IT_LAB_1

August 8, 2025

1 IT LAB Assignment 1

Name: Dwaipayan Datta
Roll: 10200124065
Department: Computer Science and Engineering
Subject Code:
Subject:

Kalyani Government Engineering College

2 Section 1

1.1. Program to find area, perimeter of rectangle, circle.

[1]:
```
length = float(input("Enter length of rectangle: "))
breadth = float(input("Enter breadth of rectangle: "))
area = length * breadth
perimeter = 2 * (length + breadth)
print(f"Area of rectangle = {area}\nPerimeter of rectangle = {perimeter}")

radius = float(input("Enter radius of circle: "))
area = 3.14 * 3.14 * radius
perimeter = 2 * 3.14 * radius
print(f"Area of circle = {area}\nPerimeter of circle = {perimeter}")
```

```
Enter length of rectangle: 2.5
Enter breadth of rectangle: 3.5
Area of rectangle = 8.75
Perimeter of rectangle = 12.0
Enter radius of circle: 8.4
Area of circle = 82.82064000000001
Perimeter of circle = 52.752
```

1.2. Swap two numbers using and without using thrid variable.

[2]:
```
a = int(input("a = "))
b = int(input("b = "))

# Swap using thrid varible
```

```
temp = a + b
a = b
b = temp - a
print(f"a = {a}\tb = {b}")

# Swap without using third variable
a, b = b, a # returns to original state since already swapped before
print(f"a = {a}\tb = {b}")
```

```
a = 5
b = 30
a = 30  b = 5
a = 5   b = 30
```

1.3. Generate multiplication table using python.

[4]:
```
num = int(input("Enter number you want table of: "))
print(f"\n==== Table upto 12 ====\n")
for i in range(1, 13):
    print(f"{num} x {i} = {num * i}")
```

```
Enter number you want table of: 5

==== Table upto 12 ====

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
5 x 12 = 60
```

1.4. Generate prime numbers within 1 to 500.

[30]:
```
import math
def prime_check(x: int) -> bool:
  if x < 2:
    return False
  for i in range(2, (int)(math.floor(math.sqrt(x)))+1):
    if x % i == 0:
      return False
  return True;
```

```
for i in range(1, 501):
  if(prime_check(i)):
    print(f"{i}\t", end='')
```

```
2       3       5       7       11      13      17      19      23      29
31      37      41      43      47      53      59      61      67      71
73      79      83      89      97      101     103     107     109     113
127     131     137     139     149     151     157     163     167     173
179     181     191     193     197     199     211     223     227     229
233     239     241     251     257     263     269     271     277     281
283     293     307     311     313     317     331     337     347     349
353     359     367     373     379     383     389     397     401     409
419     421     431     433     439     443     449     457     461     463
467     479     487     491     499
```

1.5. Display leap years within 1900 to 2012 and show the checking of leap year.

[29]:
```
for yr in range(1900, 2012+1):
  if(yr % 4 == 0 and yr % 100 != 0 or yr % 400 == 0):
    print(f"{yr}\t", end='')
```

```
1904    1908    1912    1916    1920    1924    1928    1932    1936    1940
1944    1948    1952    1956    1960    1964    1968    1972    1976    1980
1984    1988    1992    1996    2000    2004    2008    2012
```

3 Section 2

2.1. Check number is armstrong or not.

[11]:
```
num = int(input("Enter a number: "))
order = len(str(num))
sum_of_powers = 0

temp = num
while (temp > 0):
    digit = temp % 10
    sum_of_powers += digit ** order
    temp //= 10

if num == sum_of_powers:
    print(f"{num} is an Armstrong number")
else:
    print(f"{num} is not an Armstrong number")
```

```
Enter a number: 153
153 is an Armstrong number
```

