(Approved by AICTE and Affiliated to KTU)

Mampara, Pazhoor P.O, Kuttippuram, Malappuram

kitm@kmct.org | kmctitm.org

LAB MANUAL

PROGRAMME & BRANCH	**B.TECH. ARTIFICIAL INTELLIGENCE & DATA SCIENCE,** **B.TECH.COMPUTER SCIENCE AND ENGINEERING**
SEMESTER & YEAR	**1, 2024-25**
COURSE	**ALGORITHM THINKING WITH PYTHON**
CODE	**UCEST105**

EXPERIMENT 1

SIMPLE CALCULATOR

AIM

Simple desktop calculator using Python. *Only the five basic arithmetic operators*.

ALGORITHM

1. **Step 1: Input**
   a. Prompt the user to input a number and store it in num1 as a floating-point number.
   b. Prompt the user to input a second number and store it in num2 as a floating-point number.
2. **Step 2: Addition**
   a. Calculate the sum of num1 and num2 and store the result in addition.
   b. Display the result: "The result of adding num1 and num2 is addition."
3. **Step 3: Subtraction**
   a. Calculate the difference between num1 and num2 and store the result in subtraction.
   b. Display the result: "The result of subtracting num1 and num2 is subtraction."
4. **Step 4: Multiplication**
   a. Calculate the product of num1 and num2 and store the result in product.
   b. Display the result: "The result of multiplying num1 and num2 is product."
5. **Step 5: Division**
   a. Check if num2 is not equal to zero:
   b. If num2 is not zero, calculate the division of num1 by num2 and store the result in division.
   c. Display the result: "The result of dividing num1 and num2 is division."
   d. If num2 is zero, display: "Division by zero is undefined."
6. **Step 6: Modulus**
   a. Check if num2 is not equal to zero:

   b. If num2 is not zero, calculate the modulus of num1 and num2 and store the result in modulus.
   c. Display the result: "The result of modulus operation with num1 and num2 is modulus."
   d. If num2 is zero, display: "Modulus by zero is undefined."

7. **Step 7: Exponentiation**
   a. Calculate num1 raised to the power of num2 and store the result in exponentiation.
   b. Display the result: "The result of exponentiation operation with num1 and num2 is exponentiation."
8. **Step 8: Floor Division**
   a. Check if num2 is not equal to zero:
      a. If num2 is not zero, calculate the floor division of num1 by num2 and store the result in floor_division.
      b. Display the result: "The result of floor division operation with num1 and num2 is floor_division."
      c. If num2 is zero, display: "Floor division by zero is undefined."

PROGRAM

```
num1 = float(input("Enter a number: "))
num2 = float(input("Enter 2nd number: "))
# Addition
addition = num1 + num2
print("The result of adding", num1, "and", num2, "is", addition)
# Subtraction
subtraction = num1 - num2
print("The result of subtracting", num1, "and", num2, "is", subtraction)
# Multiplication
product = num1 * num2
print("The result of multiplying", num1, "and", num2, "is", product)
# Division
if num2 != 0:
    division = num1 / num2
    print("The result of dividing", num1, "and", num2, "is", division)
```

```
else:
    print("Division by zero is undefined.")

# Modulus
if num2 != 0:
    modulus = num1 % num2
    print("The result of modulus operation with", num1, "and", num2, "is", modulus)
else:
    print("Modulus by zero is undefined.")
# Exponentiation
exponentiation = num1 ** num2
print("The result of exponentiation operation with", num1, "and", num2, "is", exponentiation)
# Floor Division
if num2 != 0:
    floor_division = num1 // num2
    print("The result of floor division operation with", num1, "and", num2, "is", floor_division)
else:
    print("Floor division by zero is undefined.")
```

SAMPLE OUTPUT

```
Enter a number3
Enter 2nd number2
The result of adding  3 and  2 is 5
The result of subtracting  3 and  2 is 1
The result of multiplying  3 and  2 is 6
The result of dividing  3 and  2 is 1.5
The result of modulus operation with  3 and  2 is 1
The result of exponentiation operation with  3 and  2 is 9
The result of floor division operation with  3 and  2 is 1
```

EXPERIMENT 2

STRING OPERATIONS

AIM

Create, concatenate, and print a string and access a sub-string from a given string.

