Day 9
15/8/22

Dictionary

→ In Python, a dictionary can be created by placing a sequence of elements within curly {} braces, separted by 'comma'.

→ Dictionary Keys are case sensitive, the same name but different cases of key will be treated distinctly.

→ Dictionary is a collection of items where each item is a key: value pair

→ Values can be repeated but the keys must be unique.

syntax:-

```
Dict1 = {"key": "value", "key2": "value2"}
```

→ Dictionary methods :-

1) clear() - Remove all the elements from the dictionary.

2) copy() - Returns a copy of the dictionary

3) get() - Returns the value of specified key.

4) items() - Returns a list containing a tuple for each key value pair.

5) keys() - Returns a list containing dictionary's keys.

6) pop() - Remove the element with specified key

7) popitem() - Removes the last inserted key-value pair

8) update() - updates dictionary with specified key-value pairs.

9) values() - Returns a list of all the values of dictionary.

Examples:-

```
dict1 = { 1: "Java",
          2: "c++",
          3: "c",
          4: "Javascript"
        }
```

```
dict2={ "name":"yogu
  "years": 3,
  "location": pune
}
```

ex1) # copy() method.

```
dict2 = dict1.copy()
print(dict2)
```

ex2) # clear() method

```
dict1.clear()
print(dict1)
```

ex3) # get() method

```
print (dict2.get(1))
```

ex.4) # items() method

```
print (dict2.items())
```

ex.5) Keys() method

```
print (dict2.keys())
```

ex.6) pop() method

```
dict2.pop(4)
print(dict2)
```

ex.6) popitem() method

```
dict2.popitem()
print(dict2)
```

ex.6) update() method

```
dict2.update({3:"Javascript"})
print(dict2)
```

ex.7) values() method

```
print(dict2.values())
```

Python conditions and IF Statements.

→ Phyton supports the logical conditions :- i.e
→ logical operators i.e. and, or

Equals: a == b
Not equals: a != b
less than: a < b
less than or equal to: a <= b
Greater than: a > b
Greater than or equal to: a >= b

⇒ **If statement** -

It is used to decide whether a certain statement or block of statements will be executed or not.

ex:-
```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

⇒ **Elif** -

ex:-
```
a = 33
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a & b are equal")
```

⇒ **if-elif-else ladder** -

If statements are executed from top down. And the rest of the ladder is by passed. If none of conditions is true, then the final else statement will be executed.

ex:-
```
i = 20
if (i == 10):
    print ("i is 10")
elif (i == 15):
    print ("i is 15")
elif (i == 20):
    print ("i is 20")
else:
    print ("i is not present")
```

⇒ **Short Hand if statement** -

Whenever there is only a single statement to be executed inside the if block then shorthand if can be used.

ex:-
```
i = 10
if i < 15: print ("i is less than 15")
```

⇒ **Short Hand if-else statement** -

This can be used to write the if-else statements in a single line where there is only one statement to be executed in both if and else block.

ex:-
```
i = 10
print (True) if i < 15 else print (False)
```

Logical operators

⇒ Logical operators are used on conditional statements (either True or False)

1) Logical AND:

True if both the operands are true.

```
ex:- a = 200
     b = 100
     c = 300
     if a > b and c > a:
         print ("Both conditions are True")
```

2) Logical OR: True if either of the operands is True

```
ex:- a = 90
     b = 70
     c = 100
     if (a > b or a > c:
     print ("At least one of the conditions is True")
```

3) Logical NOT: True if operand is false.

```
ex:-
     if not a:
         print ("Value of a is true")
     if not (a % 6 == 0 or a % 9 == 0):
         print ("90 is not divisible by either 6
                                    or 9")
     else:
         print ("90 is divisible by either6
                                    or 9")
```