Glossary

General Python

Command	Description	Example
import <library>	Import the whole library	***import urllib***
from <library> import <function>	Import one function from the library	***from urllib import request***
myvariable = <value>	Initialise a variable with data or variables, functions return, etc.	***mystring = 'Hello World'***
print(variable)	Print the content of a variable	***print(mystring)***
len(variable)	Return the length of a variable	***mylength = len(mystring)***
dir(object)	Return the functions of an object	***print(dir(soup))***
type(variable)	Return the type of a variable	***print(type(mystring))***
list[<first_value>,<other_value>, <...>]	A list is a series of values. This structure has its own methods to modify it.	***mylist = ['value1','value2','value3']*** => Initialization ***print(mylist[2])*** => display the third value in the list ***mylist.sort()*** => sort the list ***mylist.append('value4')*** => add a value at the end of the list ***mylist.reverse*** => reverse the list values ***mylist.remove('value1')*** => remove a value
dict{<key>:<value>}	A dict is a key=value data structure. For example, Name=Alexandre This structure has its own methods to modify it.	***mydict = { 'firstname':'Alexandre', 'lastname':'Chevallier' }*** ***mydict.keys()*** => return keys list that you can iterate ***mydict.values()*** => return values list that you can iterate ***mydict.get('key')*** => return value for the specified key ***mydict['city']='Paris'*** => Add an element
tuple()	A tuple is a immutable list	***mytuple = ('value1', 'value2', 'value3', 'value1')*** ***mytuple.count('value1')*** => return the number of 'value1'
range(<start_number>,<end_number>)	***range*** provides a list of numbers constructed as needed. You can use in a loop	***range(3,8)*** => Return a list with ***[3,4,5,6,7]***

Glossary

General Python

Command	Description	Example
with open(<your_file>, <operation_mode>) as <handle_variable>:	Open file and return a corresponding file.	***with open('file.txt', 'w') as f:*** **f** is the file variable for operations **'file.txt'** is the file name **'w'** is the opening mode (w = write, r = read, a = append)

for <variable> in <sequence>:
    <code>
	Statements block code is executed for each item of a sequence variable. Sequence could be an iterator or list, dict, etc.
	Be careful to always indent after ':'
		for current_value in mylist:
		    print(current_value)
		
		To do an indexed loop:
		
		for i in range(0,10):
		    print(i)

Manipulating CSV file

Command	Description	Example
<write_handle_variable> = csv.writer(<opened_file_handle_variable>)	Return a writer handle responsible for converting the user's data into delimited strings on the given file variable opened before.	***writerhdl = csv.writer(filehdl)***
<read_handle_variable> = csv.reader(<opened_file_handle_variable>)	Read a file handle opened before. Returned CSV handle that can be manipulate with functions.	***csvhdl = csv.reader(filehdl)***

Glossary

Manipulating CSV file

Command	Description	Example
<write_handle_variable>.writerow(<row_to_write>)	Write the parameter to the writer's file handle, formatted according to the current dialect. For a CSV file the current dialect has to be a tuple or a list.	***writerhdl.writerow(mytuple)***

Glossary

Scraping and Parsing (BeautifulSoup)

Command	Description	Example
<variable> = urllib.request.urlopen(<URL>).read()	Download raw HTML data from a web page and return it to be used by other functions.	***raw_html = urllib.request.urlopen('https://www.google.fr').read( )***
<variable> = BeautifulSoup(<raw_html_variable>, <parser_type>)	Convert raw data in parsed HTML. Gives many functions to explore data.	***html = BeautifulSoup(raw_html, 'html.parser')***
<markups> = <beautifulsoup_variable>.find_all(<markup>)	Find all markups in the HTML and return them.	***tables = html.find_all('table')*** => Find all the markups <table> and return them in ***tables*** variable.
<one_item> = <beautifulsoup_variable>.find(<markup>, <optional_attribute>?{ id : 'thing'})	Find one item in the page and return it	***table = html.find('table')*** => Find one markup <table> and return it in ***table*** variable. ***table = html.find('table', { 'id' : 'table-class'})***

Data visualization (pandas)

Command	Description	Example
<dataframe> = pandas.read_csv(<filepath>)	Read a file handle to CSV and return it. This function call the csv.reader() function.	***mydf = pandas.read_csv('input.csv')***
<dataframe_variable> = DataFrame(<csv_data>)	Create and return a DataFrame from the given CSV data	***mydf = DataFrame(csv_content)***
<dataframe_variable>.<serie>.value_counts()	Return the item count of the serie	***mydf.mycolumn.value_counts()***

Glossary

Data visualization (pandas)

Command	Description	Example
<serie||dataframe>.plot(kind='<plot_kind>') ***<serie||dataframe>.plot.bar()*** ***<serie||dataframe>.plot.pie()*** ***<serie||dataframe>.plot.area()***	Draw plotting of the values in the serie. kind attribute is for selecting type of plot.	***mydf.mycolumn.plot.pie()***
<dataframe_variable>[['<a_column>', '<another_column']]	Select multiple columns from a data frame to do function on them	***mydf[['firstname', 'gender']]***
<dataframe_variable>.groupby(<column>)	Group items by column	***mydf[['firstname', 'gender']].groupby('gender')***
pandas.concat([<dataframe_1>, <dataframe_2>])	Concatenate Dataframes. Check the docs to see all the possibilities.	***df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2'], 'B': ['B0', 'B1', 'B2']})*** ***df2 = pd.DataFrame({'C': ['C0', 'C1', 'C2'], 'D': ['D0', 'D1', 'D2']})*** ***concatenedDF = pd.concat([df1, df2])***
pandas.merge([<left_dataframe>, <right_dataframe>], how='<method>', on=<columns>)	pandas provides a single function, `merge`, as the entry point for all standard database join operations between DataFrame. ***left*** and ***right*** are the DataFrames. ***how*** is the merge method (One of `'left'`, `'right'`, `'outer'`, `'inner'`). ***on*** are columns (names) to join on Check the docs to see all the possibilities.	***left = pd.DataFrame({'key': ['K0', 'K1'], 'A': ['A0', 'A1'], 'B': ['B0', 'B1']})*** ***right = pd.DataFrame({'key': ['K0', 'K1'], 'C': ['C0', 'C1'], 'D': ['D0', 'D1']})*** ***result = pd.merge(left, right, on='key')***

Glossary

Official documentation

When you're lost, first reflex should be to always to check the documentation:

- Jupyter: http://jupyter-notebook.readthedocs.io/en/latest/index.html and http://jupyter.org/
- Python 3: https://docs.python.org/3/
- Python Data Structures: https://docs.python.org/3/tutorial/datastructures.html?highlight=data%20structures
- Python Reading & writing files: https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files
- Open URLs (urllib library): https://docs.python.org/3/library/urllib.html?highlight=urllib#module-urllib
- CSV Files (csv library): https://docs.python.org/3/library/csv.html?highlight=csv#module-csv
- Parse HTML/XML (BeautifulSoup library): https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- Data analysis (pandas library): http://pandas.pydata.org/pandas-docs/stable/
- Merging and concat with pandas: https://pandas.pydata.org/pandas-docs/stable/merging.html
- Plotting with plot() (pandas library): https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html
- Visualization with pandas: https://pandas.pydata.org/pandas-docs/stable/visualization.html
- Plotting (matplotlib library): https://matplotlib.org/api/pyplot_summary.html