**LAB CYCLE 1**

**Experiment No : 1**

**Date :** 30/09/2024

**Aim :** Write a program that prompts the user to enter his first name and last name and then displays

a message “Greetings!!! First name Last name”.

**Pseudocode :**

1. Read the first name

2. Read the last name

3. Print("greetings!!!", fname, lname)

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| input() | Allows user input (Returns a string value) | input(prompt) |
| print() | Prints the specified message to the screen | print(object(s)) |

**Source Code :**

a=input("Enter your first name:")

b=input("Enter your last name:")

print("greetings!!!",a,b)

**Output :**

Enter your first name: John

Enter your last name: Doe

Greetings!!! John Doe

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 2**

**Date :** 30/09/2024

**Aim :** Write a program to demonstrate different number data types in python.

**Pseudocode :**

1. Set x = 2, y = 3.3, z = 3 + 2j

2. Print "Type of x =", type of x

3. Print "Type of y =", type of y

4. Print "Type of z =", type of z

5. Print "integer:", x, "float:", y, "complex number:", z

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| type() | Returns the type of the  specified object | type(object) |

**Source Code :**

x,y,z=2,3.3,3+2j

print("Type of x=",type(x))

print("Type of y=",type(y))

print("Type of z=",type(z))

print("integer:",x)

print("float:",y)

print("complex number:",z)

**Output :**

Type of x = <class 'int'>

Type of y = <class 'float'>

Type of z = <class 'complex'>

integer: 2

float: 3.3

complex number: (3+2j)

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 3**

**Date :** 30/09/2024

**Aim :** Write a program to calculate the area of a circle by reading inputs from the user.

**Pseudocode :**

1. Read r

2. Set area = 3.14 \* r \* r

3. Print "Area of the circle =", area

**Source Code :**

r = float(input("Enter the radius of the circle: "))

area = 3.14 \* r \* r

print("Area of the circle =", area)

**Output :**

Enter the radius of the circle: 5

Area of the circle = 78.5

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 4**

**Date :** 30/09/2024

**Aim :** Write a program to calculate the salary of an employee given his basic pay (to be entered by

the user) . HRA = 10 percent of the basic pay, TA = 5 percent of the basic pay.

**Pseudocode :**

1. Read BP

2. Set HRA = 0.10 \* BP

3. Set TA = 0.5 \* BP

4. Set salary = BP + HRA + TA

5. Print "Salary of the employee is:", salary

**Source Code :**

BP = float(input("Enter the basic pay of the employee: "))

HRA = 0.10 \* BP

TA = 0.5 \* BP

salary = BP + HRA + TA

print("Salary of the employee is:", salary)

**Output :**

Enter the basic pay of the employee: 25000

Salary of the employee is: 37500.0

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 5**

**Date :** 30/09/2024

**Aim :** Write a Python program to perform arithmetic operations on two integer numbers.

**Pseudocode :**

1. Read x

2. Read y

3. Print "Sum:", x, "+", y, "=", x + y

4. Print "Difference:", x, "-", y, "=", x - y

5. Print "Multiplication:", x, "\*", y, "=", x \* y

6. Print "Division:", x, "/", y, "=", x / y

7. Print "Remainder:", x, "%", y, "=", x % y

**Source Code :**

x = int(input("Enter the first number: "))

y = int(input("Enter the second number: "))

print("Sum:", x, "+", y, "=", x + y)

print("Difference:", x, "-", y, "=", x - y)

print("Multiplication:", x, "\*", y, "=", x \* y)

print("Division:", x, "/", y, "=", x / y)

print("Remainder:", x, "%", y, "=", x % y)

**Output :**

Enter the first number: 10

Enter the second number: 5

Sum: 10 + 5 = 15

Difference: 10 - 5 = 5

Multiplication: 10 \* 5 = 50

Division: 10 / 5 = 2.0

Remainder: 10 % 5 = 0

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 6**

**Date :** 30/09/2024

**Aim :** Write a Python program to get a string which is n (non-negative integer) copies of a given

string.

