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

STRING OPERATORS “Hello world”

str(29)	> Int/Float to string
len(“string”)	> Total string length
“My” + “age is:” + “28”	> Sum strings
“Hey!” * 3	> Repeat string by 3
“a“ in “chartlstown“	> True if str in str
‘letters‘.isalpha()	> Check only letters
‘string‘.upper()	> STR to CAPS-CASE
‘string‘.lower()	> STR to lower-case
‘string‘.title()	> First letter to CAPS
list(“string“)	> Letters to list
‘my string‘.split()	> Words to List
“”.join(list)	> List to String by “”
“AB“.replace(“A”, “B”)	> Replace AB > BB
string.find(“A”)	> Index from match
“ A “.strip()	> No leading spaces
f”My age: *{28}*”	> Insert in string
“”My age: {}”.format(28)	> Old insert in string
“AB\”CD”	> Include symbol “
“\n”	> New line in string
“\t”	> Tabulator in string
var = input(‘question?’)	>Input string form

INT/FLOAT OPERATORS 3 + 2.56

int(“25“)	> String to integer
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(5, 2)	> 5 powered by 2
5**2	> 5 powered by 2
round(A, 3)	> Round 3 decimals
sum(list)	> Sum all list items

LOOPS

```
for item in list:          > for loop
      print(item)          > Iterate by items

while limit <= 5:          > while loop
      limit += 1           > Iterate by condition
```

LIST OPERATORS [“A”, “B”, 5, True]

len(list)	> Count items in list
list(range(0,10,2))	> List from range
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)	> Insert item by index
list.remove(5)	> Remove element
list.count(A)	> A frequency
list.sort()	> Sort in same list
sorted(lst)	> Sort in new list
lst.pop()	> Last item
list(zip(lst_1, lst_2))	> Entwine pair of lists
enumerate(list)	> Add index to list

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, DefVal)	> 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))	> Pair lists to Dict.

LIST & DICT COMPREHENSIONS

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

LIST COMPREHENSION NESTED
```
lst_A = [i if i < 0 else i-5 for i in lst_B]
```

LIST COMPREHENSION NESTED
```
lst_A = [[i+1 for i in x] for x in lst_B]
```

DICT COMPREHENSION
```
{key: value for key, value in dict.items()}
```

FUNCTION & LAMBDA:

```
def switch(in1, in 2):          > Code as function
      return (in2, in1)         > Switch variables
switch(“a”, “b”)                > Run function on a,b

plus_one = lambda x: x+1        > Code as expression
plus_one(5)                     > Number plus 1
```

TIMING THE CODE

time.time()	> Get elapsed time
time.sleep(s)	> Pause code s secs.
time.localtime()	> Get local time

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)