1. Python Step-by-step Installation Process.

Step-by-Step Installation Process for Python

1. Installing Python on Windows

Step 1: Download Python

1. Go to the official Python website.
2. Navigate to the "Downloads" section and select "Windows".
3. Click on the latest stable release version of Python.

Step 2: Run the Installer

1. Once the download is complete, open the installer.
2. Check the box that says "Add Python to PATH". This is important for accessing Python from the command line.
3. Choose "Install now" to install Python with default settings or "Customize installation" for advanced options.

Step 3: Verify the Installation

1. Open Command Prompt.
2. Type python --version and press Enter. You should see the version of Python that you installed.

2. Implement a program based on Strings.(slicing, casting, string function)

 # String slicing

```
string1 = "Hello World"
print(string1[5:10])
print(string1[0:5])
print(string1[7:])
print(string1[:5])
print(string1[:])
print(string1[-8:-3])
print(string1[5::2])
print(string1[4:9])
print(string1[5:10].upper())
print(string1[5:8].lower())
```

 # String casting

```
number_str = "123"
n =  int(number_str)   # Casting string to integer
print(n)

number_int  = 5
n =  float(number_int)
print(n)

number_str  = "59"
n =  int(number_str)
print(n)
```

 # String functions

```
string1 = "hello world"
print(len(string1))
print(string1.upper())
print(string1.lower())
print(string1.split())
print(string1.title())
print(string1.swapcase())
print(string1.capitalize())
print(string1.casefold())
print(string1.encode())
print(string1.replace("hello", "Goodbye"))
```

Output:

Slicing:

```
Wor
Hello
orld
Hello
Hello World
```

lo Wo
ol
o Wor
WORL
Wo

Casting:
123
5.0
59

String function:
11
HELLO WORLD
hello world
['hello', 'world']
Hello World
HELLO WORLD
Hello world
hello world
'hello world'
Goodbye world

3. Implement a program based on Simple statements using Variables, Built-inData Types, Types of operator, etc.

 # variable

```
x = 20
print(x)

y = 8.9
print(y)
```

 # scope of variable

```
x = 10      # global variable

def my_function():
    y = 20    # local variable
    print(x)
    print(y)

my_function()
print("Outside function, x =", x)
```

 # Built-in Data Types

```
a = 10
print(type(a))

b = 3.14
print(type(b))

c = "Hello, World!"
print(type(c))

d = [1, 2, 3]
print(type(d))

e = (1, 2, 3)
print(type(e))

f = {"name": "John", "age": 30}
print(type(f))


# Types of Operators
x = 13
y = 2
```

 # Arithmetic Operators

```
print("Arithmetic Operators:")
print("x =", x)
print("y =", y)
print(x + y)
print(x - y)
print(x * y)
```

```
print(x / y)
print(x % y)
print(x // y)
print(x ** y)
```

 # Comparison Operators

```
print("Comparison Operators:")
print("x =", x)
print("y =", y)
print(x == y)
print(x != y)
print(x > y)
print(x < y)
print(x >= y)
print(x <= y)
```

 # Logical operator

```
print("Logical Operators:")
x = True
y = False
print("x =", x)
print("y =", y)
print(x < y and x > y)
print(x > y and x < y)
print(not(x > y and x < y))
```

 # Assignment Operators

```
print("Assignment Operators:")
x = 5
print("x =", x)
x += 3
print("x += 3, x =", x)
x -= 2
print("x -= 2, x =", x)
x *= 4
print("x *= 4, x =", x)
x /= 2
print("x /= 2, x =", x)
```

Output:

variable

```
20
8.9
```

scope of variable

```
10
20
Outside function, x = 10
```

Built-in Data Types

```
<class 'int'>
<class 'float'>
<class 'str'>
```

<class 'list'>
<class 'tuple'>
<class 'dict'>
Arithmetic Operators
Arithmetic Operators:
x = 13
y = 2
15
11
26
6.5
1
6
169
Comparison Operators
Comparison Operators:
x = 13
y = 2
False
True
True
False
True
False
Logical operator
Logical Operators:
x = True
y = False
False
False
True
Assignment Operators
Assignment Operators:
x = 5
x += 3, x = 8
x -= 2, x = 6
x *= 4, x = 24
x /= 2, x = 12.0