**Pseudocode :**

1. Read str

2. Read n

3. Print str \* n

**Source Code :**

str = input("Enter a string: ")

n = int(input("Enter the number of repetitions: "))

print(str \* n)

**Output :**

Enter a string: Hello

Enter the number of repetitions: 3

HelloHelloHello

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 7**

**Date :** 30/09/2024

**Aim :** Program to accept an integer n and compute n+nn+nnn.

[Hint : n = 5, then compute 5 + 55 + 555]

**Pseudocode :**

1. Read n

2. Print n, n\*2, n\*3

3. Print "Sum:", n + n\*2 + n\*3

**Source Code :**

n = input("Enter the value of n: ")

print(n, "+", n\*2, "+", n\*3)

sum = int(n) + int(n\*2) + int(n\*3)

print("Sum:", sum)

**Output :**

Enter the value of n: 5

5 + 55 + 555

Sum: 615

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 8**

**Date :** 30/09/2024

**Aim :** Find biggest of 3 numbers entered.

**Pseudocode :**

1. Read the 1st number (a)

2. Read the 2nd number (b)

3. Read the 3rd number (c)

4. Compare the numbers:

a. If a > b and a > c, largest = a

b. Else if b > a and b > c, largest = b

c. Else, largest = c

5. Print the largest number

**Source Code :**

a = int(input("Enter the 1st number: "))

b = int(input("Enter the 2nd number: "))

c = int(input("Enter the 3rd number: "))

if a > b and a > c:

largest = a

elif b > a and b > c:

largest = b

else:

largest = c

print("The largest number is:", largest)

**Output :**

Enter the 1st number: 12

Enter the 2nd number: 25

Enter the 3rd number: 8

The largest number is: 25

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 9**

**Date :** 30/09/2024

**Aim** : Program to determine whether a year is a leap year or not.

**Pseudocode :**

1. Read year

2. If year is divisible by 4:

a. If year is not divisible by 100 or year is divisible by 400:

Print "year is a leap year"

3. Else:

a. Print "year is not a leap year"

**Source Code :**

year = int(input("Enter a year: "))

if year % 4 == 0 and (year % 100 != 0 or year % 400 == 0):

print(year, "is a leap year")

else:

print(year, "is not a leap year")

**Output :**

Enter a year: 2024

2024 is a leap year

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 10**

**Date :** 30/09/2024

**Aim :** Write a Python program to determine the rate of entry-ticket in a trade fair

based on age as follows :

|  |  |
| --- | --- |
| **Age** | **Rate** |
| <10 | 7 |
| >10 and <60 | 10 |
| >=60 | 5 |

**Pseudocode :**

1. Read age

2. If age is 60 or greater:

a. Print "Ticket rate = 5"

3. Else if age is less than 60 but 10 or greater:

a. Print "Ticket rate = 10"

4. Else:

a. Print "Ticket rate = 7

**Source Code :**

age = int(input("Enter the age: "))

if age >= 60:

print("Ticket rate = 5")

elif age < 60 and age >= 10:

print("Ticket rate = 10")

else:

print("Ticket rate = 7")

**Output :**

Enter the age: 65

Ticket rate = 5

Enter the age: 25

Ticket rate = 10

Enter the age: 5

Ticket rate = 7

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 11**

**Date** : 30/09/2024

**Aim :** Write a Python program to solve a quadratic equation.

**Pseudocode :**

1. Read the coefficients a, b, and c

2. Calculate the discriminant (d) as d = b^2 - 4ac

3. If d < 0:

a. Calculate two complex solutions using the formula:

sol1 = (-b - sqrt(d)) / (2a)

sol2 = (-b + sqrt(d)) / (2a)

b. Print the complex solutions

4. If d == 0:

a. Calculate the single solution as x = -b / (2a)

b. Print the single solution

5. If d > 0:

a. Calculate two real solutions using the formula:

sol1 = (-b - sqrt(d)) / (2a)

sol2 = (-b + sqrt(d)) / (2a)

b. Print the real solutions

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| cmath.sqrt() | Computes the square root for real numbers | cmath.sqrt(value) |
| math.sqrt() | Displays the output to the user | math.sqrt(value) |

