Date: 17 / 12 / 2025

Work Done Today

- Loop
- Data Structures

Description

- **LOOPs**
    - **while loop**

      Syntax :
      ```
      while condition:
              Block of code
      ```

      Example :
      ```
      i = 1
      while i <= 10:
              print(i)
      ```

      → **break** keyword used for terminate the loop when you want , even condition is true

      Example:
      ```
      i = 1
      while i <= 10:
              print(i)
              if(i==6):
                      break
      ```

      → **continue** keyword used to skip specific iteration and then again restart the loop

      Example:
      ```
      i = 1
      while i <= 10:
              print(i)
              if(i==6):
                      continue
      ```

      → We can also use **else** with while loop

      Example :
      ```
      i = 1
      while i <= 10:
              print(i)
      else:
      ```

```
        print(“loop is ended after 10 ”)
```

- **for loop**

  Syntax:
  ```
  for variable_name in range:
          Block of code
  ```

  Example :
  ```
  fruit = [“apple”,”banana”,”fig”]
  for x in fruit :
          print(x)
  ```

  - Looping through a string
    Example :
    ```
    for x in “banana”:
            print(x)
    ```

  - Using **break** keyword
    Example:
    ```
    for x in “banana”:
            print(x)
            if(x==”n”):
                    break
    ```
  - Using **continue** keyword
    Example:
    ```
    Alpha = [‘A’,’B’,’C’,’D’]
    for x in Alpha :
            print(x)
            if(x == ‘C’):
                    continue
    ```
  - range() function is used for sequence of numbers start from 0 (by default) and increment by 1 (by default)

    Example :
    ```
    for num in range(10):
            print(num)
    ```

    - **range(start , limit)**
      Example:
      ```
      for num in range(2,10):
              print(num)
      o/p: 2,...,9
      ```

    - **range(start,limit,iteration)**

```
Example:
for i in range(2,13,2):
        print(i)
o/p: 2,4,6,8,...,12
```

- **Using else in for loop**

```
Example :
for a in “Sayma”:
        print(a)
else:
        print(“loop is finished …”)
```

- **Nested for loop**

```
Example:
for a in “sayma”:
        For b in “kazi”:
                print(a,b)
```

- **Using pass keyword (used for future changes or code in loop)**

```
Example :
for i in “banana”:
        Pass
```

- **Data Structures**
  - **List**
    - Lists are used to store multiple values in a single variable.
    - List is changeable.
    - List is ordered.
    - List allows duplicate values.
    - List is created using square brackets [ ].
    - List can store different values with different datatypes.

```
Example:
        List1 = [“sayma”,”arish”,”saniya”,”mahek”]
        print(List1[1]) # sayma
        print(List1[-1]) # mahek

        List2 = [56 , 24.90, “sayma”,”sayma”]
        print(List2[1:3]) # 24.90,sayma
        print(List2[:3]) # 56, 24.90,sayma
        print(List2[1:]) # 24.90,sayma,sayma
        print(List2[-3:-1]) # sayma,sayma
```

- **Changing value of item**

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]
thislist[1] = blackberry
print(thislist) # apple,blackberry,cherry,orange,kiwi,mango

➔ **Changing multiple values of multiple items**
thislist[1:3] = [“sayma”,”happy”]
print(thislist) #"apple", "sayma", "happy ", "orange", "kiwi", "mango"

➔ **Change second value with three items**
thislist[1:2] = [“joy”,”angry”,”tired”]
print(thislist) #"apple", "joy",”angry”,”tired”, "happy ", "orange", "kiwi", "mango"

➔ **len() function - return the length**
Example:
List1 = [“sayma”,”arish”,”saniya”,”mahek”]
print(len(List1))

➔ **type() function - return type of object**
Example:
List1 = [“sayma”,”arish”,”saniya”,”mahek”]
print(type(List1))

➔ **list() constructor**
Example :
Mylist = list((“apple”,”cherry”))
print(mylist)

➔ **insert() function**
Example:
Mylist = [56,25,90,72,69]
Mylist.insert(2,400)
print(Mylist)# 56,25,400,72,69

➔ **append() function -** used to insert item at end of list
Example:
Mylist = [56,25,90,72,69]
Mylist.append(88)#56,25,90,72,69,88

➔ **extend() function -** append list or any iterable or list at the end of list
Example:

Mylist1 = [56,25,90,72,69]
Mylist2 = [‘a’,’b’,’c’]
Mylist1.extend(Mylist2)#56,25,90,72,69,‘a’,’b’,’c’

➔ **remove() function -** remove specific item or first occurrence of specific item
Example:
Mylist = [56,25,90,72,69,90]
Mylist.remove(90)#56,25,72,69,90

➔ **pop() function -** remove item at specific index
Example:
Mylist = [56,25,72,69,90]
Mylist.pop(3)#56,25,72,90

➔ **del keyword -** clear item at specific index
Example:
Mylist = [56,25,72,69,90]
del Mylist[1] # 56,72,69,90

➔ **clear() function -** clear the all values of list(empty list)
Example:
Mylist = [56,25,72,69,90]
Mylist.clear()

➔ **sort() function -** sorting list in ascending order(by default)
Example:
Mylist = [56,25,72,69,90]
Mylist.sort()
print(Mylist)# 25,56,69,72,90

➔ **sort(reverse = True) -** sort list in descending order
Example:
Mylist = [56,25,72,69,90]
Mylist.sort(reverse = True)
print(Mylist) # 90,72,69,56,25

➔ **reverse() -** give reverse order list
Example:
Mylist = [25,56,69,72,90]
Mylist.reverse()
print(Mylist) # 90,72,69,56,25

➔ **copy() -** copying list to another list

```
Mylist = [ 25,56,69,72,90]
Thislist = Mylist.copy()
print(thislist) #25,56,69,72,90
```

- **list() -** copying list

```
Mylist = [ 25,56,69,72,90]
Thislist = list(Mylist)
print(thislist) #25,56,69,72,90
```

- **Slice operator for copying list**

```
Mylist = [ 25,56,69,72,90]
Thislist = Mylist[:]
print(thislist) #25,56,69,72,90
```

❖ **Tuple**

- Tuple is used for storing multiple values in a single variable.
- Tuple is ordered.
- Tuple is unchangeable.
- Tuple allows duplicate values
- Tuple is created using round brackets ( ).
- Tuple items are separated by commas.
- If you want to insert one value in tuple then it should be written like this tuple1 = (“sayma” ,) . you must separate it by comma .
- Use len() function for identifying its length.
- Use tuple() constructor to make tuple.

```
Example:
mytuple(“rose”,”lily”,”sunflower”)
print(mytuple[1])
print(mytuple)
```

❖ **Set**

- The set is unordered.
- The set is unchangeable.
- The set is unindexed
- The set does not allow duplicate values.
- Once a set is created , it cannot be changed but you can remove items or add items.
- In set True and 1 is considered the same .
- In set False and 0 is considered the same .
- type() function is used to identify data type of variable.
- len() function is used to measure the length of a set .
- set() constructor is used to create sets from iterable .
- The set is created using curly brackets { }.

Example:
Myset = {“laptop”,”computer”,”printer”}

Accessing set :

```
Myset = {“laptop”,”computer”,”printer”}
for x in Myset:
        print(x)
```

- ➔ **add()** - add the item
  Example:
  ```
  Myset = {“laptop”,”computer”,”printer”}
  Myset.add(“mouse”) #“laptop”,”computer”,”printer”,“mouse”
  ```

- ➔ **update() -** add another set or iterable to set
  example:
  ```
  Myset = {“laptop”,”computer”,”printer”}
  tup = {“no”,”yes”}
  Myset.update(tup)#“laptop”,”computer”,”printer”,“no”,”yes”
  ```

- ➔ **remove() -** remove specific item(if item not present, it raise error)
  Example :
  ```
  Myset = {“laptop”,”computer”,”printer”}
  Myset.remove(“laptop”) #”computer”,”printer”
  ```

- ➔ **discard() -** remove specific item (if item is not present , doesn’t raise error)
  Example :
  ```
  Myset = {“laptop”,”computer”,”printer”}
  Myset.discard(“laptop”) # ”computer”,”printer”
  ```

- ➔ **del -** delete complete set
  ```
  Myset = {“laptop”,”computer”,”printer”}
  del Myset
  print(Myset) #give error
  ```

- ➔ **pop() -** remove random item of set (because it is unordered)
  ```
  Myset = {“laptop”,”computer”,”printer”}
  Myset.pop() #“laptop”,”computer”
  ```