2.2. Show university gradation system using if-else.

[13]:
```
marks = float(input("Enter the student's marks: "))

if marks >= 90:
    grade = "A"
elif marks >= 80:
    grade = "B"
elif marks >= 70:
    grade = "C"
elif marks >= 60:
    grade = "D"
elif marks >= 50:
    grade = "E"
else:
    grade = "F"

print(f"The student's grade is: {grade}")
```

```
Enter the student's marks: 88
The student's grade is: B
```

2.3. Find whether a character is vowel, consonant, number, special character or not.

[14]:
```
char = input("Enter a character: ")

if len(char) == 1:
    if 'a' <= char <= 'z' or 'A' <= char <= 'Z':
        if char.lower() in 'aeiou':
            print(f"'{char}' is a vowel.")
        else:
            print(f"'{char}' is a consonant.")
    elif '0' <= char <= '9':
        print(f"'{char}' is a number.")
    else:
        print(f"'{char}' is a special character.")
else:
    print("Please enter only a single character.")
```

```
Enter a character: #
'#' is a special character.
```

2.4. Find roots of Quadraic equation and show all condition clearly in program.

[15]:
```
import cmath

# Input coefficients
a = float(input("Enter coefficient a: "))
b = float(input("Enter coefficient b: "))
c = float(input("Enter coefficient c: "))
```

```
# Calculate the discriminant
discriminant = (b**2) - (4*a*c)

# Find roots based on the discriminant
if discriminant > 0:
    # Two distinct real roots
    x1 = (-b - discriminant**0.5) / (2*a)
    x2 = (-b + discriminant**0.5) / (2*a)
    print("Two distinct real roots:")
    print(f"x1 = {x1}")
    print(f"x2 = {x2}")
elif discriminant == 0:
    # One real root (or two equal real roots)
    x = -b / (2*a)
    print("One real root:")
    print(f"x = {x}")
else:
    # Two complex roots
    x1 = (-b - cmath.sqrt(discriminant)) / (2*a)
    x2 = (-b + cmath.sqrt(discriminant)) / (2*a)
    print("Two complex roots:")
    print(f"x1 = {x1}")
    print(f"x2 = {x2}")
```

```
Enter coefficient a: 5
Enter coefficient b: 30
Enter coefficient c: 6
Two distinct real roots:
x1 = -5.792848008753788
x2 = -0.20715199124621186
```

2.5. Check whether a number is power of 3 or power of 2 or not.

[16]:
```
import math

def is_power_of(n, base):
  if n <= 0:
    return False
  if n == 1:
    return True
  # Repeatedly divide n by base until it's 1 or not divisible
  while n % base == 0:
    n /= base
  return n == 1

num = int(input("Enter a number: "))

if is_power_of(num, 2):
```

```
  print(f"{num} is a power of 2.")
elif is_power_of(num, 3):
  print(f"{num} is a power of 3.")
else:
  print(f"{num} is neither a power of 2 nor a power of 3.")
```

```
Enter a number: 81
81 is a power of 3.
```

2.6. Check whether a triangle is valid or not in terms of side and angle.

[17]:
```
def is_valid_triangle_sides(a, b, c):
  """Checks if a triangle is valid based on side lengths."""
  return (a + b > c) and (a + c > b) and (b + c > a)

def is_valid_triangle_angles(angle1, angle2, angle3):
  """Checks if a triangle is valid based on angles."""
  return ((angle1 > 0 and angle2 > 0 and angle3 > 0) and
          (angle1 + angle2 + angle3 == 180))

# Check by sides
side1 = float(input("Enter length of side 1: "))
side2 = float(input("Enter length of side 2: "))
side3 = float(input("Enter length of side 3: "))

if is_valid_triangle_sides(side1, side2, side3):
  print("The triangle is valid based on side lengths.")
else:
  print("The triangle is not valid based on side lengths.")

# Check by angles
angle1 = float(input("Enter angle 1: "))
angle2 = float(input("Enter angle 2: "))
angle3 = float(input("Enter angle 3: "))

if is_valid_triangle_angles(angle1, angle2, angle3):
  print("The triangle is valid based on angles.")
else:
  print("The triangle is not valid based on angles.")
```

```
Enter length of side 1: 3
Enter length of side 2: 4
Enter length of side 3: 5
The triangle is valid based on side lengths.
Enter angle 1: 90
Enter angle 2: 60
Enter angle 3: 30
The triangle is valid based on angles.
```