**Source Code :**

import math, cmath

print("Quadratic equation solver: ax^2 + bx + c = 0")

a = float(input("Enter coefficient of x^2 (a): "))

b = float(input("Enter coefficient of x (b): "))

c = float(input("Enter constant value (c): "))

d = (b\*\*2) - (4\*a\*c)

if d < 0:

sol1 = (-b - cmath.sqrt(d)) / (2 \* a)

sol2 = (-b + cmath.sqrt(d)) / (2 \* a)

print(f"The solution has complex solutions: {sol1}, {sol2}")

elif d == 0:

x = -b / (2 \* a)

print(f"Equation has one solution: {x}")

else:

sol1 = (-b - math.sqrt(d)) / (2 \* a)

sol2 = (-b + math.sqrt(d)) / (2 \* a)

print(f"Equation has two solutions: {sol1}, {sol2}")

**Output :**

Enter coefficient of x^2 (a): 1

Enter coefficient of x (b): -3

Enter constant value (c): 2

Equation has two solutions: 1.0, 2.0

**Result :** The program is successfully executed and the output is verified.

**LAB CYCLE 2**

**Experiment No : 1**

**Date** : 21/10/2024

**Aim :** Create a string from the given string where the first and last character are exchanged.

**Pseudocode :**

1. Read string

2. Set newstring = string[-1] + string[1:-1] + string[0]

3. Print newstring

**Source Code :**

string = input("Enter a string: ")

newstring = string[-1] + string[1:-1] + string[0]

print(newstring)

**Output :**

Enter a string: hello

oellh

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 2**

**Date :** 21/10/2024

**Aim :** Get a string from an input string where all occurrences of the first character are replaced with

‘$’, except the first character**.**

**Pseudocode :**

1. Read string

2. Set first\_char = string[0]

3. Set new\_string = first\_char + string[1:].replace(first\_char, "$")

4. Print "Modified string:", new\_string

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| replace() | Replaces all occurrences of a character with another | string.replace(old, new) |

**Source Code :**

string = input("Enter a string: ")

first\_char = string[0]

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

print("Modified string:", new\_string)

**Output :**

Enter a string: restart

Modified string: resta$t

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 3**

**Date :** 21/10/2024

**Aim :** Create a single string separated with space from two strings by swapping the character at

position 1.

**Pseudocode :**

1. Read string1

2. Read string2

3. Set swap\_str1 = string1[0] + string2[1] + string1[2:]

4. Set swap\_str2 = string2[0] + string1[1] + string2[2:]

5. Set string3 = swap\_str1 + " " + swap\_str2

6. Print string3

**Source Code :**

string1 = input("Enter your string 1: ")

string2 = input("Enter your string 2: ")

swap\_str1 = string1[0] + string2[1] + string1[2:]

swap\_str2 = string2[0] + string1[1] + string2[2:]

string3 = swap\_str1 + " " + swap\_str2

print(string3)

**Output :**

Enter your string 1: hello

Enter your string 2: world

hallo werld

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 4**

**Date :** 21/10/2024

**Aim :** Count the number of characters (character frequency) in a string.

**Pseudocode :**

1. Read n and convert it to lowercase

2. Initialize an empty dictionary s

3. For each character i in n:

a. If i is in s, increment s[i] by 1

b. Else, set s[i] = 1

6. Print s

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| lower() | Converts all characters in a string to lowercase | string.lower() |

**Source Code :**

n = input("Enter the string: ").lower()

s = {}

for i in n:

if i in s:

s[i] += 1

else:

s[i] = 1

print(s)

**Output :**

Enter the string: Hello

{'h': 1, 'e': 1, 'l': 2, 'o': 1}

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 5**

**Date :** 21/10/2024

**Aim :** Add ‘ing’ at the end of a given string. If it already ends with ‘ing’, then add

‘ly’.

