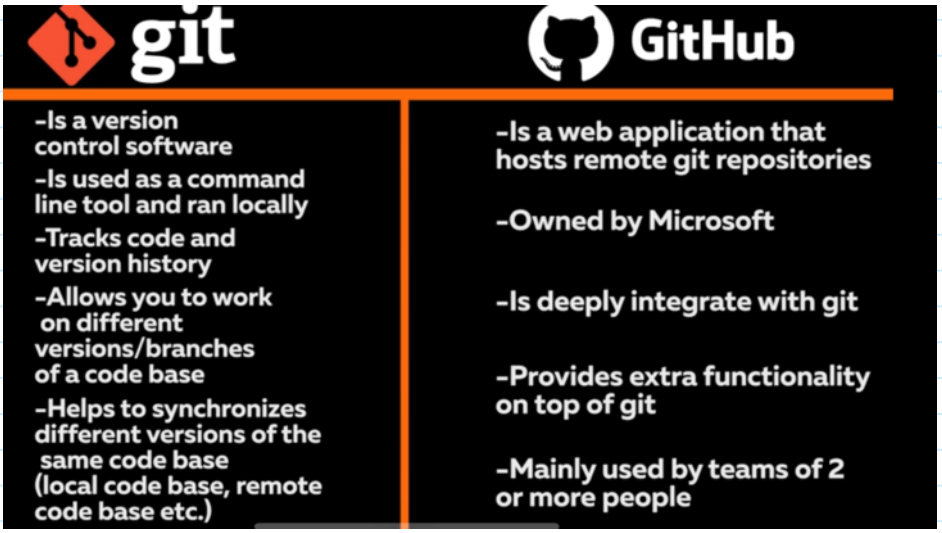
# Basic Python

Dr. Mejdal Alqahatni

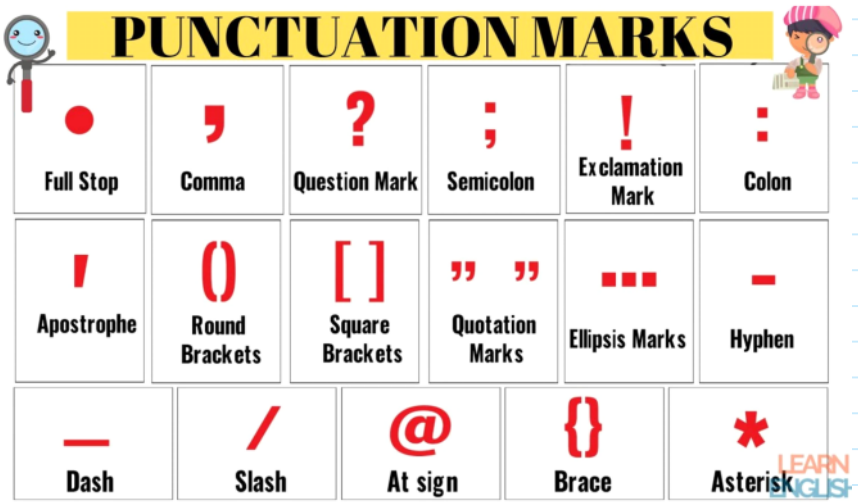
Thursday, October 21, 2021 5:28 PM

- **Git** and **Gith**ub
    - Objective → Sharing and track changes
    - Differences → Git (local) vs Github (cloud)




- How to make clone?
    1. Download Git Software
    2. Create file named as SDAIA on your Desktop
    3. Copy file location from your local device
    4. Go to Anaconda and type " cd file location"
    5. Go to Github and clone the repo.
    6. Go to Anaconda and write " git (or conda) clone link"

- Punctations:



- Summary of data Types:

## Main data types

**boolean** = *True* / *False*
**integer** = 10
**float** = 10.01
**string** = "123abc"
**list** = [ value1, value2, ... ]
**dictionary** = { key1:value1, key2:value2, ...}

- Summary of operations:

## Numeric operators

| | |
|---|---|
| + | addition |
| - | subtraction |
| * | multiplication |
| / | division |
| ** | exponent |
| % | modulus |
| // | floor division |

## Comparison operators

| | |
|---|---|
| == | equal |
| != | different |
| > | higher |
| < | lower |
| >= | higher or equal |
| <= | lower or equal |

## Boolean operators

| | |
|---|---|
| **and** | logical AND |
| **or** | logical OR |
| **not** | logical NOT |

- Summary of list operations and methods:

