MODULE - 2

Solved Question & answers of python programming.

1] What is Lists ? Explain the concept of list slicing with example. [-6M-] [July/Aug 2022]

Ans: Refer Notes Page 1 to 3

2] What is Dictionary ? How it is different from Lists ? Write a program to count the number of occurences of character in a string. [-7M-] [July/Aug 2022] V.V.V important

Ans [Chapter - 5]

Refer Notes Page 25 - 26. for theory

```
string = input ("Enter a string : ")
char = input ("Enter the character to count : ")
count = string.count(char)
print (f"The character '{char}' appears {count} times in the string")
```

Input :-

Enter a string : hello world

Enter the character to count : l

Output :-

The character 'l' appears 3 times in the string.

3] Write a python program that accepts a sentence & find the no of words, digits, uppercase letters & lower case letters. [-7M-] July/Aug 2022.

```
# Input sentence
sentence = input ("Enter a sentence : ")

# Initialize counters
words = len (sentence. split())
digits = sum ( c.isdigit() for c in sentence)
uppercase = sum ( c.isupper() for c in sentence)
lowercase = sum ( c.islower() for c in sentence)

# Display results
print("Words :", words)
print ("Digits :", digits)
print ("Uppercase letters :", uppercase)
print ("Lowercase letters :", lowercase)
```

4] List out all the useful string methods which supports in python Explain with an example for each method. [-10M-]
July/Aug 2022.

Ans Pag no ~~got~~ 91 [Sequence data types]
21 to 24

5] What is the diff^ce b/w copy.copy() and copy.deepcopy(). Function applicable to a list or dictionary in python? Give suitable examples for each [-6M-] July/Aug 2022 (vvv.important)

Ans The key diff^ce b/w copy.copy() & copy.deepcopy() in python lies in how they handle nested structures (like lists/dictionaries within

lists or dictionaries.

copy.copy()

↳ Creates a shallow copy of the object.

↳ The outer object is copied but references to inner objects are preserved. Modifying the inner objelt in the copy affects the original.

Example (Shallow copy) or copy.copy

```
import copy
original = [[1,2,3],[4,5,6]]
shallow_copy = copy.copy(original))
shallow_copy[0][0] = 99    # modify inner list
print("Original", original)  # [[99,2,3],[4,5,6]]
print("Shallow Copy:"        # [[99,2,3],[4,5,6]]
      shallow_copy)
```

copy.deepcopy()

↳ Creates a deep copy of the object

↳ Both the outer & inner objects are fully duplicated. Changes in the copy don't affect the original

Example (Deep copy)

```
import copy
original = [[1,2,3],[4,5,6]]
```

```
deep_copy = copy.deepcopy(original)

deep_copy[0][0] = 99      # Modify inner list
print("Original:", original) # [[1,2,3],[4,5,6]]
print("Deep Copy:", deep_copy) # [[99,2,3],[4,5,6]]
```

Key Difference :-

↳ Shallow Copy : Changes to inner elements in the copy affect the original

↳ Deep Copy : Changes to inner elements in the copy don't affect the original.

↳ Use copy.copy() for a shallow copy [useful for non-nested objects]

↳ Use copy.deepcopy() for a deep copy [to copy nested structures completely].

6] Write a python Program to swap cases of a given string

Input : Java
Output : JAVA

[-4M-] July/Aug 2022

Ans

```
# Input string
input_string = "Java"

# Swap cases using the swapcase() method
output_string = input_string.swapcase()

# Print the output
print("Input", input_string)
```

```
print ("Output ", output_string)
```

OUTPUT :-

Input : Java
Output : JAVA

The swapcase() method automatically converts uppercase letters to lowercase & vice versa.

3] Explain with a programming example to each :
(ii) get()
(iii) setdefault() [6M-] [2022-23]

(ii) * The get() method in python is used to access the value of a key in a dictionary. It takes one mandatory argument, which is the key to be searched in the dictionary.

```
# create a dictionary
my_dict = {'apple': 10, 'banana': 20, 'orange': 30}

# access the value of a key using get()
apple_count = my_dict.get('apple')

# print the values
print (apple_count)
```

Output : 10//

(iii) setdefault() :- The setdefault() method in python is used to insert a key value pair into a dictionary if the key does not exist. It takes two arguments, the 1st argument is

the key to be inserted. & the second argument (optional) is the value to be assigned to the key.

```
# create a dictionary
my_dict = {'apple': 10, 'banana': 20, 'orange': 30}

# add a new key-value pair using setdefault()
mango_count = my_dict.setdefault('mango', 40)

# update the value of an existing key using setdefault()
apple_count = my_dict.setdefault('apple', 50)

# print the dictionary
print(my_dict)
```

Output :-

```
{'apple': 10, 'banana': 20, 'orange': 30, 'mango': 40}
```

8] Explain append() and index() function w.r.t. to list in Python [-6M-] [2022-23]

Ans. For index(), refer page no 17 //

For append(), refer pag no 18 //

9] Explain the dif ways to delete an element from a list with suitable Python syntax & Programming examples.