**Pseudocode :**

1. Read s

2. If s ends with "ing":

Print s + "ly"

4. Else:

Print s + "ing"

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| endswith() | Checks if the string ends with a specific substring | string.endswith("substring") |

**Source Code :**

s = input("Enter a string: ")

if s.endswith("ing"):

print(s + "ly")

else:

print(s + "ing")

**Output :**

Enter a string: playing

playingly

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 6**

**Date :** 21/10/2024

**Aim :** Store a list of first names. Count the occurrences of ‘a’ within the list.

**Pseudocode :**

1. Initialize an empty list names

2. Read n (limit)

3. For i from 0 to n-1:

a. Read el (name)

b. Append el to names

4. Print "Occurrences of 'a' in names:"

5. For each name in names:

a. Print name and count of 'a' in name (convert name to lowercase and count 'a')

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| append() | Adds an item to the end of the list | list.append(item) |
| lower() | Converts a string to lowercase | string.lower() |
| count() | Counts occurrences of a substring | string.count('substring') |

**Source Code :**

names = []

n = int(input("Enter limit: "))

for i in range(n):

el = input(f"Enter name{i+1}: ")

names.append(el)

print("Occurrences of ‘a’ in names:")

for name in names:

print(f"{name}: {name.lower().count('a')}")

**Output :**

Enter limit: 3

Enter name1: Alice

Enter name2: Bob

Enter name3: Anna

Occurrences of ‘a’ in names:

Alice: 1

Bob: 0

Anna: 2

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 7**

**Date :** 21/10/2024

**Aim :** Write a python program to read two lists color-list1 and color-list2. Print out all colors from

color-list1 not contained in color-list2.

**Pseudocode :**

1. Read lst1

2. Read lst2

3. Convert lst1 and lst2 to sets

4. Print set(lst1) - set(lst2)

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| split() | Splits a string into a list | string.split() |
| set() | Converts a list to a set | set(list) |
| list() | Converts a set back to a list | list(set) |

**Source Code :**

lst1 = input("Enter list1 elements by space separated: ").split()

lst2 = input("Enter list2 elements by space separated: ").split()

print("Output list:")

print(list(set(lst1) - set(lst2)))

**Output :**

Enter list1 elements by space separated: apple banana cherry

Enter list2 elements by space separated: banana orange

Output list: ['apple', 'cherry']

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 8**

**Date :** 21/10/2024

**Aim :** Create a list of colors from comma-separated color names entered by the user. Display first

and last colors.

**Pseudocode :**

1. Read colors as a comma-separated string

2. Split the string into a list of colors

3. Print the list of colors

4. Print the first color (colors[0])

5. Print the last color (colors[-1])

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| split() | Splits a string into a list based on a separator | string.split(",") |

**Source Code :**

colors = input("Enter colors (comma separated): ").split(",")

colors = [color for color in colors]

print(colors)

print("First color:", colors[0])

print("Last color:", colors[-1])

**Output :**

Enter colors (comma separated): red,green,blue,yellow

['red', 'green', 'blue', 'yellow']

First color: red

Last color: yellow

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 9**

**Date :** 21/10/2024

**Aim :** Write a program to prompt the user for a list of integers. For all values greater than 100,store

‘over’ instead.

**Pseudocode :**

1. Read input and split it into a list of integers

2. For each element in the list:

If element > 100, replace it with "over"

4. Print the modified list

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| split() | Splits a string into a list based on a separator | string.split() |

**Source Code :**

lst = [int(num) for num in input("Enter list elements (space separated): ").split()]

for i in range(len(lst)):

if lst[i] > 100:

lst[i] = "over"

print(lst)

**Output :**

Enter list elements (space separated): 50 120 85 200 75

[50, 'over', 85, 'over', 75]

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 10**

**Date :** 21/10/2024

**Aim** : From a list of integers, create a list after removing even numbers.

**Pseudocode :**

1. Read input and split it into a list of integers

2. Create a new list odd\_lst containing only odd numbers from lst

3. Print odd\_lst

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| split() | Splits a string into a list based on a separator | string.split() |

**Source Code :**

lst = [int(num) for num in input("Enter a list of numbers (space separated): ").split()]

odd\_lst = [odd for odd in lst if odd % 2 != 0]

print(odd\_lst)

**Output :**

Enter a list of numbers (space separated): 10 15 20 25 30

[15, 25]

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 11**

**Date :** 21/10/2024

**Aim** : Accept a list of words and return the length of the longest word.

**Pseudocode :**

1. Read input and split it into a list of words

2. Find the maximum length of words in the list

3. Create a new list lg\_word containing all words with the maximum length

4. Print the largest word(s) and their size

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| split() | Splits a string into a list based on a separator | string.split() |
| max() | Finds the maximum value in a list | max(list) |
| len() | Returns the length of an object | len(object) |