ALGORITHM

Step 1:Input the first string and store it in str1.

Step 2:Display the message "String created".

Step 3:Input the second string and store it in str2.

Step 4:Concatenate str1 and str2 with a space in between and store the result in str3.

Step 5:Display the concatenated string str3.

Step 6:Input the word position pos as an integer.

Step 7:Extract the substring starting from position pos + 1 to the end of str3 and store it in sub_string.

Step 8:Display the substring sub_string.

PROGRAM

```
str1=input("Enter a string")

print("String created")

str2=input("Enter a string to concatenate")

str3=str1+" "+str2

print("The concatenate String is",str3)

pos=int(input("Enter the number of word position: "))

sub_string=str3[pos+1:]

print("Substring: ",sub_string)
```

SAMPLE OUTPUT

Enter a stringBritannia

String created

Enter a string to concatenateBiscuit

The concatenate String is Britannia Biscuit

Enter the number of word position: 3

Substring: annia Biscuit

EXPERIMENT 3

DATE AND TIME FORMATS

AIM

Familiarize time and date in various formats (Eg. “Thu Jul 11 10:26:23 IST 2024”).

(a) Display current date and time

(b) Display the format [YYYY-MM-DD HH:MM:SS]

(c) Display the format [MM/DD/YYYY]

(d) Display the format [Day, Month DD, YYYY]

(e) Display the format [Day, Month DD, YYYY HH:MM:SS AM/PM]

(f) Format the date and time like "Thu-Jul-11 10:26:23 IST 2024"

(g) Display [Abbr Weekday Name-Abbr month name-DD HH:MM:SS IST YYYY]

Eg: Wed-Oct-02 12:41:18 IST 2024

(h) Display format- 8 [ISO format:]

(i) Display Date only

(j) Display Time only

(k) Display month only

(l) Display Year only

THEORY

-To work with time and date in various formats in Python, you can use the datetime module.

-This module allows you to format dates and times in different ways using strftime() and parse them using strptime().

Commonly Used Date and Time Format Codes

- %a: Abbreviated weekday name (e.g., Mon, Tue)
- %A: Full weekday name (e.g., Monday, Tuesday)
- %b: Abbreviated month name (e.g., Jan, Feb)
- %B: Full month name (e.g., January, February)
- %d: Day of the month (zero-padded) (e.g., 01, 02, 31)
- %m: Month as a zero-padded decimal number (e.g., 01, 02, 12)

- %Y: Year with century (e.g., 2024)
- %H: Hour (24-hour clock) (e.g., 00, 23)
- %I: Hour (12-hour clock) (e.g., 01, 12)
- %M: Minute as a zero-padded decimal number (e.g., 00, 59)
- %S: Second as a zero-padded decimal number (e.g., 00, 59)
- %p: AM or PM
- %Z: Time zone name (e.g., IST, UTC)

PROGRAM

```
from datetime import datetime
# The datetime module in Python allows you to work with dates and times. Here's how you can use it to get the current date and time.
#(a)     Display current date and time
current=datetime.now() #datetime.now() returns the current local date and time
print("Current date and time",current)
#(b)     Display the format  [YYYY-MM-DD HH:MM:SS]
format_1=current.strftime("%Y-%m-%d %H:%M:%S")
print("format -1 [YYYY-MM-DD HH:MM:SS]::--->",format_1)
#(c)     Display the format  [MM/DD/YYYY]
format_2=current.strftime("%m/%d/%Y")
print("format -2 [MM/DD/YYYY]::--->",format_2)
#(d)     Display the format  [Day, Month DD, YYYY]
format_3=current.strftime("%A,%B %m, %Y")
print("format -3 [Day, Month DD, YYYY]::--->",format_3)
#(e)     Display the format  [Day, Month DD, YYYY HH:MM:SS AM/PM]
format_4=current.strftime("%A,%B %d, %Y %I:%M:%S %p")
print("format -4 [Day, Month DD, YYYY HH:MM:SS AM/PM]::--->",format_4)
#(f)     Format the date and time like "Thu-Jul-11 10:26:23 IST 2024"
format_7=current.strftime("%a-%b-%d %H:%M:%S IST %Y")
```

```
#(g)    Display [Abbr Weekday Name-Abbr month name-DD HH:MM:SS IST YYYY]
print("format -7 [Abbr Weekday Name-Abbr month name-DD HH:MM:SS IST YYYY]::--->",format_7)
#(h)    Display isoformat
format_8 = current.isoformat()
print("format- 8 [ISO format:]::---> ",format_8)
#(i)    Display date only
print("format -9 [Date Only]::--->",current.date())
#(j)    Display time only
print("format -10 [Time Only]::--->",current.time())
#(k)    Display time only
print("Current month",current.month)
#(l)    Display time only
print("Current year",current.year)
```