4. Implement a program based on decision and looping statements.

```
print("Select your Choice:")
print("1. For Loop")
print("2. While Loop")
choice = int( input())
if( choice == 1 ):
    for x in range(1,8,2):
       print(x)
elif( choice == 2 ):
   count=1;
   while( count< 8):
      print(count)
      count+=2;
else:
    print( "Ok"
```

Output:

```
Select your Choice:
1. For Loop
2. While Loop
1
1
3
5
7
Ok
```

5. Implement a program based on Tuples, Lists, Dictionaries and Sets etc.

List:

```
my_list = [1, 2, 3, 4, 5]
print(my_list)
# Access elements of the list
print("First element:",my_list[0])
print("Last element:",my_list[-1])
# Modify a list
my_list[0] = 10
print("Modified list:", my_list)
# Append an element to the list
my_list.append(6)
print("List after append:", my_list)
# Insert an element at a specific position
my_list.insert(2, 7)
print("List after insert:",my_list)
# Remove an element from the list
my_list.remove(7)
print("List after remove:", my_list)
# Sort the list
my_list.sort()
print("Sorted list:", my_list)
# Reverse the list
my_list.reverse()
print("Reversed list:", my_list)
# Get the length of the list
print("Length of the list:", len(my_list))
# Check if an element is in the list
print("Is 4  in the list?", 4 in my_list)
print("Is 8  in the list?", 8 in my_list)
#check the list
a = [5, 8,"Ram", 89.6, "shyam", 4]
b = [5, 8, 9,"Ram", 89.6, "shyam", 4]
a == b
a = [5, 8,"Ram", 89.6, "shyam", 4]
b = [5, 8, "Ram", 89.6, "shyam", 4]
a == b
```

Tuple:

```
my_tuple = (1, 2, 3, 4, 5)
print( my_tuple)
# Accessing elements of a tuple
print("Third element:", my_tuple[3])
print("Last element:", my_tuple[-1])
# Tuple concatenation
tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)
concat_tuple = tuple1 + tuple2
print("Concatenated Tuple:", concat_tuple)
# Tuple repetition
repeated_tuple = my_tuple * 2
print("Repeated Tuple:", repeated_tuple)
# Tuple addition
tuple1 = (11, 26, 34, 89)
tuple2 = (43, 57, 91, 87)
print("The first tuple is : ")
print(tuple1)
print("The second tuple is : ")
print(tuple2)
result = tuple(map(lambda i, j: i + j, tuple1, tuple2))
print("The tuple after addition is: ")
print(result)
# Get the length of the tuple
print("Length of the tuple:", len(my_tuple))
```

Dictionaries:

```
my_dict = {"name": "John", "age": 30, "city": "New York"}
print(my_dict)
# Accessing elements of a dictionary
print("Name:", my_dict["name"])
print("Age:", my_dict["age"])
# Updating a dictionary
my_dict["country"] = "USA"
print("Updated Dictionary:", my_dict)
# Deleting an element from a dictionary
del my_dict["age"]
print("Dictionary after deletion:", my_dict)
# Getting dictionary keys
print("Dictionary keys:", list(my_dict.keys()))
# Getting dictionary values
print("Dictionary values:", list(my_dict.values()))
# Getting dictionary items
print("Dictionary items:", list(my_dict.items()))
```

sets:

```
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
print(set1)
```

```
print(set2)
# Union of two sets
union_set = set1.union(set2)
print("Union of Set 1 and Set 2:", union_set)
# Intersection of two sets
intersection_set = set1.intersection(set2)
print("Intersection of Set 1 and Set 2:", intersection_set)
# Add an element to a set
set1.add(9)
print("Set 1 after adding 9:", set1)
# Remove an element from a set
set1.remove(2)
print("Set 1 after removing 2:", set1)
# Clear a set
set1.clear()
print("Set 1 after clearing:", set1)
```

Output:

List:

[1, 2, 3, 4, 5]
First element: 1
Last element: 5
Modified list: [10, 2, 3, 4, 5]
List after append: [10, 2, 3, 4, 5, 6]
List after insert: [10, 2, 7, 3, 4, 5, 6]
List after remove: [10, 2, 3, 4, 5, 6]
Sorted list: [2, 3, 4, 5, 6, 10]
Reversed list: [10, 6, 5, 4, 3, 2]
Length of the list: 6
Is 4 in the list? True
Is 8 in the list? False
False
True

Tuple:

