PYTHON 3

CHEET SHEET BY @CHARLSTOWN

CREATOR: GUIDO VAN ROSSUM
YEAR: 1991
LATEST VERSION (2019): 3.7.4
TOP LIBRARIES: TENSORFLOW, SCIKIT-LEARN, NUMPY, KERAS, PYTORCH, PANDAS, SCIPY

ELEMENTAL LIBRARIES import *

```
import datetime                 >> Date format
import math                     >> Math operations
import random                   >> Random tool
import itertools                >> Iteration tool
import re                       >> Regular exprs.

from lib import function        >> import function I
lib.function()                  >> import function II
from lib import *               >> import all funct.
dir(math)                       >> Show all functions
import library as lb            >>Library shortcut
```

STRING OPERATORS "Hello world"

```
str(50)                         >> String conversor
var.isalpha()                   >> alphanumeric
var.upper()                     >> VAR
var.lower()                     >> var
var.title()                     >> Mayus/word
var.split()                     >> String => Lista
" ".join(list)                  >> Lista => String
"".replace('A', 'B')            >> ABC => BBC
list.find("A")                  >> First letter index
"  A  ".strip()                 >> No gaps
("A %s C %S") %("B", "D")       >> ABCD
"{} B {}".format("A", "C")      >> ABC
"AB\"CD"                        >> Include "
"\n"                            >> New line
"\t"                            >> Tabulator
```

STANDARD FUNCTIONS

```
var = input('question?')        >>Input raw data
```

INT/FLOAT OPERATORS 3 + 2.56

```
type()                          >> Returns type
A//B                            >> Returns ratio
A&B                             >> Returns reminder
divmod(A, B)                    >> Ratio/reminder
len()                           >> Returns lenght
max()                           >> Max. value
min()                           >> Min. value
abs()                           >> Absolute value
Pow()       A**B                >> Power
round(A, 3)                     >> 0.123
sum()                           >> Mass addition
```

DEF METHOD() & LAMBDA:

```
def method(variables):          >> As callable library
    return (var1, var2)

var = lambda x: x+1             >> As an expression
var(n)
```

LIST OPERATORS [A, B, 5, True]

```
list(var)                       >> List conversor
list(range(0,10,2))             >>02468
list.reverse()                  >> Reverse the list
lst[idx]                        >> Element index
lst[idx] = "item"               >> Change item
lst.append('item')              >> Add item
lst[-5:]                        >> Slicing list
list.index("B")                 >> Position index
list.insert(0, A)               >> AABC
list.remove(B)                  >> AC
list.count(A)                   >> A frequency
list.sort()                     >>Sort in same list
sorted(lst)                     >>Sort in new list
lst.pop()                       >>Last item
zip(lst_1, lst_2)               >>AA, BB, CC
```

LIST COMPREHENSIONS [i for i in lst]

```
LIST COMPREHENSION IN LISTS I
lst_A = [i for i in lst_B if i < 0]

LIST COMPREHENSION IN LISTS II
lst_A = [i if i < 0 else i-5 for i in lst_B]

LIST COMPREHENSION IN LISTS III
lst_A = [[i+1 for i in x] for x in lst_B]
```

DICT. OPERATORS {ky1: "A", ky2: list}

```
dic[key] = val                  >> Add a key
dic.update({ky: v, ky: v})      >> Add multiple keys
dic[key] = val                  >>Overwrites value
dic[key]                        >> Extracts a value I
dic.get(key)                    >> Extracts a value II
dic.get(key, FVal)              >> Extracts a value III
dic.pop(key)                    >> Delete K and V I
del dic[k]                      >> Delete K and V II
dic.keys()                      >> Keys List
dic.values()                    >> Values list
dic.items()                     >> Returns K and V
key in dict                     >> Checks key
dict(zip(lst_1, lst_2))
```

DICT COMPREHENSIONS {k:v for k, v in dict}

```
{key: value for key, value in dict}
{k, v for k, v in dict.items()}
```

