Zadania z GitHub – python

STRING

1. **Write a Python program to calculate the length of a string**

def calculate\_string\_length(string):

return len(string)

# Example usage:

input\_string = input("Enter a string: ")

length = calculate\_string\_length(input\_string)

print("Length of the string:", length)

1. **Write a Python program to count the number of characters (character frequency) in a string.**

**Sample String : google.com'**

**Expected Result : {'g': 2, 'o': 3, 'l': 1, 'e': 1, '.': 1, 'c': 1, 'm': 1}**

def count\_characters(string):

# Initialize an empty dictionary to store character frequencies

char\_frequency = {}

# Iterate through each character in the string

for char in string:

# Increment the count of the character in the dictionary

char\_frequency[char] = char\_frequency.get(char, 0) + 1

return char\_frequency

# Example usage:

sample\_string = "google.com"

result = count\_characters(sample\_string)

print("Character frequency:", result)

1. **Write a Python program to get a string made of the first 2 and last 2 characters of a given string. If the string length is less than 2, return the empty string instead.**

**Sample String : 'w3resource'**

**Expected Result : 'w3ce'**

**Sample String : 'w3'**

**Expected Result : 'w3w3'**

**Sample String : ' w'**

**Expected Result : Empty String**

def get\_first\_and\_last\_two(string):

# Check if the string length is less than 2

if len(string) < 2:

return "" # Return an empty string

# Extract the first two and last two characters and concatenate them

result = string[:2] + string[-2:]

return result

# Example usage:

sample\_strings = ['w3resource', 'w3', ' w']

for sample\_string in sample\_strings:

result = get\_first\_and\_last\_two(sample\_string)

print(f"Sample String: '{sample\_string}' -> Expected Result: '{result}'")

1. **Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself.**

**Sample String : 'restart'**

**Expected Result : 'resta$t'**

def replace\_first\_char\_occurrences(string):

# Get the first character of the string

first\_char = string[0]

# Replace all occurrences of the first character (except the first one) with '$'

replaced\_string = first\_char + string[1:].replace(first\_char, '$')

return replaced\_string

# Example usage:

sample\_string = 'restart'

result = replace\_first\_char\_occurrences(sample\_string)

print("Expected Result:", result)

1. **Write a Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string.**

**Sample String : 'abc', 'xyz'**

**Expected Result : 'xyc abz'**

def swap\_and\_concatenate(string1, string2):

# Swap the first two characters of each string

swapped\_string1 = string2[:2] + string1[2:]

swapped\_string2 = string1[:2] + string2[2:]

# Concatenate the swapped strings with a space in between

result = swapped\_string1 + ' ' + swapped\_string2

return result

# Example usage:

string1 = 'abc'

string2 = 'xyz'

result = swap\_and\_concatenate(string1, string2)

print("Expected Result:", result)

1. **Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing', add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged.**

**Sample String : 'abc'**

**Expected Result : 'abcing'**

**Sample String : 'string'**

**Expected Result : 'stringly'**

def add\_ing\_ly(string):

# Check if the length of the string is less than 3

if len(string) < 3:

return string # Leave it unchanged

# Check if the string already ends with 'ing'

if string.endswith('ing'):

result = string + 'ly' # Add 'ly' at the end

else:

result = string + 'ing' # Add 'ing' at the end

return result

# Example usage:

sample\_strings = ['abc', 'string']

for sample\_string in sample\_strings:

result = add\_ing\_ly(sample\_string)

print(f"Sample String: '{sample\_string}' -> Expected Result: '{result}'")

1. **Write a Python program to find the first appearance of the substrings 'not' and 'poor' in a given string. If 'not' follows 'poor', replace the whole 'not'...'poor' substring with 'good'. Return the resulting string.  
   Sample String : 'The lyrics is not that poor!'  
   'The lyrics is poor!'  
   Expected Result : 'The lyrics is good!'  
   'The lyrics is poor!'**

def replace\_not\_poor(string):

# Find the positions of 'not' and 'poor' in the string

index\_not = string.find('not')

index\_poor = string.find('poor')

# Check if both 'not' and 'poor' are present and 'not' appears before 'poor'

if index\_not != -1 and index\_poor != -1 and index\_not < index\_poor:

# Replace the substring between 'not' and 'poor' with 'good'

string = string[:index\_not] + 'good' + string[index\_poor + 4:]

return string

# Example usage:

sample\_strings = [

'The lyrics is not that poor!',

'The lyrics is poor!'