(1, 2, 3, 4, 5)
Third element: 4
Last element: 5
Concatenated Tuple: (1, 2, 3, 4, 5, 6)
Repeated Tuple: (1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
The first tuple is :
(11, 26, 34, 89)
The second tuple is :
(43, 57, 91, 87)
The tuple after addition is:
(54, 83, 125, 176)

Length of the tuple: 5

Dictionaries:
{'name': 'John', 'age': 30, 'city': 'New York'}
Name: John
Age: 30
Updated Dictionary: {'name': 'John', 'age': 30, 'city': 'New York', 'country': 'USA'}
Dictionary after deletion: {'name': 'John', 'city': 'New York', 'country': 'USA'}
Dictionary keys: ['name', 'city', 'country']
Dictionary values: ['John', 'New York', 'USA']
Dictionary items: [('name', 'John'), ('city', 'New York'), ('country', 'USA')]

Sets:
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
Union of Set 1 and Set 2: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection of Set 1 and Set 2: {4, 5}
Set 1 after adding 9: {1, 2, 3, 4, 5, 9}
Set 1 after removing 2: {1, 3, 4, 5, 9}
Set 1 after clearing: set()

6. Implement a program based on using Functions, Function with parameters.

```
def add_numbers(a, b):
  """Calculate and return the sum of two
  numbers."""return a + b

def multiply_numbers(a, b):
  """Calculate and return the product of two
  numbers."""return a * b

def main():
  # Get user input
  num1 = float(input("Enter the first number:
  ")) num2 = float(input("Enter the second
  number: "))

  sum_result = add_numbers(num1,
  num2)
  product_result = multiply_numbers(num1, num2)

  print(f"The sum of {num1} and {num2} is:
  {sum_result}") print(f"The product of {num1} and
  {num2} is: {product_result}")

if_name_== "_main_":
    main()
```

Output:

```
Enter the first number:
10 Enter the second
number: 20
The sum of 10.0 and 20.0 is: 30.0
The product of 10.0 and 20.0 is: 200.0
```

7. Implement a program based on Functions Inside of Functions.

```
def
  arithmetic_operations
  ():def add(a, b):
    return a + b

  def multiply(a,
    b):return a *
    b

  while True:
    print("\nChoose an
    operation:")print("1.
    Addition")
    print("2.
    Multiplication")
    print("3. Exit")

    choice = input("Enter your choice

    (1-3): ")if choice == '1':
      num1 = float(input("Enter first number: "))
      num2 = float(input("Enter second
      number: "))result = add(num1, num2)
      print(f"The result of addition is: {result}")

    elif choice == '2':
      num1 = float(input("Enter first
      number: ")) num2 = float(input("Enter
      second number: "))result =
      multiply(num1, num2)
      print(f"The result of multiplication is: {result}")

    elif choice == '3':
      print("Exiting the
      program.")break

    else:
      print("Invalid choice. Please try again.")
```

```
# Run the main
function if_name_== "_
main_":
  arithmetic_operations()
```

Output

```
Choose an operation:
1. Addition
2. Multiplication
3. Exit
Enter your choice (1-
3): 1Enter first
number: 10 Enter
second number: 20
The result of addition is: 30.0

Choose an operation:
1. Addition
2. Multiplication
3. Exit
Enter your choice (1-
3): 2Enter first
number: 10 Enter
second number: 5
The result of multiplication is: 50.0

Choose an operation:
1. Addition
2. Multiplication
3. Exit
Enter your choice (1-
3): 3Exiting the
program.
```