TIMING THE CODE

```
import time

time.time
time.sleep(s)
time.localtime()
```

NUMPY LIBRARY

BASIC FUNCTIONS — import numpy as np

IMPORTING CSV FILES:
np.getfromtxt('f.csv', delimiter = ',')

- - - - - - - - - - - - - - - - - - - -

np.mean(lst)	>> Average
np.sort(lst)	>> Sort the list
np.median(lst)	>> Median
np.percentile(lst, n)	>> Percentil n%
np.std(lst)	>> Standard devi.
np.mean(mtx < n)	>> Conditions
np.var()	>> Variance

MATRIX — import numpy as np

mtx = np.array(lst1, lst2, lst3)

np.mean(mtx)	>>Total data mean
np.mean(mtx, axis = 0)	>> Columns mean
np.mean(mtx, axis = 1)	>> Rows mean

PANDAS LIBRARY

DATA FRAMES — import pandas as pd

IMPORTING CSV FILES:
pd.read_csv('f.csv')

- - - - - - - - - - - - - - - - - - - -

pd.DataFrame(Dict)	>> Create a DF I
columns = [list]	>> Create a DF II

- - - - - - - - - - - - - - - - - - - -

df.head(n)	>> shows first n rows
df.info()	>> entries and data

EXTRACTING COLUMNS AND ROWS

df.column.values	>> Extract column
df['colum']	>> Extract multiple clmns
df[[c1, c2]]	>> Extracts columns as df
df.iloc[index]	>> Extracts the Row by idx
df.iloc[i:i]	>> Extracts Rows as df
df[df.c1 > n]	>> Extracts Row by cond. I
df[df.c1.condition]	>> Extracts Row by cond. II

- - - - - - - - - - - - - - - - - - - -

df.reset_index()	>> Reset the index
drop = True	>> Without inserting it
inplace = True	>> Modify overwriting

ADD AND RENAME COLUMNS

df[columns] = list >> Adding a column

- - - - - - - - - - - - - - - - - - - -

RENAMING COLUMNS:
df.columns = list >> Modifying names
df.rename(columns = {old:new}, inplace=True)

APPLY MODIFICATIONS & LAMBDA

df[col] = df.c1.apply()	>> Modify column
df[col] = df.c1-apply(lb)	>> lb = lambda

- - - - - - - - - - - - - - - - - - - -

lb = lambda row: row.c1 >> Lambda in rows
df[col] = df.apply(lb, axis = 1)

PANDAS AGGREGATES (COMANDS)

df.c1.unique()	>> Extracts the set
df.c1.nunique()	>> Extracts len(set)
df.c1.mean()	>> Average of the column
df.c1.median()	>> Median of the column
df.c1.std()	>> Standard deviation
df.c1.max()	>> Max number
df.c1.min()	>> Min number
df.c1.count()	>> len of the set

P.A. (GROOPING)

df.groupby(c1).c2.mean()*	>> Groups c1
df.groupby(c1).id.count()*	>> Counter
df.groupby(c1).c2.apply(lb)*	>> lambda
df.groupby([c1,c2]).c3*	>> Multiple g
* > .reset_index()	>> To reset df

- - - - - - - - - - - - - - - - - - - -

df.pivot(columns = c2, index = c1, values = v)

MERGE METHODS

pd.merge(df1, df2*)	>> Merge method I
df1.merge(df2)	>> Merge method II

df1.merge(df2).merge(df3)
* > how = 'outer' \ 'inner' \ 'right' \ 'left'

- - - - - - - - - - - - - - - - - - - -

pd.merge(df1, df2, left_on = c1, right_on = c3)*
>> To merge 2 data frame with same column
* > suffixes = [name1, name2]

- - - - - - - - - - - - - - - - - - - -

pd.concat([df1, df2])

SORTING METHODS

df.sort_values(by = ['c1', 'c2'], ascending = False)