Inedxing

In [2]:
```
food = "Apple"
print (food)
```

```
Apple
```

In [3]:
```
#find lenght of the string
len(food)
```

Out[3]:
```
5
```

In [5]:
```
#indexing: It always starts from 0
food[0]
```

Out[5]:
```
'A'
```

In [6]:
```
#range in indexing
food[0:3] #Always Remember: here 3 is not included, It will run 0,1,2 only
```

Out[6]:
```
'App'
```

In [7]:
```
#range in indexing
food[-2:5] #-2 WILL SKIP First 2 Chracters
```

Out[7]:
```
'le'
```

Basic Data Structures in Python

1. Tuples
2. List
3. Dictionaries
4. Set

Tupples

A tuple is a collection which is: Ordered and unchangeable. Tuples are written with round brackets"()".

In [10]:
```
tuple = ("Car",5000000,"Civic",2002)
print(tuple) #will print all values
print(len(tuple)) #Will print just length
print(type(tuple))

#The answer is tuple due to the (), not because of the var name tuple.
```

```
('Car', 5000000, 'Civic', 2002)
4
<class 'tuple'>
```

Indexing in TUPLE

In [11]:
```
tuple[1] #It will give the complete value at index 1 which is 5000000
```

Out[11]:
```
5000000
```

In [12]:
```
#Range
tuple[1:3]
```

Out[12]:
```
(5000000, 'Civic')
```

In [14]:
```
tup2 =("van",20000)
tup3=tuple+tup2
print(tup3)
```

```
('Car', 5000000, 'Civic', 2002, 'van', 20000)
```

In [15]:
```
tup3+tup2
```

Out[15]:
```
('Car', 5000000, 'Civic', 2002, 'van', 20000, 'van', 20000)
```

In [16]:
```
tuple+tup2
```

Out[16]:
```
('Car', 5000000, 'Civic', 2002, 'van', 20000)
```

In [19]:
```
tup4=(2,3,4,5,6,7,75,2,2,2)
min(tup4)
```

Out[19]:
```
2
```

In [20]:
```
tup4*2
#it does not multiple every value with 2,
```

Out[20]:
```
(2, 3, 4, 5, 6, 7, 75, 2, 2, 2, 2, 3, 4, 5, 6, 7, 75, 2, 2, 2)
```

List

Lists are created using square brackets:

In [24]:
```
list1 = ["Car",5000000,"Civic",2002]
print(list1)
print(len(list1)) #Will print just length
print(type(list1))
```

```
['Car', 5000000, 'Civic', 2002]
4
<class 'list'>
```

In [26]:
```
list2 = [2,23,45,"Python"]
print(list2)
```

```
[2, 23, 45, 'Python']
```

In [27]:
```
list1+list2
```

Out[27]:
```
['Car', 5000000, 'Civic', 2002, 2, 23, 45, 'Python']
```

In [29]:
```
list1[0:3]
```

Out[29]:
```
['Car', 5000000, 'Civic']
```

In [30]:
```
list1[1]
```

Out[30]:
```
5000000
```

In [34]:
```
list1.reverse()
list1
```

Out[34]:
```
[2002, 'Civic', 5000000, 'Car']
```

In [43]:
```
list1.append("Code")
list1
```

Out[43]:
```
[2002, 'Civic', 5000000, 'Car', 'Daniyal', 'Daniyal', 'Daniyal', 'Code']
```

Dictionary

Dictionaries are written with curly brackets, and have keys and values:

In [47]:
```
cars = {
  "brand": "Ford",
  "model": "xyz",
  "year": 2022,

}
print(cars)
```

```
{'brand': 'Ford', 'model': 'xyz', 'year': 2022}
```

In [46]:
```
cars1 = {
  "brand": "Honda",
  "model": "yxz",
  "year": 2020
}
print(cars1)
```

```
{'brand': 'Honda', 'model': 'yxz', 'year': 2020}
```

In [48]:
```
print(type(cars1))
```

```
<class 'dict'>
```

In [50]:
```
cars.update(cars1)
cars
```

Out[50]:
```
{'brand': 'Honda', 'model': 'yxz', 'year': 2020}
```

SET

1. set uses curly "{}" brackets but donot have key and value
2. It is un odered and unchangeable

In [52]:
```
set = {"apple", "banana", "cherry"}
print(set)
```

```
{'apple', 'cherry', 'banana'}
```

In [57]:
```
set1 = ("apple1", "banana1", "cherry1")

set1
```

Out[57]:
```
('apple1', 'banana1', 'cherry1')
```