4 Section 3

3.1. Find cube of a number using function.

[18]:
```
def find_cube(num):
  """Calculates the cube of a number."""
  return num ** 3

number = float(input("Enter a number to find its cube: "))
cube = find_cube(number)
print(f"The cube of {number} is {cube}")
```

```
Enter a number to find its cube: 6
The cube of 6.0 is 216.0
```

3.2. Program to compute P(n, r) using recursive function.

[19]:
```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n-1)

def permutations(n, r):
  """Calculates the number of permutations P(n, r) using recursion."""
  if r < 0 or r > n:
    return 0  # Invalid input
  return factorial(n) // factorial(n - r)

# Input values for n and r
n = int(input("Enter the value of n: "))
r = int(input("Enter the value of r: "))

# Calculate and print the permutation
result = permutations(n, r)
print(f"The number of permutations P({n}, {r}) is: {result}")
```

```
Enter the value of n: 5
Enter the value of r: 2
The number of permutations P(5, 2) is: 20
```

3.3. Even odd checking of a number using function.

[20]:
```
def is_even(num):
  """Checks if a number is even."""
  return num % 2 == 0

def is_odd(num):
  """Checks if a number is odd."""
  return num % 2 != 0
```

```
number = int(input("Enter a number: "))

if is_even(number):
  print(f"{number} is an even number.")
elif is_odd(number):
  print(f"{number} is an odd number.")
```

```
Enter a number: 8
8 is an even number.
```

3.4. Generate Fibonacci series using function.

[21]:
```
def fibonacci_series(n_terms):
  """Generates the Fibonacci series up to n_terms."""
  if n_terms <= 0:
    return []
  elif n_terms == 1:
    return [0]
  else:
    series = [0, 1]
    while len(series) < n_terms:
      next_term = series[-1] + series[-2]
      series.append(next_term)
    return series

# Input the number of terms
num_terms = int(input("Enter the number of terms for the Fibonacci series: "))

# Generate and print the series
fib_series = fibonacci_series(num_terms)
print(f"Fibonacci series up to {num_terms} terms:")
print(fib_series)
```

```
Enter the number of terms for the Fibonacci series: 10
Fibonacci series up to 10 terms:
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

3.5. Find GCD, LCM using function.

[22]:
```
import math

def find_gcd(a, b):
  """Calculates the Greatest Common Divisor (GCD) of two numbers."""
  while(b):
    a, b = b, a % b
  return a

def find_lcm(a, b):
```

```
  """Calculates the Least Common Multiple (LCM) of two numbers."""
  # LCM * GCD = |a * b|
  return abs(a*b) // find_gcd(a, b)

# Input two numbers
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))

# Find and print GCD and LCM
gcd_result = find_gcd(num1, num2)
lcm_result = find_lcm(num1, num2)

print(f"The GCD of {num1} and {num2} is: {gcd_result}")
print(f"The LCM of {num1} and {num2} is: {lcm_result}")
```

```
Enter first number: 6
Enter second number: 81
The GCD of 6 and 81 is: 3
The LCM of 6 and 81 is: 162
```

5 Section 4

4.1. Generate pattern

```
1
2 2
3 3 3
4 4 4 4
```

[32]:
```
for i in range(1, 5):
  for j in range(1, i+1):
    print(i, end=' ')
  print()
```

```
1
2 2
3 3 3
4 4 4 4
```

4.2. Generate pattern

```
   A
  B B
 C C C
D D D D
```

[33]:
```
space = 3
for i in range(65, 69):
  for j in range(1, space+1):
```

```
    print(end=' ')
  space -= 1
  for j in range(65, i+1):
    print(chr(i), end=' ')
  print()
```