]

for sample\_string in sample\_strings:

result = replace\_not\_poor(sample\_string)

print(f"Sample String: '{sample\_string}' -> Expected Result: '{result}'")

1. **Write a Python function that takes a list of words and return the longest word and the length of the longest one.  
   Sample Output:  
   Longest word: Exercises  
   Length of the longest word: 9**

def find\_longest\_word(words):

# Initialize variables to store the longest word and its length

longest\_word = ''

longest\_length = 0

# Iterate through each word in the list

for word in words:

# Check if the current word is longer than the previous longest word

if len(word) > longest\_length:

longest\_word = word

longest\_length = len(word)

return longest\_word, longest\_length

# Example usage:

word\_list = ['Python', 'Java', 'C++', 'Exercises', 'Programming']

longest\_word, length = find\_longest\_word(word\_list)

print("Longest word:", longest\_word)

print("Length of the longest word:", length)

1. **Write a Python program to remove the nth index character from a nonempty string.**

def remove\_nth\_character(string, n):

# Check if the string is non-empty and n is a valid index

if string and 0 <= n < len(string):

# Return the string with the nth character removed

return string[:n] + string[n+1:]

else:

return "Invalid input or index out of range"

# Example usage:

input\_string = "Python"

n = 2

result = remove\_nth\_character(input\_string, n)

print("Result:", result)

**10.** **Write a Python program to change a given string to a newly string where the first and last chars have been exchanged.**  
def exchange\_first\_last\_chars(string):

# Check if the string has at least two characters

if len(string) >= 2:

# Swap the first and last characters

new\_string = string[-1] + string[1:-1] + string[0]

return new\_string

else:

return string # Return the string unchanged if it has less than two characters

# Example usage:

input\_string = "example"

result = exchange\_first\_last\_chars(input\_string)

print("Result:", result)

**11.** **Write a Python program to remove characters that have odd index values in a given string.**  
def remove\_odd\_index\_characters(string):

# Remove characters at odd index positions using string slicing

result = string[::2]

return result

# Example usage:

input\_string = "Python"

result = remove\_odd\_index\_characters(input\_string)

print("Result:", result)

**12.** **Write a Python program to count the occurrences of each word in a given sentence**.  
def count\_word\_occurrences(sentence):

# Split the sentence into words

words = sentence.split()

# Initialize an empty dictionary to store word occurrences

word\_count = {}

# Iterate through each word in the list of words

for word in words:

# Increment the count of the word in the dictionary

word\_count[word] = word\_count.get(word, 0) + 1

return word\_count

# Example usage:

input\_sentence = "This is a sentence is with repeated words. This sentence has repeated words."

result = count\_word\_occurrences(input\_sentence)

print("Word occurrences:", result)

**13.** **Write a Python script that takes input from the user and displays that input back in upper and lower cases.**  
# Get input from the user

user\_input = input("Enter a string: ")

# Convert the input to uppercase and lowercase

uppercase\_input = user\_input.upper()

lowercase\_input = user\_input.lower()

# Display the input in upper and lower cases

print("Input in uppercase:", uppercase\_input)

print("Input in lowercase:", lowercase\_input)

**14.** **Write a Python program that accepts a comma-separated sequence of words as input and prints the distinct words in sorted form (alphanumerically).  
Sample Words : red, white, black, red, green, black  
Expected Result : black, green, red, white,red**def print\_sorted\_distinct\_words(words):

# Split the input string based on commas and convert it to a set to remove duplicates

distinct\_words = set(words.split(', '))

# Sort the set alphanumerically

sorted\_words = sorted(distinct\_words)

# Print the sorted distinct words

print(", ".join(sorted\_words))

# Example usage:

input\_words = "red, white, black, red, green, black"

print("Expected Result:", end=" ")

print\_sorted\_distinct\_words(input\_words)

**15.** **Write a Python function to create an HTML string with tags around the word(s).  
Sample function and result :  
add\_tags('i', 'Python') -> '<i>Python</i>'  
add\_tags('b', 'Python Tutorial') -> '<b>Python Tutorial </b>'**  
def add\_tags(tag, content):

# Construct the HTML string with the specified tag around the content

html\_string = f"<{tag}>{content}</{tag}>"

return html\_string

# Example usage:

result1 = add\_tags('i', 'Python')

print("Result 1:", result1)

result2 = add\_tags('b', 'Python Tutorial')

print("Result 2:", result2)

**16.** **Write a Python function to insert a string in the middle of a string.  
Sample function and result :  
insert\_sting\_middle('[[]]<<>>', 'Python') -> [[Python]]  
insert\_sting\_middle('{{}}', 'PHP') -> {{PHP}}**  
def insert\_string\_middle(original, content):