SAMPLE OUTPUT:

```
Current date and time 2024-10-02 12:41:18.089058
format -1 [YYYY-MM-DD HH:MM:SS]::---> 2024-10-02 12:41:18
format -2 [MM/DD/YYYY]::---> 10/02/2024
format -3 [Day, Month DD, YYYY]::---> Wednesday,October 10, 2024
format -4 [Day, Month DD, YYYY HH:MM:SS AM/PM]::---> Wednesday,October 02, 2024 12:41:18 PM
format -7 [Abbr Weekday Name-Abbr month name-DD HH:MM:SS IST YYYY]::---> Wed-Oct-02 12:41:18 IST 2024
format- 8 [ISO format:]::--->  2024-10-02T12:41:18.089058
format -9 [Date Only]::---> 2024-10-02
format -10 [Time Only]::---> 12:41:18.089058
Current month 10
Current year 2024
```

EXPERIMENT 4

LIST USING NUMPY

AIM

Write a program to create, append, and remove lists in Python using NumPy.

ALGORITHM

Step 1 - Create a NumPy array:

Use np.array() to create an array from a list of values.

Step 2 - Append a single element:

Use np.append() to add a new element to the end of the array. This creates a new array, so we assign it back to arr.

Step 3 - Append multiple elements:

Append a list of elements into the array using the function np.append() by passing a list.

Step 4 - Remove an element by index:

Use np.delete() to remove an element at a specific index.

PROGRAM

```
import numpy as np

# 1. Create a NumPy array (which can be treated like a list)
array = np.array([321,1234,123,52,234])
print("Original Array:", array)

# 2. Append elements to the array
# NumPy doesn't support in-place appending, so we use np.append() to create a new array
new_array = np.append(array, [6, 7])
print("Array after append:", new_array)

# 3. Remove an element from the array
# You can remove elements by selecting the indices you want to keep
# Let's remove the element at index 2 (the number '3')
```

```
new_array = np.delete(new_array, 2)
print("Array after removal:", new_array)

# 4. Append more elements to the updated array
new_array = np.append(new_array, [8, 9])
print("Array after second append:", new_array)

# 5. Remove a specific element (e.g., remove number '52')
new_array = np.delete(new_array, np.where(new_array == 52))
print("Array after removing number 52:", new_array)
```

SAMPLE OUTPUT

```
Original Array: [ 321 1234  123   52  234]
Array after append: [ 321 1234  123   52  234    6    7]
Array after removal: [ 321 1234   52  234    6    7]
Array after second append: [ 321 1234   52  234    6    7    8    9]
Array after removing number 52: [ 321 1234  234    6    7    8    9]
```

EXPERIMENT 5
LARGEST OF THREE NUMBERS

AIM

Program to find the largest of three numbers.

ALGORITHM

Step 1: Start.

Step 2: Input three numbers, num1, num2, and num3.

Step 3: Compare the numbers:

Step 3.1: If num1 is greater than or equal to num2 and num1 is greater than or equal to num3, then num1 is the largest.

Step 3.2: Else if num2 is greater than or equal to num1 and num2 is greater than or equal to num3, then num2 is the largest.

Step 3.3: Else, num3 is the largest.

Step 4: Output the largest number.

Step 5: End.