**Source Code :**

lst = input("Enter a list of words (space separated): ").split()

max\_length = max(len(word) for word in lst)

lg\_word = [word for word in lst if len(word) == max\_length]

print(f"Largest word(s): {lg\_word}, size: {max\_length}")

**Output :**

Enter a list of words (space separated): apple banana orange pineapple

Largest word(s): ['pineapple'], size: 9

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 12**

**Date :** 21/10/2024

**Aim :** Write a program to prompt the user to enter two lists of integers and check

(a) Whether lists are of the same length.

(b) Whether the list sums to the same value.

(c) Whether any value occurs in both Lists.

**Pseudocode :**

1. Read lst1 as a list of space-separated integers

2. Read lst2 as a list of space-separated integers

3. Set length = (length of lst1 == length of lst2)

4. Set lsum = (sum of lst1 == sum of lst2)

5. Set common = intersection of lst1 and lst2

6. If length:

Print "Lists lengths are the same"

Else:

Print "Lists lengths are not the same"

9. Print "Lists common elements:", common

10. If lsum:

Print "List sums are the same"

Else:

Print "List sums are not the same"

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| split() | Splits a string into a list based on a separator | string.split() |
| len() | Returns the length of an object | len(object) |
| sum() | Returns the sum of all elements in a list | sum(list) |
| set() | Converts a list to a set | set(list) set(list) |

**Source Code :**

lst1 = [int(num) for num in input("Enter first list (space separated): ").split()]

lst2 = [int(num) for num in input("Enter second list (space separated): ").split()]

length = len(lst1) == len(lst2)

lsum = sum(lst1) == sum(lst2)

common = set(lst1) & set(lst2)

if length:

print("Lists lengths are the same")

else:

print("Lists lengths are not the same")

print(f"Lists common elements: {common}")

if lsum:

print("List sums are the same")

else:

print("List sums are not the same")

**Output :**

Enter first list (space separated): 1 2 3

Enter second list (space separated): 4 5 6

Lists lengths are the same

Lists common elements: set()

List sums are not the same

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 13**

**Date :** 21/10/2024

**Aim :** Write a Python program to count the occurrences of each word in a line of text.

Hint: use split() function and dictionary

**Pseudocode :**

1. Read a string, convert it to lowercase, and split it into words (sentence)

2. Initialize an empty dictionary freq\_dict

3. For each word in sentence:

4. If word exists in freq\_dict:

a. Increment freq\_dict[word] by 1

6. Else:

a. Set freq\_dict[word] = 1

8. Print "Word occurrence:"

9. For each key, value in freq\_dict:

a. Print key and value

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| lower() | Converts a string to lowercase | string.lower() |
| split() | Splits a string into a list based on a separator | string.split() |
| dict() | Creates an empty dictionary | dict() |
| items() | Returns the key-value pairs of a dictionary | dict.items() |

**Source Code :**

sentence = [word for word in input("Enter a string: ").lower().split()]

freq\_dict = {}

for word in sentence:

if word in freq\_dict:

freq\_dict[word] += 1

else:

freq\_dict[word] = 1

print("Word occurrence:")

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

print(f"{key}: {value}")

**Output :**

Enter a string: Hello hello world world world

Word occurrence:

hello: 2

world: 3

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 14**

**Date :** 21/10/2024

**Aim :** List comprehensions:

(a) Generate positive list of numbers from a given list of integers

(b) Square of N numbers

(c) Form a list of vowels selected from a given word

(d) Form a list ordinal value of each element of a word (Hint: use ord() to get ordinal

values)

**Pseudocode :**

1. Initialize numbers and find positive\_numbers from numbers

2. Print positive\_numbers

3. Set N = 6 and calculate squares of numbers from 1 to N

4. Print squares

5. Set word = "comprehension" and find vowels in word

6. Print vowels

7. Set word = "hello" and find ordinal\_values of characters in word

8. Print ordinal\_values

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| ord() | Returns the ASCII value of a character | ord(char) |
| range() | Generates a range of numbers | range(start, stop) |