# Calculate the index to insert the content in the middle of the original string

middle\_index = len(original) // 2

# Insert the content in the middle of the original string

new\_string = original[:middle\_index] + content + original[middle\_index:]

return new\_string

# Example usage:

result1 = insert\_string\_middle('[[]]<<>>', 'Python')

print("Result 1:", result1)

result2 = insert\_string\_middle('{{}}', 'PHP')

print("Result 2:", result2)

**17.** **Write a Python function to get a string made of 4 copies of the last two characters of a specified string (length must be at least 2).  
Sample function and result :  
insert\_end('Python') -> onononon  
insert\_end('Exercises') -> eseseses**  
def insert\_end(string):

# Check if the string length is at least 2

if len(string) >= 2:

# Extract the last two characters and create a new string with 4 copies

result = string[-2:] \* 4

return result

else:

return "Length of the string must be at least 2"

# Example usage:

result1 = insert\_end('Python')

print("Result 1:", result1)

result2 = insert\_end('Exercises')

print("Result 2:", result2)

**18.** **Write a Python function to get a string made of the first three characters of a specified string. If the length of the string is less than 3, return the original string.  
Sample function and result :  
first\_three('ipy') -> ipy  
first\_three('python') -> pyt**def first\_three(string):

# Check if the length of the string is at least 3

if len(string) >= 3:

# Extract the first three characters

result = string[:3]

return result

else:

return string # Return the original string if its length is less than 3

# Example usage:

result1 = first\_three('ipy')

print("Result 1:", result1)

result2 = first\_three('python')

print("Result 2:", result2)

**19. Write a Python program to get the last part of a string before a specified character.**  
def get\_last\_part\_before\_character(string, char):

# Find the index of the specified character in the string

index = string.find(char)

# Check if the specified character is found in the string

if index != -1:

# Return the substring from the beginning of the string to the index of the specified character

return string[:index]

else:

# If the specified character is not found, return the original string

return string

# Example usage:

url1 = "https://www.w3resource.com/python-exercises"

char1 = '-'

result1 = get\_last\_part\_before\_character(url1, char1)

print("Result 1:", result1)

url2 = "https://www.w3resource.com/python"

char2 = '/'

result2 = get\_last\_part\_before\_character(url2, char2)

print("Result 2:", result2)

**20.** **Write a Python function to reverse a string if its length is a multiple of 4.**  
def reverse\_string\_if\_multiple\_of\_four(string):

# Check if the length of the string is a multiple of 4

if len(string) % 4 == 0:

# Reverse the string

reversed\_string = string[::-1]

return reversed\_string

else:

return string # Return the original string if its length is not a multiple of 4

# Example usage:

result1 = reverse\_string\_if\_multiple\_of\_four("abcd")

print("Result 1:", result1)

result2 = reverse\_string\_if\_multiple\_of\_four("python")

print("Result 2:", result2)

result3 = reverse\_string\_if\_multiple\_of\_four("hello")

print("Result 3:", result3)

result4 = reverse\_string\_if\_multiple\_of\_four("abcdefgh")

print("Result 4:", result4)

----------------------------------------------------------------------------------------------------------------------------------------------

LIST

1**. Write a Python program to sum all the items in a list.**  
def sum\_list(items):

total = 0

for item in items:

total += item

return total

# Example usage:

my\_list = [1, 2, 3, 4, 5]

print("Sum of all items:", sum\_list(my\_list))

**2.** **Write a Python program to multiply all the items in a list.**def multiply\_list(items):

result = 1

for item in items:

result \*= item

return result

# Example usage:

my\_list = [1, 2, 3, 4, 5]

print("Product of all items:", multiply\_list(my\_list))

**3. Write a Python program to get the largest number from a list.**def get\_largest\_number(items):

return max(items)

# Example usage:

my\_list = [1, 3, 5, 7, 9]

print("Largest number:", get\_largest\_number(my\_list))

**4. Write a Python program to get the smallest number from a list.**def get\_smallest\_number(items):

return min(items)

# Example usage:

my\_list = [1, 3, 5, 7, 9]

print("Smallest number:", get\_smallest\_number(my\_list))

