Python Cheat Sheet

Syntax

A line is a statement
    Lines indented on the same level
    Are code blocks
        Code blocks can be nested
    # hashtags make comments

Assigning variables
var_name = object

Importing Modules
import module
import module as md
from module import object

Semantics (Objects)

Basic Data Types

Type	Content	Conversion
String	Text	str()
Integer	Whole number	int()
Float	Decimal	float()
Boolean	True/False	bool()

Numerical operators

symbol	operation	description
+	add	returns the sum of two numbers
-	subtract	returns the difference between two numbers
*	multiply	returns the product of two numbers
/	divide	returns the result of the division between two numbers
//	floor division	returns the result of division, excluding fractions
%	remainder	returns the remainder of the division
**	power	exponentiates one number by the other

Order of operations
1 **P** Parentheses
2 **E** Exponentiation
3 **M,D** Multiplication and Division
4 **A,S** Addition and subtraction,

Operators with the same preference are resolved from left to right.

Useful built-in functions:
type() help() max() min() sum() len()
print() set() range()

Strings 'in quotes' "in quotes" or '''in quotes'''

String operators
symbol	operation
+	concatenate
*	repeat
in	checks whether a substring is in the string
not in	checks whether a substring is not in the string

String indexing: str[index] position 1 = index 0
String slicing str[first_index: last_index+1] (up to, but not including)

String methods
Method	Action
string.lower()	makes string lowercase
string.upper()	makes string uppercase
string.split('separator')	returns a list of substrings, split by a separator (space, if blank)
string.find('substring')	gives you first position of a substring in a string
string.replace('old','new')	replaces all occurences of search string with the new string.
string.lstrip()	removes whitespace on left
string.rstrip()	removes whitespace on right
string.strip()	removes whitespace on both sides
line.startswith('substring')	asks whether string starts with substring, giving true/false response

Collections

Structure	Description	Notation
List	A changeable, ordered collection. Elements are accessed by position.	[,]
Dictionary	A changeable, unordered collection. Elements are accessed by name (key).	{k:v,}
Tuples	An unchangeable, ordered collection (like an unchangeable list).	(,)
Set	An unordered collection of unique values.	{,}

my_list = ['value A', 'value B', 1, 'value D']
list_name **=** []
or
list_name **=** list()

list_name[0] list_name[a:b]

Method	Action
list.append()	Adds a value to the end of the list
list.extend()	adds several values from an iterable object (such as a list) to end of list
list.insert()	adds a value to another place in list
list.remove('value')	Removes the first item with the content 'value' from the list
list.reverse()	reverses the order of list
list.sort()	sorts based on values
list.count('value')	counts number of time 'value' is in list

dict_name = {'key1': value1,'key2': value2, 'key3': value3 }

```python
dict_name = {}
```

Alternatively, you can also use the dict() command.

```python
dict_name = dict()

dict_name['key1']
```python
dict_name['newkey']='newvalue'
```

Method	Action
len(d)	Return the number of items in the dictionary d
del d[key]	Remove d[key] from d. Raises a KeyError if key is not in the map.
clear()	Remove all items from the dictionary.
pop(key[, default])	If key is in the dictionary, remove it and return its value, else return default. If default is not given and key is not in the dictionary, a KeyError is raised.
update([other])	Update the dictionary with the key/value pairs from another collection, overwriting existing keys. Return None.
dict.items() dict.keys() dict.values()	These methods respectively return a list of the key-value pairs, keys or values in the dictionary. These are useful for iterating through the dictionary with loops (Lesson 06).

my_tuple = ('value A', 'value B', 1, 'value D')

Syntax	Name		Action
set1 | set2	Union	<img src="Other_files/SetUnion.png" width='200'>	Return a new set with elements from the set and all others.
``set1 & set2``	Intersection	<img src="Other_files/SetInter.png" width='200' >	Return a new set with elements common to the set and all others.
``set1 - set2``	Difference	<img src="Other_files/SetDiff.png" width='200'>	Return a new set with elements in the set that are not in the others.
``set1 ^ set2``	Symmetric Difference	<img src="Other_files/SetSymDiff.png" width='200'>	Return a new set with elements in either the set or other but not both.

05 Conditionals

Comparison Operators
Operator	Meaning
\>	greater than
\>=	greater than or equal to
==	equal to **(!!)**
<	less than
<=	less than or equal to
!=	not equal to

Boolean Operators
Operator	For True value:
and	Two conditions must be simultaneously true
or	One or another condition must be true
not	Negate a condition

Identity operators: is / is not
Membership Operators: in / not in

if
```
    if <condition>:
    code to run when condition is True