- ➔ **Union (|)-** returns all items of different set into a another set
  Example:
  ```
  Myset = {“laptop”,”computer”,”printer”}
  ```

tup = {“no”,”yes”}
Set3 = Myset.union(tup)
print(Set3) #“laptop”,”computer”,”printer”,“no”,”yes”

Or you can write like this

Myset = {“laptop”,”computer”,”printer”}
tup = {“no”,”yes”}
Set3 = Myset | tup
print(Set3)

➔ **Intersection (&) -** return similar items in both sets
Example:
Myset = {“laptop”,”computer”,”printer”}
tup = {“no”,”yes”,”laptop”}
Set3 = Myset.union(tup)
print(Set3) # ”laptop”

Or you can write like this

Myset = {“laptop”,”computer”,”printer”}
tup = {“no”,”yes”,”laptop”}
Set3 = Myset & tup
print(Set3)

➔ **intersection_update() -** it is used to return similar item in both sets but does not require third set , it will change existing set
Example:
Myset = {“laptop”,”computer”,”printer”}
tup = {“no”,”yes”,”laptop”}
Myset.intersection_update(tup)
print(Myset) #”laptop”

➔ **difference() (-) -** return new set that contain only items of first set that are not present in other set
Example:
Myset = {“laptop”,”computer”,”printer”}
tup = {“no”,”yes”,”laptop”}
Set3 = Myset.difference(tup)
print(Set3) #computer , printer

Or you can write like this

Myset = {“laptop”,”computer”,”printer”}

```
tup = {“no”,”yes”,”laptop”}
Set3 = Myset - tup
print(Set3)
```

➔ **symmetric_difference() (^)** - keep only elements are not same in both sets

Example:
```
Myset = {“laptop”,”computer”,”printer”}
tup = {“no”,”yes”,”laptop”}
Set3 = Myset.symmetric_difference(tup)
print(Set3) # ”computer”,”printer”,“no”,”yes”
```

Or you can write like this

```
Myset = {“laptop”,”computer”,”printer”}
tup = {“no”,”yes”,”laptop”}
Set3 = Myset ^ tup
print(Set3)
```

❖ **Dictionary**

➔ The dictionary is ordered.
➔ It is changeable.
➔ It does not allow duplicates .
➔ Store in key:value pair format.
➔ Created by using curly brackets { }.
➔ Dictionary items can be any data type.

Example:
```
Dict = {
“Type” : “flower”,
“Name” : “rose”,
“Color” : [“black”,”red”,”pink”,”white”]
}
```

Accessing items:

1. Using key name
   Example : x = Dict[“Type”]

2. Using get()
   Example: x = Dict.get(“Type”)

3. Accessing all key names
   Example: x = Dict.keys() #return all key names

4. Accessing all values
   Example: x = Dict.values() #return all values of key

5. Accessing all items
   Example: x = Dict.items() #return all items of dictionary

➔ **Changing values of key**
  Example:
```
dict = {
"type" : "flower",
"name" : "rose",
"color" : ["black","red","pink","white"]
}
dict["name"] = "lily"
```

➔ **Changing values using update()**
  Example:
```
dict = {
"type" : "flower",
"name" : "rose",
"color" : ["black","red","pink","white"]
}
dict.update({"name" : "lily"})
```

➔ **Adding items**
  Example:
```
car = {
"name" : "BMW",
"model" : "S5",
"color" : "black"
}
car["color_type"] = "mate"
```

➔ **pop() -** removing item
  Example:
```
car = {
"name" : "BMW",
"model" : "S5",
"color" : "black"
}
car.pop("name")
```

➔ **popitem() -** remove last inserted item
  Example:

```
car = {
“name” : “BMW”,
“model” : “S5”,
“color” : “black”
}
car.popitem()
```

- **del -** remove all dictionary
  Example:
  ```
  car = {
  “name” : “BMW”,
  “model” : “S5”,
  “color” : “black”
  }
  del car
  ```

- **clear() -** delete all items in dictionary
  Example:
  ```
  car = {
  “name” : “BMW”,
  “model” : “S5”,
  “color” : “black”
  }
  car.clear()
  ```

- **copy() -** copying dictionary
  Example:
  ```
  car = {
  “name” : “BMW”,
  “model” : “S5”,
  “color” : “black”
  }
  C = car.copy()
  ```

- **dict() -** copy using dict()
  ```
  car = {
  “name” : “BMW”,
  “model” : “S5”,
  “color” : “black”
  }
  C = dict(car)
  ```