Intro to Scientific Computing Python Reference Sheet

Key: Python functions are in **bold**, sample placeholder text (where you would insert your own text) is in *italics*.

Day 2 - Data and Storage

Important Definitions

index	**The position of an element**
string	**Set of characters**
list	**Collections of values- can be various data types: integers,float,characters**
dictionary	**Data structure mapping a key to a related value**
f string	**Format a new string with a variable**
int	**Integers, such as 0, 1, 2, ...**
float	**Numbers in their decimal form, such as 0.0, 1.1, ...**
Boolean variables	**Binary variables, which can be either true or false**
Complex number	**A number expressed as a sum of a real part and an imaginary part**

Functions/Operations

+	**Addition**
-	**Subtraction**
*	**Multiplication**
/	**Division**
//	**Integer division**
**	**Exponent**
%	**Modulo Operator (Remainder)**
==	**Test if equal to**

>	**Test if greater than**
>=	**Test if greater than or equal to**
<	**Test if less than**
<=	**Test if less than or equal to**
!=	**Test if not equal to**
who	**See all variables and functions you created during the session**
# temperature in celsius	**Use # to add comments to your code**
print(“*Hello World!***”)**	**Print to console**
len(“*Hello World!***”)**	**Prints length of string**
variable_string **=** “*Hello World*!” *variable_integer* **=** *20* *variable_floatingpoint* **=** *22.2*	**Assign value to a variable (string, integer, or floating-point number)**
variable_string **+** “*My name is John!*”	**Append one string to existing string**
*variable_string***.replace(**“*Hello*”**,**“*Goodbye*”**)**	**Replace one word with another**
list.**append**(x)	**Adds an item ‘x’ onto end of a list**
*variable_string***.split()**	**Splits string into smaller strings**
*variable_string***.upper()**	**Makes entire string uppercase**
*variable_string***.lower()**	**Makes entire string lowercase**
*variable_string***.title()**	**Capitalizes each word in string**
fruits **= [‘***apple*’, ‘*orange*’, ‘*banana*’, ‘*mango*’**]**	**Create a list**
*fruits***[0]**	**Get first element of the list**
*fruits***[2:5]**	**Slicing the index 2 through 5 to get 2,3,4**
*fruits***[2:]**	**Index 2 and on**
*fruits***[:2]**	**Everything before index 2**
*fruits***[-1]**	**Last element of the list**
*fruits***.index(‘***orange***’)**	**Determines index of element in string**
*fruits***.count(‘***orange*’)	**Count the number of times a number or**

	string is in the list
fruits.**reverse()**	**Permanently reverses string**
Fruits **+** *fruits2*	**Append one list to another**
*my_keys***=**[‘*a*’, ‘*b*’, ‘*c*’]	**Keys for a dictionary (inputs)**
*my_values***=**[*1*,*2*,*3*]	**Values for a dictionary (outputs)**
alpha_numm_dict=**dict(zip(***my_keys*, *my_values***))**	**Joining the list**
alpha_num_dict2=**dict([**(‘*a*’, *1*),(‘*b*’, *2*),(‘*c*’,*3*)**])**	**Manually relating the keys to values**
*alpha_num_dict2***.get(**'*b*')	**Call up a value using key**
abs(*y*)	**Absolute value of a variable**
round(*y, 0*)	**Rounds a variable up or down to a specified number**
max(*x*)	**Largest value of a list, string, etc.**
min(*x*)	**Smallest value**
sum(*x*)	**Sum all values**
bool(*x*)	**Change a variable to a Boolean value**
complex(*x*)	**Change to complex number**
float(*x*)	**Change to a floating point**
int(*x*)	**Change to an integer**
str(*x*)	**Change to a string**
print(**type**(*x*))	**Check the type of x**

Day 3 - Loops and If Statements

Functions

for *x* **in** *list*	**Creates for loop that runs through 'list'**
for *i* **in range(***start*, *stop*, *step size***)**	**Creates a for loop with specific start and stop values, and specifies the step size between values**

if (*input==something*):
    print(*output*)
else:
    print(*output*)
	If/else statement, can choose between different outputs based on the input

if (*input==something*) **and** (*input>=something*):
    print(*output*)
else:
    print(*output*)
	If/else statement that tests for two conditions and chooses outputs based on if BOTH conditions in the input are true

if (*input==something*) **or** (*input>=something*):
    print(*output*)
else:
    print(*output*)
	If/else statement that tests for two conditions and chooses outputs based on if ONE OF TWO conditions in the input are true

if (*input==something*):
    print(*output*)
elif (*input <something*):
   print(*output*)
...
	Use in addition to if/else statement to test for another condition