**5. Write a Python program to count the number of strings from a given list of strings. The string length is 2 or more and the first and last characters are the same.  
Sample List : ['abc', 'xyz', 'aba', '1221']  
Expected Result : 2**def count\_strings\_with\_same\_first\_and\_last\_char(strings):

count = 0

for string in strings:

if len(string) >= 2 and string[0] == string[-1]:

count += 1

return count

# Example usage:

my\_list = ['abc', 'xyz', 'aba', '1221']

print("Number of strings with the same first and last characters:", count\_strings\_with\_same\_first\_and\_last\_char(my\_list))

**6. Write a Python program to get a list, sorted in increasing order by the last element in each tuple from a given list of non-empty tuples.  
Sample List : [(2, 5), (1, 2), (4, 4), (2, 3), (2, 1)]  
Expected Result : [(2, 1), (1, 2), (2, 3), (4, 4), (2, 5)]**def sort\_list\_by\_last\_element(tuples):

return sorted(tuples, key=lambda x: x[-1])

# Example usage:

my\_list = [(2, 5), (1, 2), (4, 4), (2, 3), (2, 1)]

print("Sorted list by last element:", sort\_list\_by\_last\_element(my\_list))

**7. Write a Python program to remove duplicates from a list.**def remove\_duplicates(items):

return list(set(items))

# Example usage:

my\_list = [1, 2, 3, 2, 4, 5, 3, 1]

print("List without duplicates:", remove\_duplicates(my\_list))

**8. Write a Python program to check if a list is empty or not.**def is\_list\_empty(items):

return len(items) == 0

# Example usage:

my\_list = []

print("Is the list empty?", is\_list\_empty(my\_list))

**9. Write a Python program to clone or copy a list.**def clone\_list(items):

return items.copy()

# Example usage:

my\_list = [1, 2, 3, 4, 5]

cloned\_list = clone\_list(my\_list)

print("Cloned list:", cloned\_list)

**10. Write a Python program to find the list of words that are longer than n from a given list of words.**def words\_longer\_than\_n(words, n):

return [word for word in words if len(word) > n]

# Example usage:

word\_list = ['apple', 'banana', 'grape', 'orange', 'strawberry']

n = 5

print("Words longer than", n, "characters:", words\_longer\_than\_n(word\_list, n))

**11. Write a Python function that takes two lists and returns True if they have at least one common member.**def have\_common\_member(list1, list2):

return any(item in list2 for item in list1)

# Example usage:

list1 = [1, 2, 3, 4, 5]

list2 = [5, 6, 7, 8, 9]

print("Do the lists have at least one common member?", have\_common\_member(list1, list2))

**12. Write a Python program to print a specified list after removing the 0th, 4th and 5th elements.  
Sample List : ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow']  
Expected Output : ['Green', 'White', 'Black']**def remove\_elements(sample\_list):

indices\_to\_remove = [0, 4, 5]

return [sample\_list[i] for i in range(len(sample\_list)) if i not in indices\_to\_remove]

# Example usage:

sample\_list = ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow']

print("List after removing specified elements:", remove\_elements(sample\_list))

**13. Write a Python program to generate a 3\*4\*6 3D array whose each element is \*.**def generate\_3d\_array(x, y, z):

return [[['\*' for \_ in range(z)] for \_ in range(y)] for \_ in range(x)]

# Example usage:

x = 3

y = 4

z = 6

print("3D array with asterisks:")

for row in generate\_3d\_array(x, y, z):

print(row)

**14. Write a Python program to print the numbers of a specified list after removing even numbers from it.**def remove\_even\_numbers(numbers):

return [num for num in numbers if num % 2 != 0]

# Example usage:

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

print("Numbers after removing even numbers:", remove\_even\_numbers(numbers))

**15. Write a Python program to shuffle and print a specified list.**

import random

def shuffle\_list(sample\_list):

shuffled\_list = sample\_list.copy()

random.shuffle(shuffled\_list)

return shuffled\_list

# Example usage:

sample\_list = ['a', 'b', 'c', 'd', 'e']

print("Shuffled list:", shuffle\_list(sample\_list))

**16. Write a Python program to generate and print a list of the first and last 5 elements where the values are square numbers between 1 and 30 (both included).**def generate\_square\_numbers():

square\_numbers = [x\*\*2 for x in range(1, 31) if x\*\*2 <= 30]

return square\_numbers[:5] + square\_numbers[-5:]

# Example usage:

print("First and last 5 square numbers between 1 and 30:", generate\_square\_numbers())

**17. Write a Python program to check if each number is prime in a given list of numbers. Return True if all numbers are prime otherwise False.  
Sample Data:  
([0, 3, 4, 7, 9]) -> False  
([3, 5, 7, 13]) -> True  
([1, 5, 3]) -> False**def is\_prime(num):

if num < 2:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

def check\_all\_primes(numbers):

return all(is\_prime(num) for num in numbers)

