Python Lab-Internal Answers

1. Write a Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number as an argument

**code**

def factorial(n):

if n == 0 or n == 1:

return 1

else:

return n \* factorial(n - 1)

# Test the function

num = int(input("Enter a non-negative integer: "))

if num < 0:

print("Factorial is not defined for negative numbers.")

else:

result = factorial(num)

print(f"The factorial of {num} is {result}")

**Output:**

Enter a non-negative integer: 5

The factorial of 5 is 120

**Description:**

You can run this code and enter a non-negative integer to calculate its factorial. The function uses recursion to calculate the factorial by repeatedly multiplying the number by the factorial of its predecessors until it reaches the base case of 0 or 1.

1. Write a Python function that accepts a string and counts the number of upper and lower case letters.

**Code:**

def count\_upper\_lower(string):

upper\_count = sum(1 for char in string if char.isupper())

lower\_count = sum(1 for char in string if char.islower())

return upper\_count, lower\_count

input\_string = 'The quick Brow Fox'

upper, lower = count\_upper\_lower(input\_string)

print(f"Number of uppercase letters: {upper}")

print(f"Number of lowercase letters: {lower}")

**Output:**

Number of uppercase letters: 3

Number of lowercase letters: 12

**Description:**

In this version, I've used list comprehensions to calculate the counts of uppercase and lowercase letters, making the code more concise while still providing the same functionality and output.

1. Write a Python GUI program to import Tkinter package and create a window and set its title

**code**

import tkinter as tk

# Create the main window

root = tk.Tk()

# Set the title of the window

root.title("My GUI Window")

# Start the GUI event loop

root.mainloop())

**Description:**

When you run this code, it will open a GUI window with the title "My GUI Window". The window will remain open until you close it manually.

1. Write a Python GUI program to add a button in your application using tkinter module.

**code**

import tkinter as tk

# Function to be executed when the button is clicked

def button\_click():

label.config(text="Button Clicked!")

# Create the main window

root = tk.Tk()

# Create a button

button = tk.Button(root, text="Click Me", command=button\_click)

button.pack()

# Create a label to display the message

label = tk.Label(root, text="")

label.pack()

# Set the title of the window

root.title("Button Example")

# Start the GUI event loop

**Description:**

When you run this code, it will open a GUI window with a button labeled "Click Me". When you click the button, the label below the button will change to "Button Clicked!". The window will remain open until you close it manually.

1. **A)** Write a NumPy program to print the NumPy version on your system

**Code:**

import numpy as np

# Print the NumPy version

print("NumPy version:", np.\_\_version\_\_)

**Output:**

NumPy version: 1.21.2

**Description:**

Note that the version number might be different based on your installed version of NumPy.

B) Write a NumPy program to create a 3x3 matrix with values ranging from 2 to 10.  
 **Code:**

import numpy as np

# Create a 3x3 matrix with values ranging from 2 to 10

matrix = np.arange(2, 11).reshape(3, 3)

# Print the matrix

print(matrix)

**Output:**

[[ 2 3 4]

[ 5 6 7]

[ 8 9 10]]

**Description:**

Importing NumPy: The program starts by importing the NumPy library with the alias np.

Creating the Matrix: The code uses the np.arange() function to generate an array of values ranging from 2 to 10 (inclusive). The reshape() function is then applied to this array to reshape it into a 3x3 matrix. This matrix is stored in the variable named matrix.

Printing the Matrix: The program uses the print() function to display the contents of the created matrix in the console

1. Write a NumPy program to convert an array to a floating type.

**Code:**

import numpy as np

# Create an integer array

integer\_array = np.array([1, 2, 3, 4, 5])

# Convert the array to a floating-point type

float\_array = integer\_array.astype(float)

# Print the original and converted arrays

print("Original array:", integer\_array)

print("Float array:", float\_array)

**Output:**

Original array: [1 2 3 4 5]

Float array: [1. 2. 3. 4. 5.]

**Description:**

In this code, the astype() function is used to convert the data type of the array elements to float.

1. Write a Python program to create a calculator class. Include methods for basic arithmetic operations

**Code:**

class Calculator:

def add(self, a, b):

return a + b

def subtract(self, a, b):

return a - b

def multiply(self, a, b):

return a \* b

def divide(self, a, b):

if b != 0:

return a / b

else:

return "Cannot divide by zero"

# Create an instance of the Calculator class

calc = Calculator()

# Perform arithmetic operations

print("Addition:", calc.add(5, 3))

print("Subtraction:", calc.subtract(10, 4))

print("Multiplication:", calc.multiply(6, 2))

print("Division:", calc.divide(15, 3))

print("Division:", calc.divide(10, 0))

**Output:**

Addition: 8

Subtraction: 6

Multiplication: 12

Division: 5.0

Division: Cannot divide by zero

**Description:**

This program defines a Calculator class with methods for addition, subtraction, multiplication, and division. It then creates an instance of the class and demonstrates the use of the methods to perform calculations.

1. Python Program to find largest element in an array

**Code:**

def find\_largest(arr):

largest = arr[0]

for num in arr:

if num > largest:

largest = num

return largest

# Test the function

array = [15, 3, 27, 8, 10, 35]

largest\_element = find\_largest(array)

print("Largest element:", largest\_element)

**Output:**

Largest element: 35

**Description:**

The ‘find\_largest’ function iterates through the array and keeps track of the largest element encountered. After iterating through the array, it returns the largest element found.

