

def cylinder\_volume(height, radius=5):

“””doc\_string”””

pi = 3.14159

return height \* pi \* radius \*\* 2

cylinder\_volume(10)

* we can still access the value of the global variable within function. But we cannot modify it
* Lambda expression can be used to create anonymous function -
  + cylinder\_volume = lambda height, radius=5: 3.14159\* height \* radius \*\* 2
* **Iterables** are objects that can return one of their elements at a time, such as a list. An **iterator**  is an object that represents a stream of data
* **Generators** are a simple way to create iterators using functions. You can also define iterators using **classes**, (\_\_init\_\_ , \_\_next\_\_ , \_\_iter\_\_[stop Iteration] ) which you can read more about [here](https://docs.python.org/3/tutorial/classes.html#iterators)
* def my\_range(x):

i = 0

while i < x:

yield i “”return values one at a time, and start where it left off each time it’s called””

i += 1

* **Note** - Generators are useful when the fully realized list would not fit in memory, or when the cost to calculate each list element is high and you want to do it as late as possible. But they can only be iterated over once.
* **Generator expressions and list comprehensions**

line\_list = [' line 1\n', 'line 2 \n', ...]

####### Generator expression -- returns iterator (surrounded by parentheses (“()”))

stripped\_iter = (line.strip() for line in line\_list)

####### List comprehension -- returns list (surrounded by square brackets (“[]”))

stripped\_list = [line.strip() for line in line\_list]

# Useful Standard Libraries

* [csv](https://docs.python.org/3/library/csv.html): very convenient for reading and writing csv files
* [collections](https://docs.python.org/3/library/collections.html): useful extensions of the usual data types including OrderedDict, defaultdict and namedtuple
* [random](https://docs.python.org/3/library/random.html): generates pseudo-random numbers, shuffles sequences randomly and chooses random items
* [string](https://docs.python.org/3/library/string.html): more functions on strings. This module also contains useful collections of letters like string.digits (a string containing all characters which are valid digits).
* [re](https://docs.python.org/3/library/re.html): pattern-matching in strings via regular expressions
* [math](https://docs.python.org/3/library/math.html): some standard mathematical functions
* [os](https://docs.python.org/3/library/os.html): interacting with operating systems
* [os.path](https://docs.python.org/3/library/os.path.html): submodule of os for manipulating path names
* [sys](https://docs.python.org/3/library/sys.html): work directly with the Python interpreter
* [json](https://docs.python.org/3/library/json.html): good for reading and writing json files (good for web work)

# Third Party Libraries and Package Managers

* Note: Always put third party import statements after standard library imports statements
* Pip manager – included in python – for installing 3rd party Libraries

## Useful Third-Party Packages

* [IPython](https://ipython.org/) - A better interactive Python interpreter
* [requests](http://docs.python-requests.org/) - Provides easy to use methods to make web requests. Useful for accessing web APIs.
* [Flask](http://flask.pocoo.org/) - a lightweight framework for making web applications and APIs.
* [Django](https://www.djangoproject.com/) - A more featureful framework for making web applications. Django is particularly good for designing complex, content heavy, web applications.
* [Beautiful Soup](https://www.crummy.com/software/BeautifulSoup/) - Used to parse HTML and extract information from it. Great for web scraping.
* [pytest](http://doc.pytest.org/) - extends Python's builtin assertions and unittest module.
* [PyYAML](http://pyyaml.org/wiki/PyYAML) - For reading and writing [YAML](https://en.wikipedia.org/wiki/YAML) files.
* [NumPy](http://www.numpy.org/) - The fundamental package for scientific computing with Python. It contains among other things a powerful N-dimensional array object and useful linear algebra capabilities.
* [pandas](http://pandas.pydata.org/) - A library containing high-performance, data structures and data analysis tools. In particular, pandas provides dataframes!
* [matplotlib](http://matplotlib.org/) - a 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments.
* [ggplot](http://ggplot.yhathq.com/) - Another 2D plotting library, based on R's ggplot2 library.
* [Pillow](https://python-pillow.org/) - The Python Imaging Library adds image processing capabilities to your Python interpreter.
* [pyglet](http://www.pyglet.org/) - A cross-platform application framework intended for game development.
* [Pygame](http://www.pygame.org/) - A set of Python modules designed for writing games.
* [pytz](http://pytz.sourceforge.net/) - World Timezone Definitions for Python

## Third-Party Library Documentation

* Third-party libraries publish their documentation on their own websites, and often times at <https://readthedocs.org/>