Advantages of Python

* Developer friendly
* Huge open source library

Data Types in Python

* Int
* float
* string
* tuple
* set

List

* Ordered mutable list of elements
* List has its own data type called ‘list’
* append() – appends the value to the last of the list
* pop() – removes the last element from the list
* Example : [1,2,3,4,5]
* Nested list - [ x\*\*2 for x in [x\*\*2 for x in range(11)]]

Dictionary

* Unordered list of elements
* Key value pairs
* E.g: d={‘key 1’: [1,2,3] , ‘key 2’ : {‘a’:100,’b’:200}}

Tuple

* Ordered immutable(can’t change/update) list of elements
* Count(), index()
* E.g: mytuple = ("apple", "banana", "cherry")

Set

* Unordered, unchangeable
* Does not allow duplicate values
* E.g: thisset = {"apple", "banana", "cherry"}

Comparison Operators : ==, <=,>=,!=

Logical Operators : AND, OR, NOT

Tuple Unpacking - We need identical pairs for tuple unpacking

E.g : new\_list = [(‘a’,100),(‘b’,300)]

For a,b in new\_list

Print(a), Print(b)

Useful commands while using for/while loop

* Pass : skips
* Continue : skips and continues
* Break : stops loop
* Enumerate-indexing data

\*Args

* \*Args – used to pass n number of arguments
* E.g:
* def myfunc(\*integers):

return sum(integers)

myfunc(1,2,3,4)

**Output** :10

\*Kwargs

* \*kwargs – used to pass n number of keyworded arguments
* E.g:

def marks(\*\*passandfail):

Return passandfail

marks(pass=’40’,fail=’30’)

**Output**: {‘pass’:’40’,’fail’:’30’}

Lambda

* Simplified function
* E.g :

x = lambda a : a + 10  
 print(x(5))

**Output** : 15

Map

* Apply same function to each element of a sequence and returns the modified list
* E.g: list(map(lambda x: x\*\*2,n)) – for each value of x lambda is performed

Filter

* Filters items out of a sequence
* E.g: list(filter(lambda x:x>2,n) – shows values only greater than 2

Reduce

* Applies same operations to items of a sequence
* Uses result of operation as param of next operation
* E.g : lis = [1, 3, 5, 6, 2, ]

# using reduce to compute sum of list

print("The sum of the list elements is : ", end="")

print(functools.reduce(lambda a, b: a+b, lis))

# using reduce to compute maximum element from list

print("The maximum element of the list is : ", end="")

print(functools.reduce(lambda a, b: a if a > b else b, lis))

NumPy

Useful commands in Numpy

|  |  |
| --- | --- |
| **Command** | **Use** |
| Np.array() | Creates an array |
| Np.arange() | Returns evenly spaced values within the specified interval |
| Np.ones() , Np.zeroes() | Generate arrays of ones & zeroes |
| Np.random.rand() | random samples from a uniform distribution over [0, 1) |
| Np.random.randn() | Return a sample (or samples) from the "standard normal" distribution. Unlike rand which is uniform |
| Np.append() | Appends data |
| Np.insert() | Inserts data |
| Np.delete() | Deletes data |
| Np.concatenate() | Concatenates multiple data into one |
| Np.split() | Splits one data into multiple |
| Np.add(),np.asubtract(),np.multiply(),np.divide() | Arithmetic Operations |
| Np.sqrt(),np.abs(),np.log() | Square root |
| Np.nan | Populates null value |
| Np.linspace() | Return evenly spaced numbers over a specified interval |
| Np.eye() | Creates an identity matrix |
| Np.max(),np.min() | Retrieves the maximum & minimum value of an array |
| Np.reshape() | Same data transforms to a new shape |

Pandas

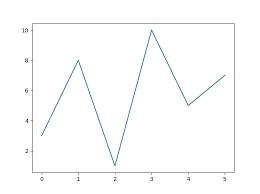
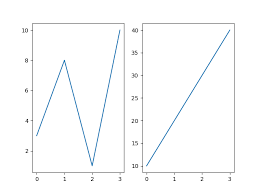
Useful commands in Pandas

|  |  |
| --- | --- |
| **Commands** | **Use** |
| Pd.Series() | Creates one-dimensional labeled array |
| Pd.dataframe() | Creates two-dimensional labeled data structure with columns of different data types |
| Df.loc() | To retrieve the |
| Df.iloc() | To retrieve the index location of a value |
| Df.drop() | To drop a column/row |
| Df.reset\_index() | To change the index value |
| Df.fillna() | To fill null values |
| Df.groupby() | To group columns |
| pd.concat() | To concatenate multiple column values |
| Pd.get\_dummies() | To populate dummy values |
| Pd.merge() | To merge values of different columns |
| Pd.read\_csv() | To read csv files |
| Df.shape() | Retrieves row and column count |
| Df.describe() | Summary of data statistics |