1. Python Program to find yesterday’s, today’s and tomorrow’s date

**Code:**

from datetime import datetime, timedelta

# Get today's date

today = datetime.now()

# Calculate yesterday's date

yesterday = today - timedelta(days=1)

# Calculate tomorrow's date

tomorrow = today + timedelta(days=1)

# Print the dates

print("Yesterday's date:", yesterday.strftime("%Y-%m-%d"))

print("Today's date:", today.strftime("%Y-%m-%d"))

print("Tomorrow's date:", tomorrow.strftime("%Y-%m-%d"))

**Output:**

Yesterday's date: 2023-08-21

Today's date: 2023-08-22

Tomorrow's date: 2023-08-23

**Description:**

The program uses the datetime module to manipulate dates and the timedelta class to perform date calculations. It then prints the formatted dates for yesterday, today, and tomorrow.

1. Python Program to find difference between current time and given time

**Code:**

from datetime import datetime

# Current time

current\_time = datetime.now().time()

# Given time (replace with your desired time)

given\_time = datetime.strptime("15:30:00", "%H:%M:%S").time()

# Calculate the time difference

time\_difference = datetime.combine(datetime.today(), current\_time) - datetime.combine(datetime.today(), given\_time)

# Print the time difference

print("Time difference:", time\_difference)

**Output:**

Time difference: 7:51:32.724287

**Description:**

In this example, the program calculates the time difference between the current time and the given time (15:30:00). You can replace the given\_time with your desired time in the format "HH:MM:SS".

1. Python program to Count Uppercase, Lowercase, special character and numeric values using Regex

**Code:**

import re

def count\_characters(input\_string):

uppercase\_count = len(re.findall(r'[A-Z]', input\_string))

lowercase\_count = len(re.findall(r'[a-z]', input\_string))

special\_count = len(re.findall(r'[^a-zA-Z0-9\s]', input\_string))

numeric\_count = len(re.findall(r'\d', input\_string))

return uppercase\_count, lowercase\_count, special\_count, numeric\_count

# Test the function

string = "Hello World! 123"

upper, lower, special, numeric = count\_characters(string)

print("Uppercase letters:", upper)

print("Lowercase letters:", lower)

print("Special characters:", special)

print("Numeric values:", numeric)

**Output:**

Uppercase letters: 2

Lowercase letters: 8

Special characters: 2

Numeric values: 3

**Description:**

This program uses regular expressions (re module) to find and count specific patterns in the input string.

1. Count number of lines in a text file in Python

**Code:**

def count\_lines(file\_name):

with open(file\_name, 'r') as file:

lines = file.readlines()

return len(lines)

# Test the function

file\_name = 'sample.txt' # Replace with your file's name

line\_count = count\_lines(file\_name)

print("Number of lines:", line\_count)

**Output:**

Number of lines: 5

**Description:**

Replace 'sample.txt' with the name of the text file you want to count lines for.

Ensure that the specified file exists in the same directory as the script or provide the correct file path if it's located elsewhere.

24. Write a Python function to multiply all the numbers in a list

**Code:**

def multiply\_list(numbers):

result = 1

for num in numbers:

result \*= num

return result

# Test the function

numbers\_list = [2, 3, 4, 5]

result = multiply\_list(numbers\_list)

print("Multiplication result:", result)

**Output:**

Multiplication result: 120

**Description:**

The function iterates through the numbers in the list and keeps updating the result by multiplying it with each number.

26. Write a python program to perform different set operations

**Code:**

# Create two sets

set1 = {1, 2, 3, 4, 5}

set2 = {4, 5, 6, 7, 8}

# Perform set operations

union\_set = set1 | set2

intersection\_set = set1 & set2

difference\_set = set1 - set2

symmetric\_difference\_set = set1 ^ set2

# Print the results

print("Set 1:", set1)

print("Set 2:", set2)

print("Union:", union\_set)

print("Intersection:", intersection\_set)

print("Difference:", difference\_set)

print("Symmetric Difference:", symmetric\_difference\_set)

**Output:**

Set 1: {1, 2, 3, 4, 5}

Set 2: {4, 5, 6, 7, 8}

Union: {1, 2, 3, 4, 5, 6, 7, 8}

Intersection: {4, 5}

Difference: {1, 2, 3}

Symmetric Difference: {1, 2, 3, 6, 7, 8}

**Description:**

This program creates two sets, performs union, intersection, difference, and symmetric difference operations, and then prints the results.

27. Write a Python program to Make a Simple Calculator

**Code:**

def add(x, y):

return x + y

def subtract(x, y):

return x - y

def multiply(x, y):

return x \* y

def divide(x, y):

if y != 0:

return x / y

else:

return "Cannot divide by zero"

print("Select operation:")

print("1. Add")

print("2. Subtract")

print("3. Multiply")

print("4. Divide")

choice = input("Enter choice (1/2/3/4): ")

num1 = float(input("Enter first number: "))

num2 = float(input("Enter second number: "))

if choice == '1':

print("Result:", add(num1, num2))

elif choice == '2':

print("Result:", subtract(num1, num2))

elif choice == '3':

print("Result:", multiply(num1, num2))

elif choice == '4':

print("Result:", divide(num1, num2))

else:

print("Invalid choice")

**Output:**

Select operation:

1. Add

2. Subtract

3. Multiply

4. Divide

Enter choice (1/2/3/4): 1

Enter first number: 5

Enter second number: 3

Result: 8.0

**Description:**

This Python program creates a simple calculator that performs basic arithmetic operations. It prompts the user to choose an operation (add, subtract, multiply, or divide), inputs two numbers, and then displays the result of the selected operation.