**Source Code :**

numbers = [-10, 15, 5, 7, -26, 18, 0]

positive\_numbers = [num for num in numbers if num > 0]

print(f"Positive numbers in {numbers}:", positive\_numbers)

N = 6

squares = [num \*\* 2 for num in range(1, N + 1)]

print("Squares of first 6 numbers:", squares)

word = "comprehension"

vowels = [char for char in word if char in 'aeiou']

print(f"Vowels in the word: {word}", vowels)

word = "hello"

ordinal\_values = [ord(char) for char in word]

print(f"Ordinal values in the word: {word}", ordinal\_values)

**Output :**

Positive numbers in [-10, 15, 5, 7, -26, 18, 0]: [15, 5, 7, 18]

Squares of first 6 numbers: [1, 4, 9, 16, 25, 36]

Vowels in the word: comprehension ['o', 'e', 'e', 'i', 'o']

Ordinal values in the word: hello [104, 101, 108, 108, 111]

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 15**

**Date :** 21/10/2024

**Aim :** Sort dictionary in ascending and descending order.

**Pseudocode :**

1. Initialize my\_dict with key-value pairs

2. Sort my\_dict by keys in ascending order and store in keys\_asc

3. Print "Sorted by keys (ascending):", keys\_asc

4. Sort my\_dict by keys in descending order and store in keys\_desc

5. Print "Sorted by keys (descending):", keys\_desc

6. Sort my\_dict by values in ascending order and store in values\_asc

7. Print "Sorted by values (ascending):", values\_asc

8. Sort my\_dict by values in descending order and store in values\_desc

9. Print "Sorted by values (descending):", values\_desc

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| |  | | --- | | sorted() |  |  | | --- | |  | | Sorts the elements of an iterable | sorted(iterable, key=key\_func, reverse=bool) |
| |  | | --- | | items() |  |  | | --- | |  | | Returns key-value pairs of a dictionary as tuples | dict.items() |
| lambda | Creates an anonymous function | lambda arguments: expression |
| dict() | Creates a dictionary from a sequence of key-value pairs | dict(sequence) |

**Source Code :**

my\_dict = {'apple': 6, 'banana': 3, 'kiwi': 4, 'orange': 10}

keys\_asc = dict(sorted(my\_dict.items()))

print("Sorted by keys (ascending):", keys\_asc)

keys\_desc = dict(sorted(my\_dict.items(), reverse=True))

print("Sorted by keys (descending):", keys\_desc)

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

print("Sorted by values (ascending):", values\_asc)

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

print("Sorted by values (descending):", values\_desc)

**Output :**

Sorted by keys (ascending): {'apple': 6, 'banana': 3, 'kiwi': 4, 'orange': 10}

Sorted by keys (descending): {'orange': 10, 'kiwi': 4, 'banana': 3, 'apple': 6}

Sorted by values (ascending): {'banana': 3, 'kiwi': 4, 'apple': 6, 'orange': 10}

Sorted by values (descending): {'orange': 10, 'apple': 6, 'kiwi': 4, 'banana': 3}

**Result :** The program is successfully executed and the output is verified.

**Experiment No : 16**

**Date :** 21/10/2024

**Aim :** Merge two dictionaries**.**

**Pseudocode :**

1. Initialize dict1 with key-value pairs

2. Initialize dict2 with key-value pairs

3. Print dict1

4. Print dict2

5. Merge dict2 into dict1 using the update() method

6. Print "Merged:", dict1

**Method :**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Syntax** |
| update() | Merges two dictionaries. If keys are the same, the values from the second dictionary overwrite those in the first. | dict1.update(dict2) |

**Source Code :**

dict1 = {'banana': 3, 'mango': 5}

dict2 = {'orange': 2, 'pineapple': 4}

print(dict1)

print(dict2)

dict1.update(dict2)

print(f"Merged: {dict1}")

**Output :**

{'banana': 3, 'mango': 5}

{'orange': 2, 'pineapple': 4}

Merged: {'banana': 3, 'mango': 5, 'orange': 2, 'pineapple': 4}

**Result :** The program is successfully executed and the output is verified.