# Example usage:

sample\_data = [0, 3, 4, 7, 9]

print("Are all numbers prime?", check\_all\_primes(sample\_data))

**18. Write a Python program to generate all permutations of a list in Python.**import itertools

def generate\_permutations(input\_list):

return list(itertools.permutations(input\_list))

# Example usage:

my\_list = [1, 2, 3]

print("All permutations of the list:", generate\_permutations(my\_list))

**19. Write a Python program to calculate the difference between the two lists.**def calculate\_difference(list1, list2):

return list(set(list1) - set(list2))

# Example usage:

list1 = [1, 2, 3, 4, 5]

list2 = [3, 4, 5, 6, 7]

print("Difference between the two lists:", calculate\_difference(list1, list2))

**20. Write a Python program to access the index of a list.**

def access\_index(sample\_list, item):

if item in sample\_list:

return sample\_list.index(item)

else:

return None

# Example usage:

my\_list = ['a', 'b', 'c', 'd', 'e']

item = 'c'

print("Index of", item, "in the list:", access\_index(my\_list, item))

**DICTIONARY**

**1. Write a Python script to sort (ascending and descending) a dictionary by value.**

# Sample dictionary

my\_dict = {'a': 20, 'b': 10, 'c': 30, 'd': 5}

# Sort the dictionary by value in ascending order

sorted\_asc = dict(sorted(my\_dict.items(), key=lambda item: item[1]))

# Sort the dictionary by value in descending order

sorted\_desc = dict(sorted(my\_dict.items(), key=lambda item: item[1], reverse=True))

print("Ascending order:", sorted\_asc)

print("Descending order:", sorted\_desc)

**2. Write a Python script to add a key to a dictionary.**

**Sample Dictionary : {0: 10, 1: 20}  
Expected Result : {0: 10, 1: 20, 2: 30}**

# Sample dictionary

sample\_dict = {0: 10, 1: 20}

# Add a key to the dictionary

sample\_dict[2] = 30

print("Updated dictionary:", sample\_dict)

**3. Write a Python script to concatenate the following dictionaries to create a new one.**

**Sample Dictionary :  
dic1={1:10, 2:20}  
dic2={3:30, 4:40}  
dic3={5:50,6:60}  
Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}**

# Sample dictionaries

dic1 = {1: 10, 2: 20}

dic2 = {3: 30, 4: 40}

dic3 = {5: 50, 6: 60}

# Concatenate dictionaries

new\_dict = {\*\*dic1, \*\*dic2, \*\*dic3}

print("Concatenated dictionary:", new\_dict)

**4. Write a Python script to check whether a given key already exists in a dictionary.**

# Sample dictionary

sample\_dict = {0: 10, 1: 20, 2: 30}

# Key to check

key\_to\_check = 1

# Check if the key exists in the dictionary

if key\_to\_check in sample\_dict:

print("Key", key\_to\_check, "exists in the dictionary")

else:

print("Key", key\_to\_check, "does not exist in the dictionary")

**5. Write a Python program to iterate over dictionaries using for loops.**

# Sample dictionary

my\_dict = {'a': 1, 'b': 2, 'c': 3}

# Iterate over keys

print("Keys:")

for key in my\_dict:

print(key)

# Iterate over values

print("\nValues:")

for value in my\_dict.values():

print(value)

# Iterate over items (key-value pairs)

print("\nKey-Value pairs:")

for key, value in my\_dict.items():

print(key, "->", value)

**6. Write a Python script to generate and print a dictionary that contains a number (between 1 and n) in the form (x, x\*x).  
Sample Dictionary ( n = 5) :  
Expected Output : {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}**def generate\_square\_dictionary(n):

square\_dict = {}

for x in range(1, n+1):

square\_dict[x] = x\*x

return square\_dict

# Example usage:

n = 5

print("Generated dictionary:", generate\_square\_dictionary(n))

**7. Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are the square of the keys.  
Sample Dictionary  
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225}**

square\_dict = {x: x\*x for x in range(1, 16)}

print("Sample Dictionary:", square\_dict)

**8. Write a Python script to merge two Python dictionaries.**def merge\_dicts(dict1, dict2):

return {\*\*dict1, \*\*dict2}

# Example usage:

dict1 = {'a': 1, 'b': 2}

dict2 = {'c': 3, 'd': 4}

merged\_dict = merge\_dicts(dict1, dict2)

print("Merged dictionary:", merged\_dict)