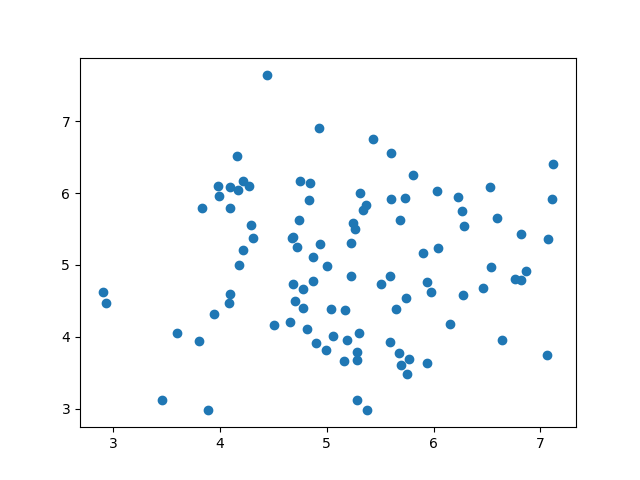
Visualization using Python Matplotlib, Seaborn

**Matplotlib.pyplot as plt**

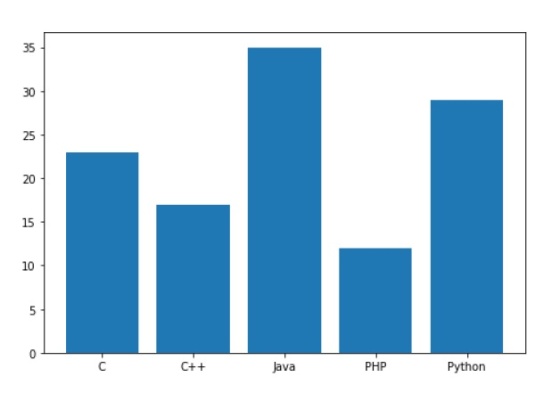
* plt.plot(), plt.subplot()

* plt.scatter(x, y)



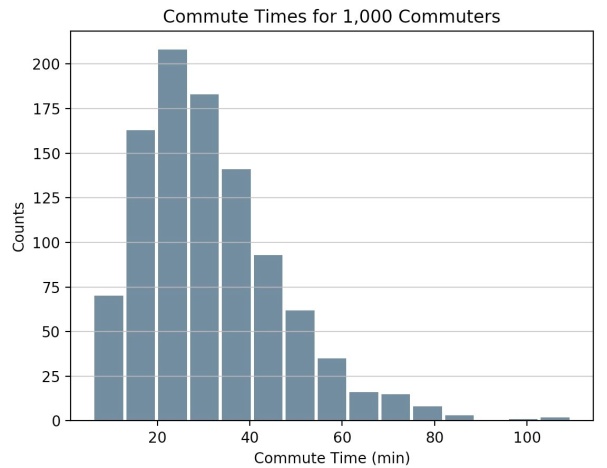
* plt.bar(x,y)



* plt.pie()

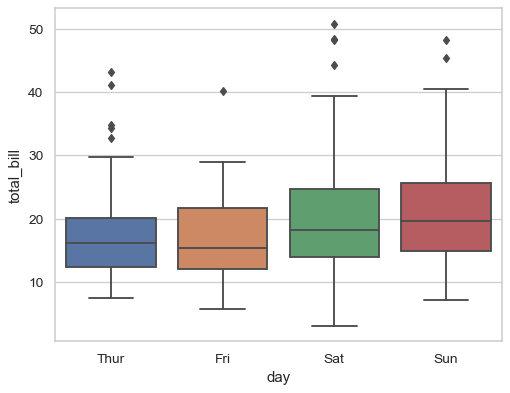


* plt.hist()