Ans refer Pg no 10

pop() → This method deletes the last element in the list, by default.

```
ls = [3,6,-2,8,10]
x = ls.pop()
print(ls)
```

Output :-

[3, 6, -2, 8]

The del statement will delete values at an index in all list. All of the values in the list after the deleted value will be moved up one index.

```
ls = [3, 6, -2, 8, 10]
del ls = [2]
print(ls)
```

Output :-

[3,6,8,10]

clear() :- This method removes all the elements in the list & makes the list empty.

```
ls = [1,2,3]
ls.clear()
Print(ls)
```

↳ You can use the remove() method to remove an element from a list by specifying its value. Here's an example

```
my_list = [1,2,3,4,5]
my_list.remove(3)
print(my_list)
```

Output :-

[1,2,4,5]

10. Read a multi-digit number (as chars) from the console. Develop a program to print the frequency of each digit with suitable message.

Ans

```
def countoccurences(n,d):
    count = 0
# Loop to find the digits of N
    while(n>0)
# check if the digit is D
    if(n%.10 == d)
      count = count + 1
      n= n/10
# return the count of the ocaurances of D in N
    return count
# Driver code
    d=2
    n= int(input("Enter the multi digit number"))
    print(countoccurences(n,d))
```

Output :-

Enter the multi digit number : 223232

4//

11. Tuples are Immutable. Explain with python programming example. [-4M-] [2022-2023]

Ans Refer pg no 24 to 25//

12. Explain the use of in and not in operators, in list with suitable examples [6M-] Jan/Feb 2023

Ans Refer pg no 13 to 14.

13. Develop a program to find mean, variance & standard deviation [6M-] Jan/Feb 2023

```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
sd = []
mean = sum(a) / len(a)
print("Mean", mean) => mean 5.0
for x in a:
    sd.append((x-mean)**2)

V = sum(sd) / len(sd)
print("Variance", V) => Varience : 6.666667
print("Standard Deviation", V**0.5) => Standard deviation: 2.581988
```

(mean 5.0 — answer; Varience : 6.666667 — answer; Standard deviation: 2.581988 — answer)

4. Explain the following methods in lists with an example
i) len() ii) sum() iii) max() iv) min() Jan/Feb 2023

Refer pg no 7 for len()

sum() : It returns the sum of all the elements (no's) of a list or tuple, passed as a parameter

Eg:-
```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(sum(a))
```

Output: 45//

max() The max() function returns the item with the highest value, or the item with the highest value in an iterable. If the elements are string then it will return the string with maximum lexicographic value

Eg:-

```
var 1 = 1
var 2 = 8
var 3 = 2

max_val = max(var1, var2, var3)
print ( max_ val)
```

Output :- 8//

```
var 1 = "geeks"
var 2 = "for"
var 3 = "geek"

max_val = max(var1, var2, var3)
print (max_val)
```

Output :- geeks

min() : The min() function returns the smallest of all the values or the smallest item in an iterable passed as its parameter

Eg:-

```
print (min( [1, 2, 3])  => 1   -> output
print (min ({'a': 1, 'b': 2, 'c': 3}))  => 'a'   -> output
print (min (7, 9, 11))  => 7//   -> output.
```

15. (R) Explain the methods of List data type in Python for the following operations with suitable code snippets for each.

(I) Adding values to a list
(II) Removing values from a list
(III) Finding a value in the list
(IV) Sorting the values in a list [-9M-] [2018-19]

V.V.V Imp

Ans For (I) ⇒ refer pg no 18 ~~to 3~~ //

For (II) ⇒ refer pg no 18 [i.e removing values from list with del statements]

For (III) ⇒ refer pg no 3 to 4 [index]

For (IV) ⇒ refer pg no 19//

16. Write a Python Program that accepts a sentence & find the no of words, digits, uppercase letters & lowercase letters.
[-6M-] [2018-19]

Ans.

```
sentence = input ("Enter a sentence")
words = len (sentence.split())
digits = sum (c.isdigit() for c in sentence)
uppercase = sum (c.isupper() for c in sentence)
lowercase = sum (c.islower() for c in sentence)

print ("words : {words}")
print ("Digits : {digits}")
print ("Uppercase letters : {uppercase}")
print ("Lowercase letters : {lowercase}")
```

How it Works :

1] sentence.split() → splits the sentence into words (separated by spaces).