**9. Write a Python program to iterate over dictionaries using for loops.**# Sample dictionary

my\_dict = {'a': 1, 'b': 2, 'c': 3}

# Iterate over keys

print("Keys:")

for key in my\_dict:

print(key)

# Iterate over values

print("\nValues:")

for value in my\_dict.values():

print(value)

# Iterate over items (key-value pairs)

print("\nKey-Value pairs:")

for key, value in my\_dict.items():

print(key, "->", value)

**10. Write a Python program to sum all the items in a dictionary.**def sum\_items(dictionary):

return sum(dictionary.values())

# Example usage:

my\_dict = {'a': 1, 'b': 2, 'c': 3}

print("Sum of all items in the dictionary:", sum\_items(my\_dict))

**11. Write a Python program to multiply all the items in a dictionary.**def multiply\_items(dictionary):

result = 1

for value in dictionary.values():

result \*= value

return result

# Example usage:

my\_dict = {'a': 2, 'b': 3, 'c': 4}

print("Product of all items in the dictionary:", multiply\_items(my\_dict))

**12. Write a Python program to remove a key from a dictionary.**def remove\_key(dictionary, key):

if key in dictionary:

del dictionary[key]

return dictionary

# Example usage:

my\_dict = {'a': 1, 'b': 2, 'c': 3}

key\_to\_remove = 'b'

print("Dictionary after removing key:", remove\_key(my\_dict, key\_to\_remove))

**13. Write a Python program to map two lists into a dictionary.**def map\_lists\_to\_dict(keys, values):

return dict(zip(keys, values))

# Example usage:

keys = ['a', 'b', 'c']

values = [1, 2, 3]

print("Mapped dictionary:", map\_lists\_to\_dict(keys, values))

**14. Write a Python program to sort a given dictionary by key.**def sort\_dict\_by\_key(dictionary):

return dict(sorted(dictionary.items()))

# Example usage:

my\_dict = {'b': 2, 'a': 1, 'c': 3}

print("Sorted dictionary by key:", sort\_dict\_by\_key(my\_dict))

**15. Write a Python program to get the maximum and minimum values of a dictionary.**def get\_max\_min\_values(dictionary):

max\_value = max(dictionary.values())

min\_value = min(dictionary.values())

return max\_value, min\_value

# Example usage:

my\_dict = {'a': 1, 'b': 2, 'c': 3}

max\_value, min\_value = get\_max\_min\_values(my\_dict)

print("Maximum value:", max\_value)

print("Minimum value:", min\_value)

**16. Write a Python program to get a dictionary from an object's fields.**class Student:

def \_\_init\_\_(self, name, age, grade):

self.name = name

self.age = age

self.grade = grade

def object\_to\_dict(obj):

return obj.\_\_dict\_\_

# Example usage:

student1 = Student("Alice", 18, "A")

student\_dict = object\_to\_dict(student1)

print("Dictionary from object:", student\_dict)

**17. Write a Python program to remove duplicates from the dictionary.**def remove\_duplicates\_from\_dict(dictionary):

return {k: v for k, v in dict.fromkeys(dictionary).items()}

# Example usage:

my\_dict = {'a': 1, 'b': 2, 'c': 1, 'd': 3}

print("Dictionary without duplicates:", remove\_duplicates\_from\_dict(my\_dict))

**18. Write a Python program to check if a dictionary is empty or not.**def is\_dict\_empty(dictionary):

return len(dictionary) == 0

# Example usage:

empty\_dict = {}

print("Is the dictionary empty?", is\_dict\_empty(empty\_dict))

**19. Write a Python program to combine two dictionary by adding values for common keys.  
d1 = {'a': 100, 'b': 200, 'c':300}  
d2 = {'a': 300, 'b': 200, 'd':400}  
Sample output: Counter({'a': 400, 'b': 400, 'd': 400, 'c': 300})**from collections import Counter

def combine\_dictionaries(d1, d2):

return Counter(d1) + Counter(d2)

# Example usage:

d1 = {'a': 100, 'b': 200, 'c': 300}

d2 = {'a': 300, 'b': 200, 'd': 400}

combined\_dict = combine\_dictionaries(d1, d2)

print("Combined dictionary with added values for common keys:", combined\_dict)