## List operations

| | |
|---|---|
| **list = []** | defines an empty list |
| **list[i] = x** | stores x with index i |
| **list[i]** | retrieves the item with index I |
| **list[-1]** | retrieves last item |
| **list[i:j]** | retrieves items in the range i to j |
| **del list[i]** | removes the item with index i |

## List methods

| | |
|---|---|
| **list.append(x)** | adds x to the end of the list |
| **list.extend(L)** | appends L to the end of the list |
| **list.insert(i,x)** | inserts x at i position |
| **list.remove(x)** | removes the first list item whose value is x |
| **list.pop(i)** | removes the item at position i and returns its value |
| **list.clear()** | removes all items from the list |
| **list.index(x)** | returns a list of values delimited by x |
| **list.count(x)** | returns a string with list values joined by S |
| **list.sort()** | sorts list items |
| **list.reverse()** | reverses list elements |
| **list.copy()** | returns a copy of the list |

- Built in functions:

## Built-in functions

| | |
|---|---|
| **print(x, sep='y')** | prints x objects separated by y |
| **input(s)** | prints s and waits for an input that will be returned |
| **len(x)** | returns the length of x (s, L or D) |
| **min(L)** | returns the minimum value in L |
| **max(L)** | returns the maximum value in L |
| **sum(L)** | returns the sum of the values in L |
| **range(n1,n2,n)** | returns a sequence of numbers from n1 to n2 in steps of n |
| **abs(n)** | returns the absolute value of n |
| **round(n1,n)** | returns the n1 number rounded to n digits |
| **type(x)** | returns the type of x (string, float, list, dict ...) |
| **str(x)** | converts x to string |
| **list(x)** | converts x to a list |
| **int(x)** | converts x to a integer number |
| **float(x)** | converts x to a float number |
| **help(s)** | prints help about x |
| **map(function, L)** | Applies function to values in L |

- Conditional statements:

**Conditional statements**

```
if <condition> :
    <code>
else if <condition> :
    <code>
...
else:
    <code>

if <value> in <list>:
```

  - Some rules for Boolean:
    - x=2 , bool(x) → TRUE
    - x=' ', bool(x) → FALSE
    - x=None , bool(x) → FALSE

    - `0` converts to `False` , all other numbers convert to `True`
    - `''` converts to `False` , all other strings convert to `True`
    - `None` converts to `False`

  - **AND, OR and NOT logic gates:**

| A | B | A AND B | A OR B | NOT A |
|---|---|---|---|---|
| FALSE | FALSE | FALSE | FALSE | TRUE |
| FALSE | TRUE | FALSE | TRUE | TRUE |
| TRUE | FALSE | FALSE | TRUE | FALSE |
| TRUE | TRUE | TRUE | TRUE | FALSE |

- For loop and while -loop
  - For → list or Range (1,2,..)
  - While → Condition

## Loops

```
while <condition>:
    <code>

for <variable> in <list>:
    <code>

for <variable> in
range(start,stop,step):
    <code>

for key, value in
dict.items():
    <code>
```

- Ex. for i in range(len(x)):
  - Range (n2,n1, n)

- Break vs continue  vs pass (i.e., use with for-loop and while-loop)
  - Break → exit
  - Continue → Skip
  - Pass → continue (do nothing)

## Loop control statements

| | |
|---|---|
| **break** | finishes loop execution |
| **continue** | jumps to next iteration |
| **pass** | does nothing |

- Function
  - Parameters :
    - Position (must have a value) → order  important
    - Keyword (optional to have value (default value))→ order not important
    - Position (value)+  Keyword  → position (first)

## Functions

```
def function(<params>):
    <code>
    return <data>
```

- String operations and methods:

| String operations | |
|---|---|
| **string[i]** | retrieves character at position i |
| **string[-1]** | retrieves last character |
| **string[i:j]** | retrieves characters in range i to j |

| String methods | |
|---|---|
| **string.upper()** | converts to uppercase |
| **string.lower()** | converts to lowercase |
| **string.count(x)** | counts how many times x appears |
| **string.find(x)** | position of the x first occurrence |
| **string.replace(x,y)** | replaces x for y |
| **string.strip(x)** | returns a list of values delimited by x |
| **string.join(L)** | returns a string with L values joined by string |
| **string.format(x)** | returns a string that includes formatted x |

- Find vs index (to find location)

| | Find | Index |
|---|---|---|
| Use | String | String, list, and tuples |
| Output (if it is not founded) | -1 | Error |
| If-condition | Yes | No |

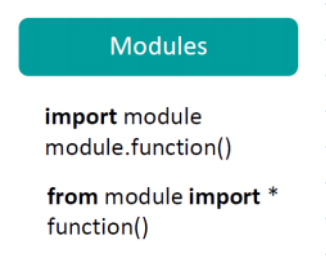
- x.strip ("A") → remove A from first and last value from the list , x.strip () → remove spaces from first and last value.