```
   A
  B B
 C C C
D D D D
```

4.3. Generate pattern

```
4 4 4 4
3 3 3
2 2
1
```

[35]:
```
for i in range(4, 0, -1):
  for j in range(1, i+1):
    print(i, end=' ')
  print()
```

```
4 4 4 4
3 3 3
2 2
1
```

4.4. Generate pattern

```
      1
    1 2
  1 2 3
1 2 3 4
```

[37]:
```
space = 3
for i in range(1, 5):
  for j in range(1, space+1):
    print(end='  ')
  space -= 1
  for j in range(1, i+1):
    print(j, end=' ')
  print()
```

```
      1
    1 2
  1 2 3
1 2 3 4
```

6 Section 5

5.1. Perform string concatenation, reverse, upper and lower case conversion with python.

[23]:
```
str1 = "Hello"
str2 = " World"

# String Concatenation
concatenated_string = str1 + str2
print(f"Concatenated String: {concatenated_string}")

# String Reverse
reversed_string = concatenated_string[::-1]
print(f"Reversed String: {reversed_string}")

# Upper Case Conversion
upper_string = concatenated_string.upper()
print(f"Upper Case String: {upper_string}")

# Lower Case Conversion
lower_string = concatenated_string.lower()
print(f"Lower Case String: {lower_string}")
```

```
Concatenated String: Hello World
Reversed String: dlroW olleH
Upper Case String: HELLO WORLD
Lower Case String: hello world
```

5.2. Show string splitting, replacing operating.

[24]:
```
my_string = "Python programming is fun and easy to learn."

# String Splitting
# Split by space
split_string_space = my_string.split()
print(f"Splitting by space: {split_string_space}")

# Split by a specific character (e.g., "is")
split_string_is = my_string.split("is")
print(f"Splitting by 'is': {split_string_is}")

# String Replacing
# Replace "fun" with "exciting"
replaced_string = my_string.replace("fun", "exciting")
print(f"Replacing 'fun' with 'exciting': {replaced_string}")

# Replace all occurrences of a character
replaced_all = my_string.replace(" ", "-")
print(f"Replacing all spaces with '-': {replaced_all}")
```

```
Splitting by space: ['Python', 'programming', 'is', 'fun', 'and', 'easy', 'to',
'learn.']
Splitting by 'is': ['Python programming ', ' fun and easy to learn.']
Replacing 'fun' with 'exciting': Python programming is exciting and easy to
learn.
Replacing all spaces with '-': Python-programming-is-fun-and-easy-to-learn.
```

5.3. Program to count number of occurences of character in a string.

[25]:
```python
my_string = input("Enter a string: ")
char_to_count = input("Enter the character to count: ")

count = 0
for char in my_string:
  if char == char_to_count:
    count += 1

print(f"The character '{char_to_count}' appears {count} times in the string.")
```

```
Enter a string: Hello World!
Enter the character to count: o
The character 'o' appears 2 times in the string.
```

5.4. Perform string slicing operation.

[26]:
```python
my_string = "Python Slicing Example"

# Slice the first 6 characters
slice1 = my_string[0:6]
print(f"Slice 1 (first 6 characters): {slice1}")

# Slice from index 7 to the end
slice2 = my_string[7:]
print(f"Slice 2 (from index 7 to end): {slice2}")

# Slice from the beginning to index 6 (exclusive)
slice3 = my_string[:6]
print(f"Slice 3 (beginning to index 6): {slice3}")

# Slice with a step of 2
slice4 = my_string[::2]
print(f"Slice 4 (with step of 2): {slice4}")

# Reverse the string using slicing
slice5 = my_string[::-1]
print(f"Slice 5 (reversed string): {slice5}")
```

```
Slice 1 (first 6 characters): Python
Slice 2 (from index 7 to end): Slicing Example
Slice 3 (beginning to index 6): Python
```

```
Slice 4 (with step of 2): Pto lcn xml
Slice 5 (reversed string): elpmaxE gnicilS nohtyP
```