2] c.isdigit(), c.isupper() & c.islower() check for digits, upper case letters, & lowercase letters, respectively for each character in the sentence.

3] sum() → counts how many characters match the condition in each case.

Example :

```
Enter a sentence : Hello World 123
Words            : 3
Digits           : 3
Upper Case letters : 2
Lowercase letters : 8
```

17] Discuss the following Dictionary methods in Python with example

(R) (i) get() (ii) items() (iii) keys() (iv) values()

[8M-] [2018-19] [V.V.V imp]

Ans Chapter 5, Refer pg no 25 to 26.

(i) → get() method is used to access the value of a specific key in a dictionary. If the key doesn't exist, it won't throw an error, it can return a default value instead.

Eg:- my_dict = {"name": "Alice", "age": 25, "city": "New York"}

```
# Get the value of the "name" key
print(my_dict.get("name")) → output : Alice
# Get a non-existing key with a default value.
print(my_dict.get("country", "Not found")) → output : Not found
```

(ii) items() method :-

↳ The items() method returns all key value pairs as a list of tuple. Each tuple contains one key & its value.

```
my_dict = {"name": "Alice", "age": 25}
# Get all key-value pairs
print(my_dict.items())
# Output : dict_items([('name', 'Alice'), ('age', 25)])

# Loop through key-value pairs
for key, value in my_dict.items():
    print("{key}: {value}")
# Output
name: Alice
age: 25
```

(iii) keys() Method :-

The key() method returns all the keys in the dictionary.

```
my_dict = {"name": "Alice", "age": 25, "city": "New York"}
# Get all keys
print(my_dict.keys())
```

```
# Output : dict_keys (['name', 'age', 'city'])
# Loop through all keys.
    for key in my_dict.keys():
        print (key)
# Output
    name
    age
    city
```

(iv) values() Method.

The values() method returns all the values in the dictionary.

```
my_dict = {"name": "Alice", "age": 25}
# Get all the values.
    print (my_dict.values())
# Output : dict_values (['Alice', 25])
# Loop through values.
        for value in my_dict.values():
            print (value)
# Output :
    Alice
    25
```

Conclusion :-

Method	What it does	Example Output
get (key)	Gets a value by key (safe)	"Alice" or "Not found"
items()	Returns all key value pairs	[('name', 'Alice'), ('age', 25)]
keys()	Returns all the keys	['name', 'age']
values()	Returns all the values	['Alice', 25]

18] Explain the various string methods for the following operations with examples.

(i) Removing whitespace characters from the beginning end or both sides of a string.

(ii) To right-justify, left-justify & center a string [-GM-]

Ans (i) & (ii) Removing Whitespace Characters

To remove spaces (or other unwanted characters) from the beggining, end or both sides of a string.

We use :-

1. strip() → Removes whitespaces (or specified characters) from both sides of the string.

2. lstrip() → Removes whitespaces (or specified characters) from the left side (beginning)

3. rstrip() → Removes whitespaces (or specified characters) from the right side (end)

Eg :-

```
text = "    Hello World!  "
# Removes spaces from both sides
print(text.strip())      # Output : "Hello World!"
# Removes spaces from the left sede only
print(text.lstrip())     # Output : "Hello world!  "
# Remove spaces from the right side only
print(text.rstrip())     # Output : "    Hello World!"
```

String Justification

To align a string (right, left or center) we use these methods

1] rjust(width) → Adds spaces to the left of the string to make it right aligned

2] ljust(width) → Adds spaces to the right of the string to make it left-aligned.

3] center(width) → Adds spaces to both sides to make the string centered.

Conclusion :-

* Removing Whitespace :-
  - strip() → Both sides
  - lstrip() → Left only
  - rstrip() → Right only

* Aligning Text :-
  - rjust(width) → Right-align (add spaces to the left)
  - ljust(width) → Left-align (add spaces to the right)
  - center(width) → center-align (add spaces on both sides)

These methods make text formatting & cleanup super simple in Python.