8. Implement a program based on built-in functions.

```
def main():
    # len()
    list5 = [1, 2, 3, 4, 5, 10]
    print("Length of list5:", len(list5))  # Output: 6

    # max()
    list5 = [1, 2, 3, 4, 5, 10]
    print("Max of list5:", max(list5))  # Output: 10

    # min()
    my_list = [1, 2, 3, 4, 5, 10]
    print("Min of my_list:", min(my_list))  # Output: 1

    # zip()
    list1 = [1, 2, 3]
    list2 = ['a', 'b', 'c']
    print("Zipped lists:", list(zip(list1, list2)))  # Output: [(1, 'a'), (2, 'b'), (3,
'c')]

    # sum()
    my_list = [1, 2, 3, 4, 5, 10]
    print("Sum of my_list:", sum(my_list))  # Output: 25

    # range()
    print("Numbers from 0 to 4:", list(range(5)))  # Output: [0, 1, 2, 3, 4]

    # filter() - Filter even numbers
    def is_even(num):
        return num % 2 == 0

    numbers = [1, 2, 3, 4, 5, 6, 10]
    even_numbers = list(filter(is_even, numbers))
    print("Even numbers:", even_numbers)  # Output: [2, 4, 6, 10]
```

```
    # map() - Map function to square the numbers
    def square(num):
        return num ** 2

    squared_numbers = list(map(square, numbers))
    print("Squared numbers:", squared_numbers)  # Output: [1, 4, 9, 16, 25, 36, 100]

    # pow() - Calculate power
    print("2 to the power of 3:", pow(2, 3))  # Output: 8
    print("9 to the power of 3:", pow(9, 3))  # Output: 729

    # sorted() - Sort the list
    list5 = [34, 98, 87, 54, 12, 65, 46, 65]
    print("Sorted list5:", sorted(list5))  # Output: [12, 34, 46, 54, 65, 65, 87, 98]

    # type() - Check the type of a variable
    print("Type of list5:", type(list5))  # Output: <class 'list'>

    # count() - Count occurrences of an element
    Student_Name = ["Sanket", "Bhupendra", "Munish", "Ketan"]
    print(Student_Name.count("Munish"))  # Output: 1

    # index() - Find the index of an element
    Student_Name = ["Sanket", "Bhupendra", "Munish", "Ketan"]
    print(Student_Name.index("Sanket"))  # Output: 0

# Call the main function
if __name__ == "__main__":
    main()
```

Output:

```
Length of list5: 6
Max of list5: 10
Min of my_list: 1
Zipped lists: [(1, 'a'), (2, 'b'), (3, 'c')]
Sum of my_list: 25
Numbers from 0 to 4: [0, 1, 2, 3, 4]
Even numbers: [2, 4, 6, 10]
Squared numbers: [1, 4, 9, 16, 25, 36, 100]
2 to the power of 3: 8
9 to the power of 3: 729
Sorted list5: [12, 34, 46, 54, 65, 65, 87, 98]
Type of list5: <class 'list'>
1
0
```

9. **Implement a program using Anonymous Functions - Lambda and Filter.**

```
# lambda

    triangle = lambda m, n : 1/2*m*n
    res = triangle(34, 24)
    print("Area of the triangle:", res)



    #Arithmetic
    add = lambda a, b: a + b;
    sub = lambda a, b: a - b;
    multiply = lambda a, b: a * b;
    div = lambda a, b: a / b;
    print("value of the a:",add(45, 98))
    print("value of the b:",sub(98, 76))
    print("value of the c:",multiply(45, 98))
    print("value of the d:",div(102, 19))

# filter

     series = [23,45,57,39,1,3,95,3,8,85]

    result = filter (lambda m: m > 29, series)

    print('All the numbers greater than 29 in the series are :',list(result))
```

Output:

```
Area of the triangle: 408.0
value of the a: 143
value of the b: 22
value of the c: 4410
value of the d: 5.368421052631579

All the numbers greater than 29 in the series are : [45, 57, 39, 95, 85]
```

10. Implement a program using Maps, List Comprehension, Dictionaries.

Output:

Name to Age in Months

Conversion:Alice: 300 months

Bob: 360 months

Charlie: 264 months

Diana: 336 months

11. Implement a program for Class and Object.

```
 class Parent:# define parent class
   parentAttr = 100
   def __init__(self):
      print ("Calling parent constructor")
   def parentMethod(self):
      print ("Calling parent method")
   def setAttr(self, attr):
      self.parentAttr = attr
   def getAttr(self):
      print ("Parent attribute :", self.parentAttr)

class Child(Parent):  # define child class
   def __init__(self):
      print ("Calling child constructor")
   def childMethod(self):
      print ("Calling child method")
c = Child()
c.childMethod()
c.parentMethod()
c.setAttr(200)
c.getAttr()
```

Output:

```
Calling child constructor
Calling child method
Calling parent method
Parent attribute : 20
```