if condition :
    code A runs when condition is True
else :
    code B runs when condition is False
```
```

```
if condition1 :
    code A runs when condition1 is True
elif condition2 :
    code B runs when condition1 is False and condition2 is True
else:
    code C runs when condition1 and condition2 are False
```

ifs can be nested

while loops

```python
i = 5

while <condition based on i>:
    code
    i= i-1

other code...
```

for loops

break
continue
pass

for loops

```
for i in <set>:

    code

other code
```

range(len)

07 Files

```
myfile= open('filename.txt', 'mode')
```

check working directory

```
import os
OS.getcwd()
```

```
myfile= open('filename or path', 'mode')
data= myfile.read()
theRaven.close()

with open('file path', 'a') as myfile:
   data = myfile.read()

myfile= open('filename', 'mode')
for line in myfile:
    print(line)

file.write()
```

Pandas - DataFrames & Series for Tabular Data

```python
import pandas as pd

data = pd.read_csv("filepath or name")
```

08 Functions

```python
def function_name(parameter1, parameter2):
    "docstring"
    code depending on parameter1 and parameter2
    #Can be multiple lines or nested blocks

def function_name(parameter1='text', parameter2=1):
    "docstring"
    code depending on parameter1 and parameter2
    #Can be multiple lines or nested blocks

def function_name(parameter1='text', parameter2=1):
    "docstring"
    code depending on parameter1 and parameter2
    #Can be multiple lines or nested blocks
    return <result>
```

09-10 Pandas

```python
file = pd.read_csv('filename.csv')

file = pd.read_excel('file.xlsx', sheet_name='sheetname')
```

df.head() and df.tail()
df.columns
df.dtypes

```python
df = pd.read_csv('filename.csv', index_col='col_name')
```

To specify an index **after creating** the DataFrame, use:

```python
df = df.set_index('col_name')
```

Accessing info
One dimensional: df['column_name'] df[['column_1','column_2']]

df[['column_1','column_2']].copy()

conditional selections df[df['columname']>0] (|) and and (&)

df.column_name (not recommended)

Two-Axis selection: .loc[] using labels
df.loc[rows,columns]

single cell: df.loc[row_index,'columname']
group of cells df.loc[[row1,row2,row3],['column1','column5']]
ranges: df.loc[row1:row3, column1:column5]
conditionals df.loc[df['colname'] < 1, :]

df.iloc[] using integers

Saving to CSV:

df.to_csv('filename.csv',encoding='',index=**False**)
index=False to drop index (useful if index was autocreated)

Saving to Excel:

df.to_excel('filename.xlsx',sheet_name='sheet')

Saving to JSON:

df.to_json('filename.json', orient='records')

Manipulating DFs
sorting: df.sort_values()
uniques: df['column_name'].unique()
count: **df.value_counts()**
sum(),mean(), median(),max(), min()
df['newcolumn'] = df['column1'] + df['column2']
df.transpose()
df.to_dict()
df.to_numpy()
df.to_json()
df.to_csv()
adding columns: df['newname'] = x
dropping columns: df = df.drop(columns = 'column_name')
dropping rows: df = df.drop(index = [index numbers])
melting: df.melt(id_vars = [list of variables to remain unchanged], value_vars = [list of variables to be made into values], var_name = 'nameforvariable', value_name = 'nameforvalues')

Joining
Types of Joins	
Left	Keeps all rows in the left table, and will join these to rows in right table with a matching key.
Inner	Only gives us rows with matching in both tables (will also drop Left-only rows)
Outer	Returns all rows from both tables, regardless of if they have a match.
Right	Returns all rows in righ table, joined to any rows with a matching key from the left table.

df3 = df1.merge(df2, how='left', on=Ôkey_columnÕ_name)
or left_on, right_on

Appending rows:

df.append()

dfslist =[DataFrame1, DataFrame2, Dataframe3]
newdf = pd.concat(dfslist)

Plotting
df.plot(kind='line', x='column', y=['column1', 'column2'])
Line: df.plot.line(x='column',y=['column1', 'column2'])
Bar: df.mean().plot.bar() vertical df.mean().plot.barh() horizontal
Histogram: df.hist() or df.plot.hist() (different results)
Box: df.plot.box()