- Dictionary operations and methods:

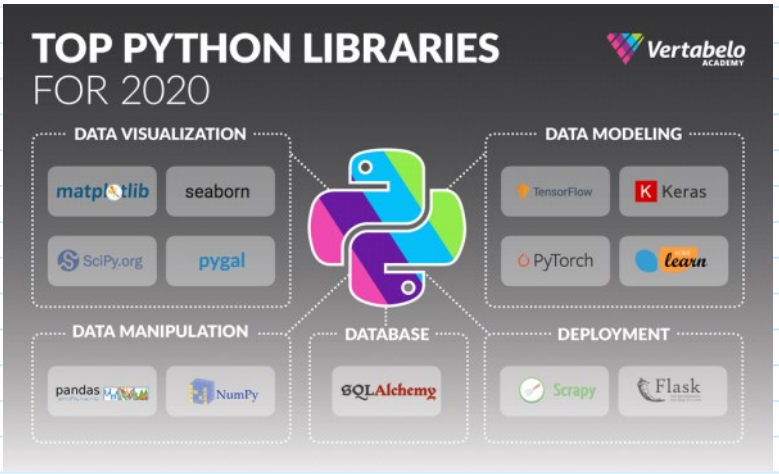
| Dictionary operations | |
|---|---|
| **dict = {}** | defines an empty dictionary |
| **dict[k] = x** | stores x associated to key k |
| **dict[k]** | retrieves the item with key k |
| **del dict[k]** | removes the item with key k |

| Dictionary methods | |
|---|---|
| **dict.keys()** | returns a list of keys |
| **dict.values()** | returns a list of values |
| **dict.items()** | returns a list of pairs (key,value) |
| **dict.get(k)** | returns the value associtated to the key k |
| **dict.pop()** | removes the item associated to the key and returns its value |
| **dict.update(D)** | adds keys-values (D) to dictionary |
| **dict.clear()** | removes all keys-values from the dictionary |
| **dict.copy()** | returns a copy of the dictionary |

- Library and packages:



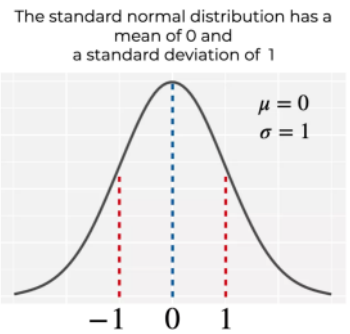

- Most important libraries:




- Tuples vs list
  - Tuples ( cannot be modified e.g., id and birth data) vs list (can be modified)

- Pickle: to save objects for later

- Generate random number from [ integer, distribution (e.g., normal and uniform]]
  - np.random.seed(2)
- Sample from [ list , vector , …. ]
  - df.sample(5, random_state=42)
- Generate Random variables from standard **normal distribution**
  - e.g., random.randn(6,4)

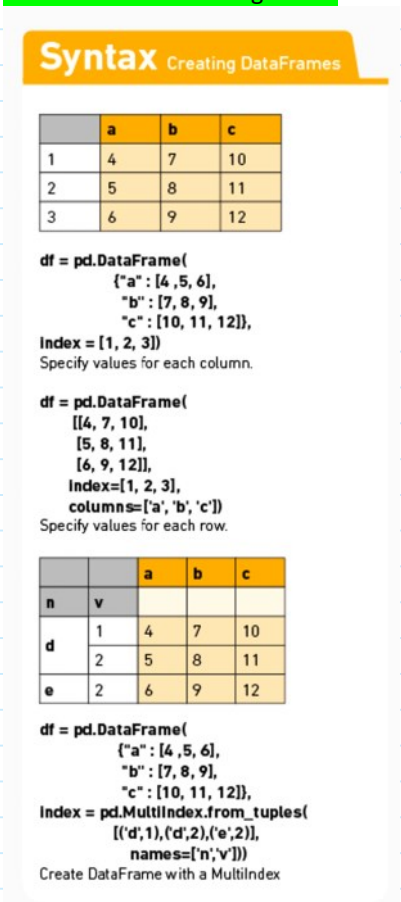



- How to write in Arabic
  - Install package: pip install --upgrade **arabic-reshaper**