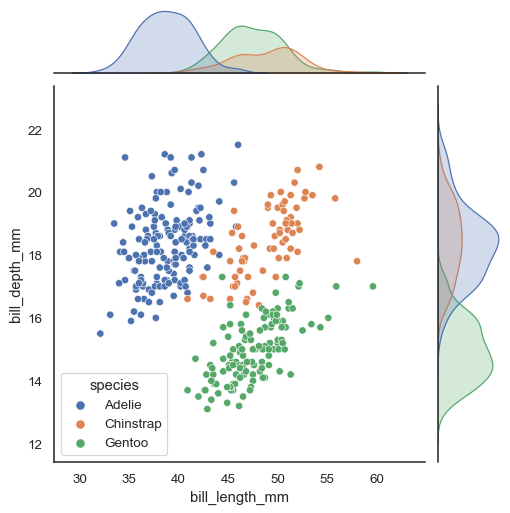


**Seaborn as sns**

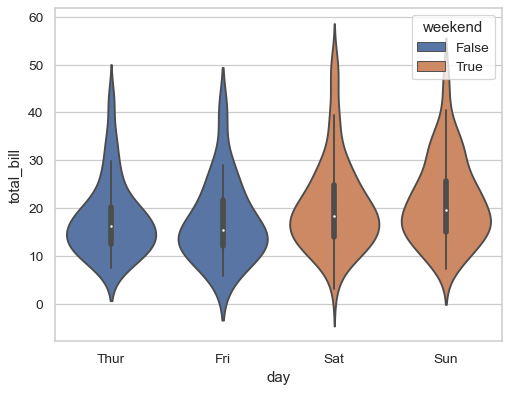
* Sns.boxplot()



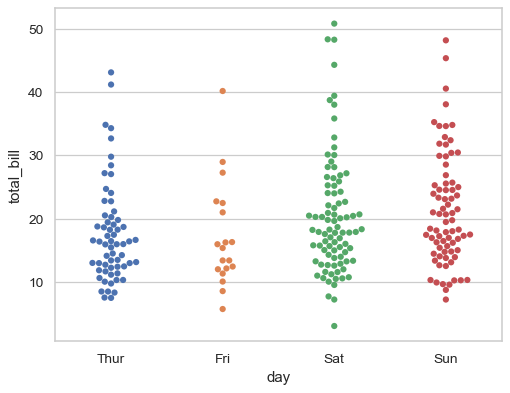
* Sns.jointplot()



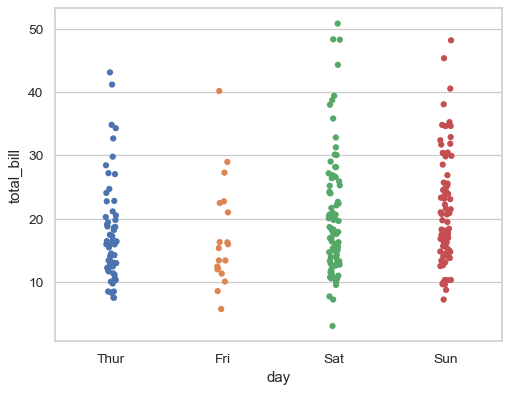
* Sns.violinplot()



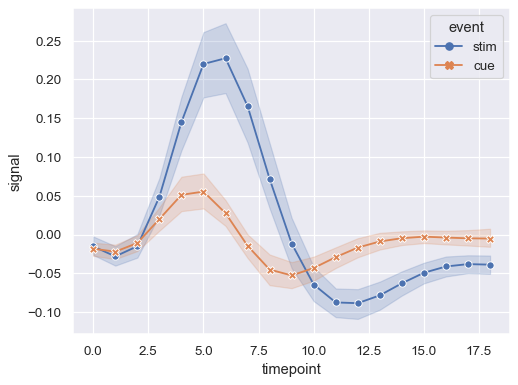
* Sns.swarmplot()



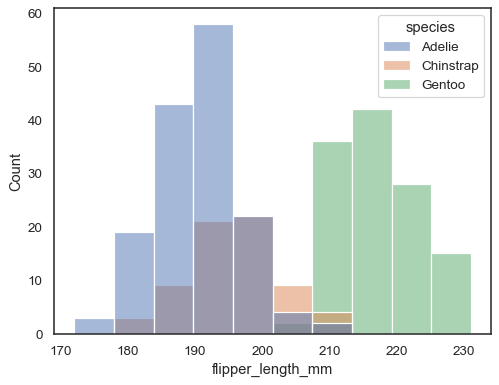
* Sns.striplot()



* Sns.lineplot()



* Sns.histplot()



* Sns.heatmap()