**20. Write a Python program to print all distinct values in a dictionary.  
Sample Data : [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S005"}, {"V":"S009"},{"VIII":"S007"}]  
Expected Output : Unique Values: {'S005', 'S002', 'S007', 'S001', 'S009'}**

def get\_unique\_values(dictionary\_list):

values = set()

for d in dictionary\_list:

values.update(d.values())

return values

# Example usage:

sample\_data = [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S005"}, {"V":"S009"},{"VIII":"S007"}]

print("Unique Values:", get\_unique\_values(sample\_data))

STATEMENTS AND LOOPS

**1.Write a Python program to find those numbers which are divisible by 7 and multiples of 5, between 1500 and 2700 (both included).**result = [num for num in range(1500, 2701) if num % 7 == 0 and num % 5 == 0]

print("Numbers divisible by 7 and multiples of 5 between 1500 and 2700:", result)

**2. Write a Python program to convert temperatures to and from Celsius and Fahrenheit.  
[ Formula : c/5 = f-32/9 [ where c = temperature in celsius and f = temperature in fahrenheit ]  
*Expected Output* :  
60°C is 140 in Fahrenheit  
45°F is 7 in Celsius**def celsius\_to\_fahrenheit(celsius):

return celsius \* 9/5 + 32

def fahrenheit\_to\_celsius(fahrenheit):

return (fahrenheit - 32) \* 5/9

# Example usage:

celsius\_temp = 60

fahrenheit\_temp = 45

print(celsius\_temp, "°C is", celsius\_to\_fahrenheit(celsius\_temp), "in Fahrenheit")

print(fahrenheit\_temp, "°F is", fahrenheit\_to\_celsius(fahrenheit\_temp), "in Celsius")

**3. Write a Python program to guess a number between 1 and 9.  
Note : User is prompted to enter a guess. If the user guesses wrong then the prompt appears again until the guess is correct, on successful guess, user will get a "Well guessed!" message, and the program will exit.**import random

number\_to\_guess = random.randint(1, 9)

while True:

guess = int(input("Guess a number between 1 and 9: "))

if guess == number\_to\_guess:

print("Well guessed!")

break

else:

print("Try again!")

**4. Write a Python program to construct the following pattern, using a nested for loop.**

**\*   
\* \*   
\* \* \*   
\* \* \* \*   
\* \* \* \* \*   
\* \* \* \*   
\* \* \*   
\* \*   
\***

for i in range(6):

print('\* ' \* i)

for i in range(4, 0, -1):

print('\* ' \* i)

**5. Write a Python program that accepts a word from the user and reverses it.**word = input("Enter a word: ")

print("Reversed word:", word[::-1])

**6. Write a Python program to count the number of even and odd numbers in a series of numbers  
*Sample numbers* : numbers = (1, 2, 3, 4, 5, 6, 7, 8, 9)   
*Expected Output* :  
Number of even numbers : 5  
Number of odd numbers : 4**numbers = (1, 2, 3, 4, 5, 6, 7, 8, 9)

even\_count = sum(1 for num in numbers if num % 2 == 0)

odd\_count = len(numbers) - even\_count

print("Number of even numbers:", even\_count)

print("Number of odd numbers:", odd\_count)

**7. Write a Python program that prints each item and its corresponding type from the following list.  
*Sample List* : datalist = [1452, 11.23, 1+2j, True, 'w3resource', (0, -1), [5, 12], {"class":'V', "section":'A'}]**datalist = [1452, 11.23, 1+2j, True, 'w3resource', (0, -1), [5, 12], {"class":'V', "section":'A'}]

for item in datalist:

print(item, "->", type(item))

**8. Write a Python program that prints all the numbers from 0 to 6 except 3 and 6.  
Note : Use 'continue' statement.  
Expected Output : 0 1 2 4 5**for num in range(7):

if num == 3 or num == 6:

continue

print(num, end=" ")

**9. Write a Python program to get the Fibonacci series between 0 and 50.  
Note : The Fibonacci Sequence is the series of numbers :  
0, 1, 1, 2, 3, 5, 8, 13, 21, ....  
Every next number is found by adding up the two numbers before it.  
Expected Output : 1 1 2 3 5 8 13 21 34**a, b = 0, 1

while a < 50:

print(a, end=" ")

a, b = b, a + b

**10. Write a Python program that iterates the integers from 1 to 50. For multiples of three print "Fizz" instead of the number and for multiples of five print "Buzz". For numbers that are multiples of three and five, print "FizzBuzz".  
*Sample Output* :  
fizzbuzz  
1  
2  
fizz  
4  
buzz**for num in range(1, 51):

if num % 3 == 0 and num % 5 == 0:

print("FizzBuzz")

elif num % 3 == 0:

print("Fizz")

elif num % 5 == 0:

print("Buzz")

else:

print(num)