        - conda install -c mpcabd arabic-reshaper
    - Import **arabic-reshaper**
    - Write following code:

```
plt.plot(data_list)
text = 'نجربي بالعربي'
text_r= arabic_reshaper.reshape(text)
rev = text_r[::-1]
plt.title(rev)
#plt.title(arabic_reshaper.reshape[1-::] (' نجرب بالعربي '))
```

- Create data frame using Panda



**Syntax** Creating DataFrames

| | a | b | c |
|---|---|---|---|
| 1 | 4 | 7 | 10 |
| 2 | 5 | 8 | 11 |
| 3 | 6 | 9 | 12 |

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
Index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    Index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```

Specify values for each row.

| | | a | b | c |
|---|---|---|---|---|
| n | v | | | |
| d | 1 | 4 | 7 | 10 |
| | 2 | 5 | 8 | 11 |
| e | 2 | 6 | 9 | 12 |

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
Index = pd.MultiIndex.from_tuples(
        [('d',1),('d',2),('e',2)],
         names=['n','v']))
```

Create DataFrame with a MultiIndex

- Summary statistics (Panda):

## Summarise Data

**df['w'].value_counts()**
Count number of rows with each unique value of variable

**len(df)**
# of rows in DataFrame.

**df['w'].nunique()**
# of distinct values in a column.

**df.describe()**
Basic descriptive statistics for each column (or GroupBy)

pandas provides a large set of **summary functions** that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

**sum()**
Sum values of each object.

**count()**
Count non-NA/null values of each object.

**median()**
Median value of each object.

**quantile([0.25,0.75])**
Quantiles of each object.

**apply(function)**
Apply function to each object

**min()**
Minimum value in each object.

**max()**
Maximum value in each object.

**mean()**
Mean value of each object.

**var()**
Variance of each object.

**std()**
Standard deviation of each object.

- Subset observations (Panda):

## Subset Observations (Rows)

**df[df.Length > 7]**
Extract rows that meet logical criteria.

**df.drop_duplicates()**
Remove duplicate rows (only considers columns).

**df.head(n)**
Select first n rows.

**df.tail(n)**
Select last n rows.

**df.sample(frac=0.5)**
Randomly select fraction of rows.

**df.sample(n=10)**
Randomly select n rows.