PROGRAM

```
num1=int(input("Enter 1st number "))
num2=int(input("Enter 2nd number "))
num3=int(input("Enter 3rd number "))
if (num1>=num2) and (num1>=num3):
   print(num1,"is the greatest")
elif (num2>=num1) and (num2>=num3):
   print(num2,"is the greatest")
else:
```

```
    print(num3,"is the greatest")
```

SAMPLE OUTPUT

```
Enter 1st number 6764

Enter 2nd number 123

Enter 3rd number 57

6764 is the greatest
```

EXPERIMENT 6

TEMPERATURE CONVERSION

AIM

Convert temperature values back and forth between Celsius (c), and Fahrenheit (f). [Formula: c/5 = f-32/9]

ALGORITHM

1. Start

2. Define a function fahrenheit_to_celsius(temp) that:

Calculates celsius = (temp - 32) × 5/9

Returns celsius

3. Define a function celsius_to_fahrenheit(temp) that:

Calculates fahrenheit = (temp × 9/5) + 32

Returns fahrenheit

4. Display: "Choose conversion: 1 for Fahrenheit to Celsius, 2 for Celsius to Fahrenheit"

5. Input choice

6. If choice == 1:

    6.1 Display: "Enter temperature in Fahrenheit:"

    6.2 Input temp

    6.3 Call fahrenheit_to_celsius(temp)

    6.4 Display: "The temperature in Celsius is: [result]"

7. Else if choice == 2:

    7.1Display: "Enter temperature in Celsius:"

    7.2 Input temp

    7.3 Call celsius_to_fahrenheit(temp)

    7.4 Display: "The temperature in Fahrenheit is: [result]"

8. Else:

    Display: "Invalid choice. Please enter 1 or 2."

9. End

PROGRAM

```
# Function to convert Fahrenheit to Celsius
def fahrenheit_to_celsius(temp):
    return (temp - 32) * 5 / 9

# Function to convert Celsius to Fahrenheit
def celsius_to_fahrenheit(temp):
    return (temp * 9 / 5) + 32

print("Choose conversion:")
print("1: Fahrenheit to Celsius")
print("2: Celsius to Fahrenheit")

    # Input conversion choice
choice = int(input("Enter your choice (1 or 2): "))

if choice == 1:
        # Input temperature in Fahrenheit
    temp = float(input("Enter the temperature in Fahrenheit: "))
        # Convert to Celsius
    celsius = fahrenheit_to_celsius(temp)
    print(f"The temperature in Celsius is: {celsius:.2f}")
elif choice == 2:
        # Input temperature in Celsius
    temp = float(input("Enter the temperature in Celsius: "))
        # Convert to Fahrenheit
    fahrenheit = celsius_to_fahrenheit(temp)
    print(f"The temperature in Fahrenheit is: {fahrenheit:.2f}")
else:
```

```
    print("Invalid choice. Please enter 1 or 2.")
```

SAMPLE OUTPUT

```
Choose conversion:
1: Fahrenheit to Celsius
2: Celsius to Fahrenheit
Enter your choice (1 or 2): 1
Enter the temperature in Fahrenheit: 212
The temperature in Celsius is: 100.00
```

EXPERIMENT 7

NESTED FOR LOOP

AIM

Program to construct patterns of stars (*), using a nested for loop.

ALGORITHM

a) Algorithm for Right-Angled Triangle

1. Start.

2. Set the number of rows (rows).

3. For each row i from 1 to rows:

Print i stars (*) in a single line.

4. End.

b) Algorithm for Inverted Triangle

1. Start

2. Set the number of rows (rows).

3. For each row i from rows down to 1:

Print i stars (*) in a single line.

4. End.

c)Algorithm for Pyramid

1. Start.

2. Set the number of rows (rows).

3. For each row i from 1 to rows:

Print (rows - i) spaces for alignment.

Print (2 * i - 1) stars (*) to form the pyramid shape.

4. End.

PROGRAM

```
# Right-Angled Triangle
print("Right-Angled Triangle:")
rows = 5
for i in range(1, rows + 1):
    for j in range(1, i + 1):
        print("*", end="")
    print()
# Inverted Triangle
print("\nInverted Triangle:")
rows = 5
for i in range(rows, 0, -1):
    for j in range(1, i + 1):
        print("*", end="")
    print()
# Pyramid
print("\nPyramid:")
rows = 5
for i in range(1, rows + 1):
    # Print leading spaces
    for j in range(rows - i):
        print(" ", end="")
    # Print stars
    for k in range(2 * i - 1):
        print("*", end="")
    print()
```

SAMPLE OUTPUT

Right-Angled Triangle:

```
*
**
***
****
*****
```

Inverted Triangle:

```
*****
****
***
**
*
```

Pyramid:

```
    *
   ***
  *****
 *******
*********
```

EXPERIMENT 8

FACTORIAL USING RECURSION

AIM

Program to find the factorial of a number using Recursion.