Day 4 - Functions

Definitions

argument	**A value provided to a function**
docstring	**Describes what the function does**

Functions

def *function*(*argument*)	**Defines/names a function**
'''*Welcomes the user with input name*'''	**Sets a description for the function purpose (docstring)**
function.__doc__	**Calls upon the function description set in docstring**
print(f"*Welcome* {*argument*}!")	**Creates f string**
joined = **str**(*argument*)	**Creates string from argument**
return(*string_name*)	**Returns a string in a specified format**
help(*function*)	**Get information on a function**
Variable_name = *function*	**Save function output to variable**
dictionary_name= {*key1*:*value1*, *key2*:*value2*}	**Set dictionary**
function(**dictionary_name)	**Input dictionary into function**
*args	**Accepts any number of positional arguments in a function**
**kwargs	**Tells the function you are going to pass an arbitrary number of keyword arguments that you haven't defined beforehand**

Day 5 - Intro to Numpy

Definitions

NumPy	**Short for Numerical Python, this is a library containing built-in functions used for scientific computing**

Functions

Import numpy as np	**Import is used to import a library. This specific line imports numpy and gives it the nickname np**
np.arange(*stop*)	**Creates an array starting from 0 and listing integers until stop**
np.arange(*start*,*stop*)	**Creates an array from start to stop value**
np.arange(*start*,*stop*,*step*)	**Creates an array listing integers from 'start' to 'stop' with indicated 'step' between values**
np.linspace(*start*,*stop*,*num*)	**Similar to arrange() function, except can output floats -> num represents how many entries you would like- default is 50**
np.array()	**Creates a Numpy array**
np.zeros(*x*)	**Creates an array full of zeros with x number of elements**
np.ones(*x*)	**Makes an array full of ones with x number of elements**
print(*array*.**size**)	**Prints size of array**
print(*array*.**shape**)	**Prints shape of array**
np.random.uniform(**size**=*x*)	**Creates a 1D array with 'x' random elements**
np.random.randn()	**Returns an array of random samples from the Gaussian standard distribution**
print(*array*.**dtype**)	**Determines data type of elements in array**
np.std()	**Finds standard deviation of numpy array**
np.percentile()	**Finds percentile of numpy array**

np.mean()	**Finds mean of numpy array**
np.median()	**Finds median of numpy array**
np.sum()	**Summation of numpy array**
np.exp()	**Takes e^ of all elements in array**
np.power(*array*,*power*)	**Raises all elements in specified array to a specified power**
np.loadtxt(**loaddir**+'*file name*', **delimiter**=',')	**Asks python to output data that was loaded**
[[*element1*,*element2***]]**	**Specify elements of 1D array**
[[*row1*,*row3***]**,**[***col4*,*col6***]]**	**Specify coordinates of 2D array**
print(*arr***[::-1]**) OR **print**(**np.flip**(*arr*))	**Reverse the order of an array**
sum_array **=** *first_array* **+** *second_array*	**Add together two arrays- must be the same length**
*vec_1***.dot**(*vec_2*)	**Dot product of two vectors (define the vectors first)**

Day 6 - Intro to Plotting

Definitions

Pyplot	**Library used for plotting data (For these definitions we will use the nickname ‘plt’)**
fig	**A variable representing an entire set of figures in matplotlib.pyplot library**
ax	**A variable representing a single plot within fig**
mu	**Mean**
sigma	**Standard deviation**

Functions

datadirectory **= '.*/datalocation/*'**	**Sets directory to retrieve data from**
*Data_out_directory***= '.*/outputlocation/*'**	**Sets the directory for storing data**
print(*data_out_directory***)**	**Checks where data is being stored**
plt.subplots()	**Used to create multiple plots at once**
plt.show()	**Show figure**
ax.plot(*x*,*y*)	**Create a line plot within the subplot with x and y values**
ax.scatter()	**Create a scatter plot**
ax.**set_title**(*‘Title’*, **fontsize**=*num*)	**Create plot title**
ax.**set_xlabel**(**“***Label***”**, **fontsize**=*num*) **ax**.**set_ylabel**(**“***Label***”**, **fontsize**=*num*)	**Create axis labels**
ax.legend(**loc =** “*upper right*”)	**Creates a legend with a location**
ax.set(**xlim**=(*min*,*max*), **ylim**=(*min*, *max*))	**Sets axis limits with assigned minimum and maximum values**
ax.tick_params(**axis** = “*both*”, **labelsize** = *num*)	**Increases font sizes for the ticks on both axes**
y1 = **np.sin**(x) *y2* = **np.cos**(x)	**Creates a plot for a sin and cosine curve respectively**

