Algorithms

A set of rules or steps used to solve a problem

Data Structures

A particular way to organized data in a computer

List Constants

List Constants are surrounded by square brackets and the element in the list are seprated by commas A list can be any python object even another list A list can be empty

In [1]:
```
print([1, 50, 100])
```

```
[1, 50, 100]
```

In [2]:
```
print(['Red', 'Yellow', 'Blue'])
```

```
['Red', 'Yellow', 'Blue']
```

In [3]:
```
print(['Red', 29, 98.6, 'watch'])
```

```
['Red', 29, 98.6, 'watch']
```

In [4]:
```
print([4.5, 'Python', 45.7])
```

```
[4.5, 'Python', 45.7]
```

In [11]:
```
print([3,5.5, 9.1, 'Pythonistas', 9.8 ])
```

```
[3, 5.5, 9.1, 'Pythonistas', 9.8]
```

In [12]:
```
print([2.3, 45, 'Amazing', [14, 10], 10])
```

```
[2.3, 45, 'Amazing', [14], 10]
```

In [13]:
```
print([1, [5, 6], 20])
```

```
[1, [5, 6], 20]
```

In [14]:
```
print(['Owais_Iqbal', [20 + 20], 12])
```

```
['Owais_Iqbal', [40], 12]
```

In [15]:
```
print([])
```

```
[]
```

In [16]:
```
print([])
```

```
[]
```

In [17]:
```
print([])
```

```
[]
```

In [18]:
```
print([])
```

```
[]
```

In [19]:
```
# we already use lists
for number in [3,5,6,7,80,90]:
    print(number)
```

```
3
5
6
7
80
90
```

In [20]:
```
for j in [45,78,90,12,34]:
    print(j)
```

```
45
78
90
12
34
```

In [11]:
```
for t in ['Awais', 'Umair', 'Joule']:
    print(t)
```

```
Awais
Umair
Joule
```

In [12]:
```
# Looking inside of a list
friends = ['Awais', 'Hamza', 'Junaid']
for friend in friends:
    print("Happy to see you again!", friend)
```

```
Happy to see you again! Awais
Happy to see you again! Hamza
Happy to see you again! Junaid
```

In [13]:
```
print(friends[1])
```

```
Hamza
```

In [14]:
```
print(friends[0])
```

```
Awais
```

In [15]:
```
print(friends[-1])
```

```
Junaid
```

String are immutable

String are immutable we can not the contents of a string we must make a new string to make any changes

List are mutable

list are mutable we can change the contents of and element of a list using index operator

In [16]:
```
fruit = "Banana"
fruit[0] = 'b'
print(b)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[16], line 2
      1 fruit = "Banana"
----> 2 fruit[0] = 'b'
      3 print(b)

TypeError: 'str' object does not support item assignment
```

In [17]:
```
x = fruit.lower()
print(x)
```

```
banana
```

In [18]:
```
lotto = [40, 12, 80, 50, 100]
print(lotto)
```

```
[40, 12, 80, 50, 100]
```

In [19]:
```
lotto[0] = 30
print(lotto)
```

```
[30, 12, 80, 50, 100]
```

In [20]:
```
lotto[-1] = 70
print(lotto)
```

```
[30, 12, 80, 50, 70]
```

In [21]:
```
lotto[3] = 20
print(lotto)
```

```
[30, 12, 80, 20, 70]
```

In [22]:
```
lotto[4] = 85
print(lotto)
```

```
[30, 12, 80, 20, 85]
```

Range Function

the range function returns a list of number

In [24]:
```
print(range(4))
```

```
range(0, 4)
```

In [25]:
```
print(range(8))
```

```
range(0, 8)
```

In [26]:
```
print(range(10))
```

```
range(0, 10)
```

In [27]:
```
print(range(3))
```

```
range(0, 3)
```

In [28]:
```
print(range(4))
```

```
range(0, 4)
```

In [29]:
```
print(range(100))
```

```
range(0, 100)
```

In [30]:
```
print(range(30))
```

```
range(0, 30)
```

In [31]:
```
print(range(12))
```

```
range(0, 12)
```

Building a list from scratch

In [12]:
```
stuff = list()
stuff.append('book')
stuff.append('owais')
stuff.append(99)
print(stuff)
```

```
['book', 'owais', 99]
```

In [13]:
```
Books = list()
Books.append("python")
Books.append("Excel")
Books.append("SQL")
print(Books)
```

```
['python', 'Excel', 'SQL']
```

In [4]:
```
names = list()
names.append("Owais Iqbal")
names.append("Umair Iqbal")
names.append("Maqbool Ahmed")
names.append("Hamza Jaffar")
names.append("Junaid Saeed")
names.append("Wasiq Jatt")
print(names)
```

```
['Owais Iqbal', 'Umair Iqbal', 'Maqbool Ahmed', 'Hamza Jaffar', 'Junaid Saeed', 'Wasiq Jatt']
```

In [5]:
```
programming_languages = list()
programming_languages.append("python")
programming_languages.append("SQL")
programming_languages.append("Java")
programming_languages.append("Ruby")
programming_languages.append("c++")
programming_languages.append("Cobol")
print(programming_languages)
```

```
['python', 'SQL', 'Java', 'Ruby', 'c++', 'Cobol']
```

In [6]:
```
# with the help of the append method we can create a list from scratch
```

In [7]:
```
task = list()
task.append("Create a file")
task.append("learning a new tech")
print(task)
```

```
['Create a file', 'learning a new tech']
```

Lists are in order

a list can be sorted it's means to changes it's order

In [9]:
```
friends = ["owais", "Hamza", "junaid", "jaffar"]
friends.sort()
print(friends)
```

```
['Hamza', 'jaffar', 'junaid', 'owais']
```

In [10]:
```
friends[1]
```

Out[10]:
```
'jaffar'
```

In [11]:
```
friends[0]
```

Out[11]:
```
'Hamza'
```

In [14]:
```
friends[0] = "Owais"
print(friends)
```

```
['Owais', 'jaffar', 'junaid', 'owais']
```

Split

Split breaks a string into the parts and produce a list of string

In [17]:
```
sentence = "I am learning list data structre in python:"
var = sentence.split()
print(var)
```

```
['I', 'am', 'learning', 'list', 'data', 'structre', 'in', 'python:']
```

In [19]:
```
fav = "Python is my favorite language"
x = fav.split()
print(x)
```

```
['Python', 'is', 'my', 'favorite', 'language']
```

In [20]:
```
file = open('mbox.txt')
for line in file:
    line = line.rstrip()
    if not line.startswith('From:'):
        continue
        words = line.split()
        print(words[2])
```

In [21]:
```
file = open('mbox.txt')
for line in file:
    line = line.rstrip()
    if not line.startswith('of'):
        continue
        words = line.split()
        print(words[4])
```

In [22]:
```
file = open('mbox.txt')
for line in file:
    line = line.rstrip()
    if not line.startswith('C550x'):
        continue
        words = line.split()
        print(words[2])
```

List

A linear collection of values that stay in order

Dictionaries

A bag of values each with it's own label

```
In [ ]:
```