**11. Write a Python program that takes two digits m (row) and n (column) as input and generates a two-dimensional array. The element value in the i-th row and j-th column of the array should be i\*j.  
Note :  
i = 0,1.., m-1  
j = 0,1, n-1.**

**Test Data : Rows = 3, Columns = 4  
Expected Result : [[0, 0, 0, 0], [0, 1, 2, 3], [0, 2, 4, 6]]**def generate\_2d\_array(rows, cols):

return [[i\*j for j in range(cols)] for i in range(rows)]

# Example usage:

rows = 3

cols = 4

result = generate\_2d\_array(rows, cols)

print("Generated 2D array:", result)

**12. Write a Python program that accepts a sequence of lines (blank line to terminate) as input and prints the lines as output (all characters in lower case).**print("Enter lines (type blank line to terminate):")

lines = []

while True:

line = input()

if not line:

break

lines.append(line.lower())

print("Lines in lowercase:")

for line in lines:

print(line)

**13. Write a Python program that accepts a sequence of comma separated 4 digit binary numbers as its input. The program will print the numbers that are divisible by 5 in a comma separated sequence.  
Sample Data : 0100,0011,1010,1001,1100,1001  
Expected Output : 1010**binary\_numbers = input("Enter comma-separated 4 digit binary numbers: ").split(',')

result = [num for num in binary\_numbers if int(num, 2) % 5 == 0]

print("Numbers divisible by 5:", ','.join(result))

**14. Write a Python program that accepts a string and calculates the number of digits and letters.  
Sample Data : Python 3.2  
Expected Output :  
Letters 6  
Digits 2**text = input("Enter a string: ")

letters = sum(1 for char in text if char.isalpha())

digits = sum(1 for char in text if char.isdigit())

print("Letters:", letters)

print("Digits:", digits)

**15. Write a Python program to check the validity of passwords input by users.  
Validation :**

* **At least 1 letter between [a-z] and 1 letter between [A-Z].**
* **At least 1 number between [0-9].**
* **At least 1 character from [$#@].**
* **Minimum length 6 characters.**
* **Maximum length 16 characters.**

import re

def validate\_password(password):

if len(password) < 6 or len(password) > 16:

return False

if not re.search("[a-z]", password):

return False

if not re.search("[A-Z]", password):

return False

if not re.search("[0-9]", password):

return False

if not re.search("[$#@]", password):

return False

return True

password = input("Enter a password: ")

if validate\_password(password):

print("Valid password")

else:

print("Invalid password")

**16. Write a Python program to find numbers between 100 and 400 (both included) where each digit of a number is an even number. The numbers obtained should be printed in a comma-separated sequence.**result = [str(num) for num in range(100, 401) if all(int(digit) % 2 == 0 for digit in str(num))]

print("Numbers with even digits between 100 and 400:", ','.join(result))

**17. Write a Python program to print the alphabet pattern 'A'.  
*Expected Output:***

**\*\*\***

**\* \***

**\* \***

**\*\*\*\*\***

**\* \***

**\* \***

**\* \***

print(" \*\*\* ")

print(" \* \* ")

print(" \* \* ")

print(" \*\*\*\*\* ")

print(" \* \* ")

print(" \* \* ")

print(" \* \* ")

**18. Write a Python program to print the alphabet pattern 'D'.  
*Expected Output:***

**\*\*\*\***

**\* \***

**\* \***

**\* \***

**\* \***

**\* \***

**\*\*\*\***

print(" \*\*\*\* ")

print(" \* \* ")

print(" \* \* ")

print(" \* \* ")

print(" \* \* ")

print(" \* \* ")

print(" \*\*\*\* ")

**19. Write a Python program to print the alphabet pattern 'E'.  
*Expected Output:***

**\*\*\*\*\***

**\***

**\***

**\*\*\*\***

**\***

**\***

**\*\*\*\*\***

print(" \*\*\*\*\* ")

print(" \* ")

print(" \* ")

print(" \*\*\*\* ")

print(" \* ")

print(" \* ")

print(" \*\*\*\*\* ")

**20. Write a Python program to print the alphabet pattern 'G'.  
*Expected Output:***

**\*\*\***

**\* \***

**\***

**\* \*\*\***

**\* \***

**\* \***

**\*\*\***

print(" \*\*\* ")

print(" \* \* ")

print(" \* ")

print(" \* \*\*\* ")

print(" \* \* ")

print(" \* \* ")

print(" \*\*\* ")