plt.tight_layout()	**Makes sure that plots within subplot are not overlapping**
plt.savefig(*data_out_directory*+**'***filename***'**)	**Save figure to a specific destination**
plt.scatter(x, y, s = *datatype*, c = *x*, alpha = *number*)	**Sets a scatter plot with c being what is colored and s determining the shape. Alpha controls the opacity of the points- can be a number between 0(most faint) and 1(most dark)**
ax.hist(*data*, *# of bins*)	**Create histogram**
plt.colorbar()	**Adds colorbar to plot**
np.random.lognormal(*mean,sigma,size*)	**Creates an array of random values from a lognormal distribution**
np.random.normal(*loc,scale,size*)	**Creates an array of random values from a normal distribution**
ax[0,0] **ax**[0,1] **ax**[1,0] **ax**[1,1]	**Sets up a top left plot** **Sets up a top right plot** **Sets up a bottom left plot** **Sets up a bottom right plot**

Day 7-9: Pandas

pd.DataFrame()	**Create an empty data frame**
dataframe.**append({**‘*Listname* **’:** *addedrow_name* **’}, ignore_index =** *True/False*)	**Add a row to your data frame**
pd.read_csv(*csv file location*)	**Read in a csv data file**
filename.**head()**	**View top lines of a data frame**
filename.**tail()**	**View last lines of a data frame**
filename.**columns()**	**View column names**
filename.**index()**	**View row/index names**
filename.**set_index()**	**Set the index**
filename.**info()**	**Gives information for each column and the total memory usage**
filename.**sort_values(by =** *“Value”*, **ascending =** *True/False*, **inplace =** *True/False***)**	**Sort data frame by specified value**
filename.**loc[***‘rowstart’:’rowend’,’columnstart’:’ columnend’***]**	**Select specific data by defining the value**
filename.**iloc[***rowstart:rowend*, *[column1,column2,column3]***]**	**Select data, indexing with the integer location**
filename.**drop**([*'value names', value names’, 'value names'*], **axis=**0/1**)**	**Remove certain parts of a data set**
filename.**rename(columns/rows=**{*'original name':'new name'*},**inplace=**True**)**	**Rename certain columns or rows**
dataname.**shape**	**Returns dimensions of the data frame**
pd.merge(*Table 1, Table 2*)	**Merge tables from two data sets**
dataname.**describe**()	**Returns overview of the data**
dataname.**add_prefix(**‘{}_’**.format(**“*x*”))	**Adds a prefix to the column names**
dataname.**add_suffix(**‘_{}’**.format(**“*y*”))	**Adds a suffix to the column names**
map	**Translates or maps values to other values**

	using dictionaries
apply	**Applies a function on a series or a dataframe**
applymap	**Applies of a function to all the elements in a dataframe (all the cells)**
from functools **import** reduce	**Imports a library that allows you to reduce dataframes to list and combine them**
concat	**Aka concatenate, allows you to combine dataframes that have columns with the same names but no data similarity**
from glob **import** glob	**Imports a library that allows you to read all the files in a directory and append them together without having to read in each file**
dataname.**pivot_table**(**values=** ‘_’ , **index=** ‘_’, **columns=** ‘_’, **aggfunc=** ‘_’)	**Creates a pivot table, specifies what you want in the index and columns and how you want to aggregate your values**
sns.regplot()	**Shows a scatterplot with a correlation line**
sns.heatmap()	**Creates a heatmap out of specified data set**
dataname.**corr()**	**Creates a correlation table**
sns.pairplot	**Creates a pairplot, which allows you to see relationships between variables**
sns.swarmplot	**Creates a swarmplot, used for categorical data**
sns.violinplot	**Creates a violinplot, which can also be used for categorical data**

Other Resources

- https://docs.python.org/3/tutorial/
  - Further information on Python basics (operators, lists, if statements, for loops, etc.)
- NumPy: Absolute Basics
  - Beginner's look at NumPy including basics of creating and manipulating arrays
- NumPy Quickstart Guide
  - Quick overview of NumPy arrays
- Visual Guide to NumPy by Lev Maximov
  - A unique, illustrated look at NumPy
- Pyplot Tutorial
  - Introduction to Pyplot
- Pandas Cookbook by Julia Evans
  - Dive deeper into Pandas by working through some examples