12. Demonstrate a program using Array in python.

```
import array
def main():

    # 1. Create an array of integers
    arr = array.array('i', [1, 2, 3, 4, 5])
    print("Array of integers:", arr)

    # 2. Append an element to the array
    arr.append(6)
    print("Array after appending 6:", arr)

    # 3. Insert an element at a specific index
    arr.insert(2, 10)  # Insert 10 at index 2
    print("Array after inserting 10 at index 2:", arr)

    # 4. Remove an element by value
    arr.remove(4)  # Remove the first occurrence of 4
    print("Array after removing 4:", arr)

    # 5. Pop an element by index
    popped_element = arr.pop(3)  # Pop the element at index 3
    print(f"Array after popping element at index 3: {arr}")
    print(f"Popped element: {popped_element}")

    # 6. Accessing elements by index
    print("Element at index 2:", arr[2])

    # 7. Find index of an element
    index_of_10 = arr.index(10)  # Find the index of 10
    print("Index of 10 in array:", index_of_10)

    # 8. Array slicing (similar to list slicing)
    sliced_arr = arr[1:4]  # Slice the array from index 1 to 3
    print("Sliced array from index 1 to 3:", sliced_arr)
```

```
    # 9. Length of the array
    print("Length of the array:", len(arr))

    # 10. Convert array to list
    arr_list = arr.tolist()
    print("Array converted to list:", arr_list)

# Call the main function
if __name__ == "__main__":
    main()
```

Output:

```
Array of integers: array('i', [1, 2, 3, 4, 5])
Array after appending 6: array('i', [1, 2, 3, 4, 5, 6])
Array after inserting 10 at index 2: array('i', [1, 2, 10, 3, 4, 5, 6])
Array after removing 4: array('i', [1, 2, 10, 3, 5, 6])
Array after popping element at index 3: array('i', [1, 2, 10, 5, 6])
Popped element: 3
Element at index 2: 10
Index of 10 in array: 2
Sliced array from index 1 to 3: array('i', [2, 10, 3])
Length of the array: 5
Array converted to list: [1, 2, 10, 5, 6]
```

13. **Implement a program to Create, Import and use Package and Modules.**

1. **Create a directory called** my_package1**.**
   !mkdir my_package1

2. **Inside this directory, create a file called __init__.py. This can be empty or containinitialization code. It marks the directory as a package.**
   %%writefile my_package1/_init__.py
   #_init_.py can be empty or contain initialization
   codeOutput: Overwriting my_package1/_init_.py

3. **Move the math_operations.py module into the my_package1 directory.**
   %%writefile
   my_package1/math_operations.py#
   math_operations.py
   def add(a,
   b): return
   a + b

   def subtract(a,
   b):return a -
   b

   def multiply(a,
   b):return a *
   b

   def divide(a,
   b):if b == 0:
   raise ValueError("Cannot divide by
   zero")return a / b
4. **# main.py**
   from my_package1 import math_operations

   # Using the functions from math_operations
   modulea = 10
   b = 5

```
print(f"{a} + {b} = {math_operations.add(a, b)}")
print(f"{a}{b}={math_operations.subtract(a,
b)}")
print(f"{a}*{b}={math_operations.multiply(a
,b)}")
print(f"{a}/{b}={math_operations.divide(a,
b)}")
```

Output: 10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0

In [1]:
```
!mkdir my_package1
```

In [13]:
```
%%writefile my_package1/__init__.py
# __init__.py can be empty or contain initialization code
```

```
Overwriting my_package1/__init__.py
```

In [14]:
```
%%writefile my_package1/math_operations.py
# math_operations.py
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b
```

```
Writing my_package1/math_operations.py
```

In [15]:
```
# main.py
from my_package1 import math_operations

# Using the functions from math_operations module
a = 10
b = 5

print(f"{a} + {b} = {math_operations.add(a, b)}")
print(f"{a} - {b} = {math_operations.subtract(a, b)}")
print(f"{a} * {b} = {math_operations.multiply(a, b)}")
print(f"{a} / {b} = {math_operations.divide(a, b)}")

```

```
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0
```

This example demonstrates how to create, import, and use a package and modules inPython. The package my_package contains the module math_operations, which is imported and used in the main.py script.