ALGORITHM

Step 1. Base Case: If n == 0 or n == 1, return 1.

Step 2. Recursive Case: Otherwise, calculate the factorial of n by recursively calling the function for n-1 and multiply it by n.

Step 3. Recursive Call: The recursive call continues until n reaches the base case (n == 0 or n == 1).

Step 4. Return the Result: After reaching the base case, the results of the recursive calls are multiplied and returned as the final resulT

PROGRAM

```
# Function to find the factorial using recursion
def factorial(n):
    # Base case: if n is 0 or 1, return 1
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)  # Recursive call
# Input from the user
num = int(input("Enter a number: "))
# Call the factorial function and display the result
result = factorial(num)
print(f"The factorial of {num} is {result}")
```

SAMPLE OUTPUT

```
Enter a number: 5
The factorial of 5 is 120
```

EXPERIMENT 9

FUNCTIONS

AIM

Write a program that accepts the lengths of three sides of a triangle as inputs. The program should output whether or not the triangle is a right triangle (Recall from the Pythagorean Theorem that in a right triangle, the square of one side equals the sum of the squares of the other two sides). Implement using functions.

ALGORITHM

1. Start.

2. Input the three sides of the triangle: a, b, c.

3. Determine the largest side (the potential hypotenuse):

If a > b and a > c, consider a as the hypotenuse.

Else if b > a and b > c, consider b as the hypotenuse.

Else, consider c as the hypotenuse.

4. Apply the Pythagorean theorem:

For the chosen hypotenuse:

Check if the square of the hypotenuse equals the sum of the squares of the other two sides.

5. Decision:

If the condition in step 4 is true, the triangle is a right triangle.

Otherwise, the triangle is not a right triangle.

6. Output the result.

7. Stop.

PROGRAM

```
def is_right_triangle(a, b, c):
    #Check if the triangle with sides a, b, c is a right triangle.
    # Identify the largest side as the hypotenuse
    if a > b and a > c:
        return a**2 == b**2 + c**2
    elif b > a and b > c:
        return b**2 == a**2 + c**2
    else:
        return c**2 == a**2 + b**2
# Input and processing
print("Enter the lengths of the three sides of the triangle:")
a = float(input("Side 1: "))
b = float(input("Side 2: "))
c = float(input("Side 3: "))
# Check if it's a right triangle and output the result
if is_right_triangle(a, b, c):
    print("The triangle is a right triangle.")
else:
    print("The triangle is not a right triangle.")
```

SAMPLE OUTPUT

```
Enter the lengths of the three sides of the triangle:
Side 1: 18
Side 2: 25
Side 3: 16
The triangle is not a right triangle.
Enter the lengths of the three sides of the triangle:
Side 1: 12
Side 2: 13
Side 3: 5
The triangle is a right triangle.
```

EXPERIMENT 10

MODULARIZTION

AIM

Write a Program to define a module to find Fibonacci Numbers and import the module to another program

ALGORTHM

Step a) Create the Fibonacci Module

1. Start.

2. Define the fib(n) function:

If , return .

Otherwise, recursively calculate fib(n-1) + fib(n-2).

3. End.

Step b) Main Program to Use the Module

step 1 Import the fib function from the fib_module.py.

Step 2 Ask the user for input: Prompt the user to enter the number of Fibonacci terms n they want.

Step 3 Display the Fibonacci sequence:

Step 4 Loop from 0 to .

For each iteration, call fib(i) to get the Fibonacci number at index i.

Print each Fibonacci number, separated by spaces.

PROGRAM

```
# save as fib_module.py

def fib(n):

   #Compute the n-th Fibonacci number using recursion.
```

```
    if n <= 1:
        return n
    else:
        return fib(n - 1) + fib(n - 2)
# save as main_program.py
# Import the Fibonacci function from the module
from fib_module import fib
# Input the number of Fibonacci terms from the user
n = int(input("Enter the number of Fibonacci terms you want: "))
# Display the Fibonacci sequence up to the n-th term
print("Fibonacci sequence:")
for i in range(n):
    print(fib(i), end=" ")
```

SAMPLE OUTPUT

```
Enter the number of Fibonacci terms you want: 8
Fibonacci sequence:
0 1 1 2 3 5 8 13
```