**df.iloc[10:20]**
Select rows by position.

**df.nlargest(n, 'value')**
Select and order top n entries.

**df.nsmallest(n, 'value')**
Select and order bottom n entries.

**Logic in Python (and pandas)**

| | | | |
|---|---|---|---|
| < | Less than | != | Not equal to |
| > | Greater than | df.column.isin(values) | Group membership |
| == | Equal to | pd.isnull(obj) | Is NaN |
| <= | Less than or equal to | pd.notnull(obj) | Is not NaN |
| >= | Greater than or equal to | &,\|,~,^,df.any(),df.all( | Logical and, or, not, xor, any, all |

## Subset Variables (Columns)

**df[['width','length','species']]**
Select multiple columns with specific names.

**df['width'] or df.width**
Select single column with specific name.

**df.filter(regex='regex')**
Select columns whose name matches regular expression regex.

**Logic in Python (and pandas)**

| | |
|---|---|
| '\.' | Matches strings containing a period '.' |
| 'Length$ | Matches strings ending with word 'Length' |
| '^Sepal' | Matches strings beginning with the word 'Sepal' |
| '^x[1-5]$' | Matches strings beginning with 'x' and ending with 1,2,3,4,5 |
| '^(?!Species$).*' | Matches strings except the string 'Species' |

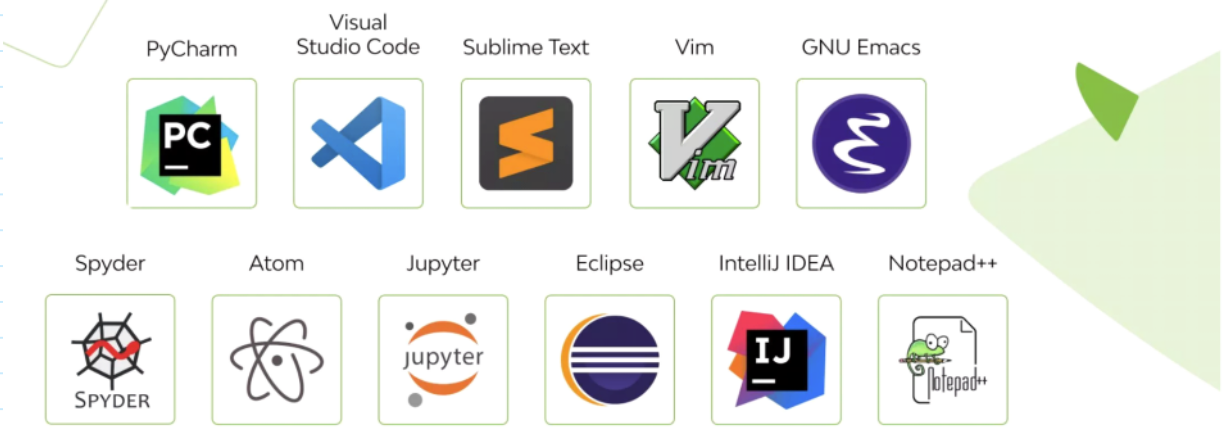
**df.loc[:,'x2':'x4']**
Select all columns between x2 and x4 (inclusive).

**df.iloc[:,[1,2,5]]**
Select columns in positions 1, 2 and 5 (first column is 0).

**df.loc[df['a'] > 10, ['a','c']]**
Select rows meeting logical condition, and only the specific columns .

- Best Python Integrated Development Environment (IDE)
  - Cloud-based Environments: Jopyter + Google colab
  - Local-based Environments: Spyder + PyCarm

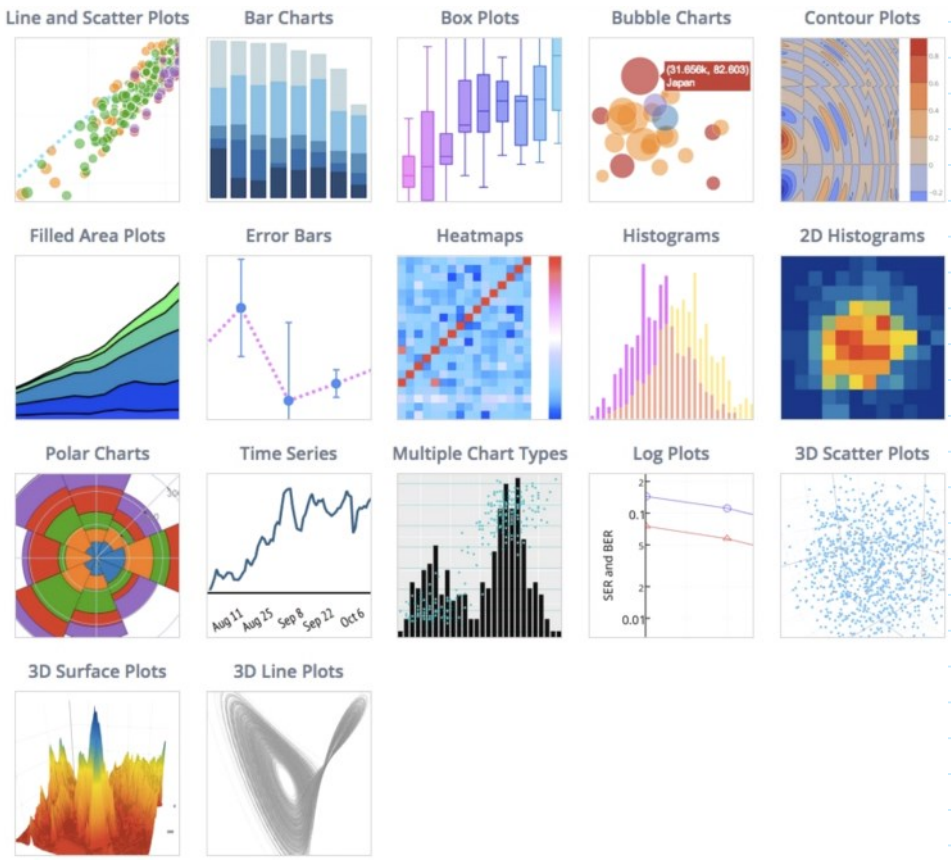
- Difference Examples of IDE:




- Cloud computing: on-demand availability of computer system resources [data storage + and computing power] without direct active management by the user.
  - Here is a list of my top 10 cloud service providers:
    - Amazon Web Services (AWS)
    - Microsoft Azure.
    - Google Cloud.
    - IBM Cloud.

- Plot types:


- Also there are others:




- Different software for data analysis (not-open source):
  - Power BI
  - Tableau
  - Orange
- Different software similar to Python (open source):
  - R