Another Program:

File 1)

mymodule.py

```
person1 = {
 "name":
"John","age":
36,
"country": "Norway"
 }
```

File 2) File2.py

```
import mymodule #mymodule is the first
file namea = mymodule.person1["age"]
print(a)
```

Note: Run File2.py (save File 1 as a name mymodule.py and this name use asa package in File 2)

14. Implement a program based on Writing Text Files, Appending Text to a File,Reading Text Files.

```
# Define fil path
file_path
='abc.txt’

# 1. Writing to a Text File
def write_to_file(file_path,
  text): with open(file_path,
  'w') as file:
    file.write(text)
  print("Text has been written to the file.")

# 2. Appending Text to a File
def append_to_file(file_path,
  text):with open(file_path,
  'a') as file:
    file.write(text)
  print("Text has been appended to the file.")

# 3. Reading from a Text
Filedef
read_from_file(file_path:
  try:
    with open(file_path, 'r') as
      file:content = file.read()
    print("File content read
    successfully.")return content
  except File Not Found
    Error:
  print("File not found.")
    return None

# Main program logic
if__name__== "__main__":
  # Write initial content to the file
  initial_text = "This is the initial text in the
```

```
file.\n"write_to_file(file_path, initial_text)

# Append additional content to the file
additional_text = "This is the additional text being
appended.\n"append_to_file(file_path, additional_text)

# Read the content of the file
content =
read_from_file(file_path)if
content is not None:
   print("File
   Content:")
   print(content)
```

Output:

```
Text has been written to the
file. Text has been appended
to the file.File content read
successfully.
File Content:
This is the initial text in the file.
This is the additional
text being appended.
```

15. Demonstrate a program Paths and Directories, File Information, Renaming,Moving, Copying, and Removing Files.

```
import os
import shutil

# Step 1: Paths and Directories
new_dir = os.path.join(os.getcwd(), "test_dir")
os.makedirs(new_dir, exist_ok=True)

# Step 2: File Information and Creation
file_name = os.path.join(new_dir, "sample.txt")
with open(file_name, 'w') as f: f.write("This is a sample file.")
print(f"\nFile Path: {file_name}\nExists: {os.path.exists(file_name)}"
    f"\nSize: {os.path.getsize(file_name)} bytes\nModified:
{os.path.getmtime(file_name)}"
    f"\nAbsolute Path: {os.path.abspath(file_name)}")

# Step 3: Renaming, Moving and Copying the file
new_file_name = os.path.join(os.getcwd(), "renamed_sample.txt")
shutil.move(file_name, new_file_name)
copied_file_path = new_file_name.replace("renamed", "copied")
shutil.copy(new_file_name, copied_file_path)
print(f"File Moved to: {new_file_name}\nFile Copied to:
{copied_file_path}")

# Step 4: Removing the files and Directory
os.remove(new_file_name)
os.remove(copied_file_path)
os.rmdir(new_dir)
print(f"Files and Directory Removed.")
```

output:

File Path: C:\Users\IMRDCOM04\test_dir\sample.txt
Exists: True
Size: 22 bytes
Modified: 1734674147.1673594
Absolute Path: C:\Users\IMRDCOM04\test_dir\sample.txt
File Moved to: C:\Users\IMRDCOM04\renamed_sample.txt
File Copied to: C:\Users\IMRDCOM04\copied_sample.txt
Files and Directory Removed.

16. Implement a program to creating Types of GUI Widgets, Resizing, ConfiguringOptions.

```
import tkinter as tk
from tkinter import ttk

    # Initialize the main
    windowroot = tk.Tk()
    root.title("GUI Widget Example")

    # Resize the window
    root.geometry("400x300")

    # Step 1: Create GUI
    Widgets# Label Widget
    label = tk.Label(root, text="Hello, Tkinter!", font=("Arial", 16),
    fg="blue")label.pack(pady=10)  # Add some padding for spacing

    # Entry Widget (Text box)
    entry = tk.Entry(root, width=30, font=("Arial",
    12))entry.insert(0, "Enter text here...")
    entry.pack(pady=10)

    # Button Widget
    def on_button_click():
    label.config(text=f"Hello, {entry.get()}!")  # Update label based on entry text

    button = tk.Button(root, text="Click Me!", command=on_button_click,
    font=("Arial",12))
    button.pack(pady=10)

    # Combobox Widget
    combo = ttk.Combobox(root, values=["Option 1", "Option 2", "Option 3"],
    font=("Arial", 12))
    combo.current(0) # Set default value
    combo.pack(pady=10)
```

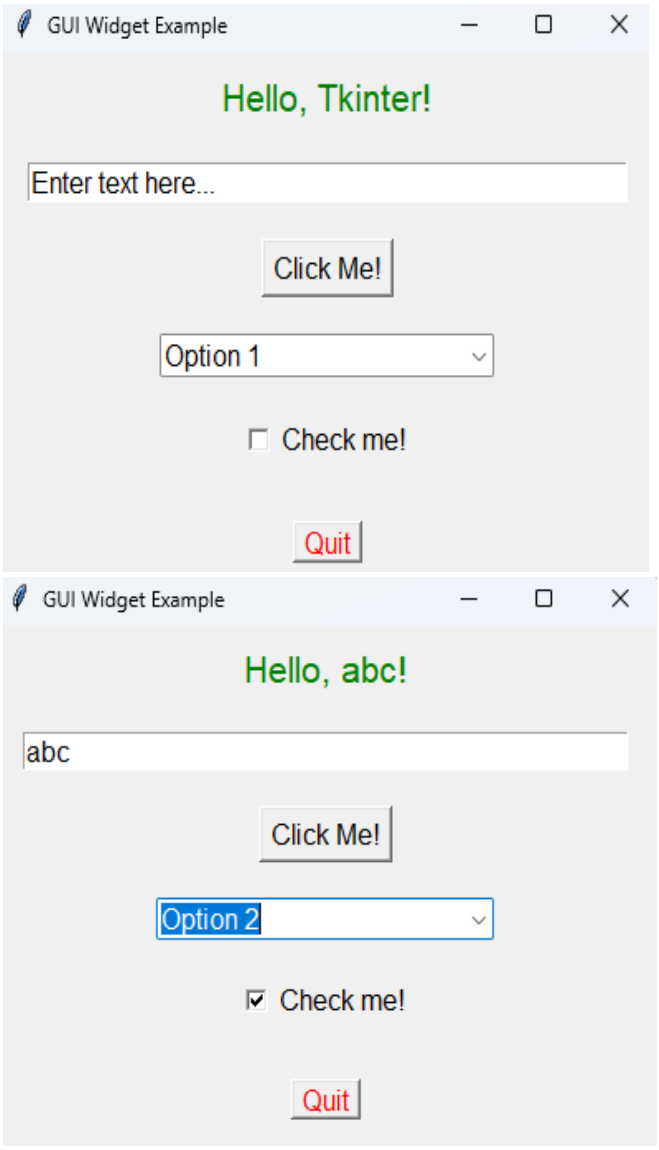
```
# Checkbox Widget
check_var =
tk.IntVar()

checkbox = tk.Checkbutton(root, text="Check me!", variable=check_var,
font=("Arial",12))
checkbox.pack(pady=10)
# Step 2: Resizing and Configuring Options
# Resize widgets using the `pack` method options
# pady and padx are used for vertical and horizontal
padding label.config(fg="green") # Configure label to
change text colorentry.config(width=40)  # Configure
entry to change width

# Button to quit the application
quit_button = tk.Button(root, text="Quit", command=root.quit,
font=("Arial", 12),fg="red")
quit_button.pack(pady=20)

#start the GUI event loop
root.mainloop()
```

Output:


GUI Widget Example
Hello, Tkinter!
Enter text here...
Click Me!
Option 1
Check me!
Quit
GUI Widget Example
Hello, abc!
abc
Click Me!
Option 2
Check me!
Quit

17. Implement a program to Creating Layouts, Packing Order, Controlling WidgetAppearances.

```
import tkinter as tk

# Initialize the main
windowroot = tk.Tk()
root.title("Layout and Packing Order
Example")root.geometry("400x400")

# Step 1: Create Frames for Layout
Management# Frame 1 at the top
(packing at the top)
frame1 = tk.Frame(root, bg="lightblue", height=100,
width=400)frame1.pack(fill="x", padx=10, pady=10)

# Frame 2 in the middle (packing at the left and right inside
this frame)frame2 = tk.Frame(root, bg="lightgreen",
height=100, width=400) frame2.pack(fill="both",
expand=True, padx=10, pady=10)

# Frame 3 at the bottom (packing at the bottom)
frame3 = tk.Frame(root, bg="lightyellow", height=100, width=400)
frame3.pack(fill="x", padx=10, pady=10)

# Step 2: Add Widgets to Frame 1 (Top)
label1 = tk.Label(frame1, text="Top Frame (Light Blue)", font=("Arial",
14))
label1.pack(pady=20)

# Step 3: Add Widgets to Frame 2 (Middle)
label2 = tk.Label(frame2, text="Middle Frame (Light Green)",
font=("Arial", 14))label2.pack(side="left", padx=20, pady=20)

button1 = tk.Button(frame2, text="Button 1",
font=("Arial", 12))button1.pack(side="left", padx=20)
```

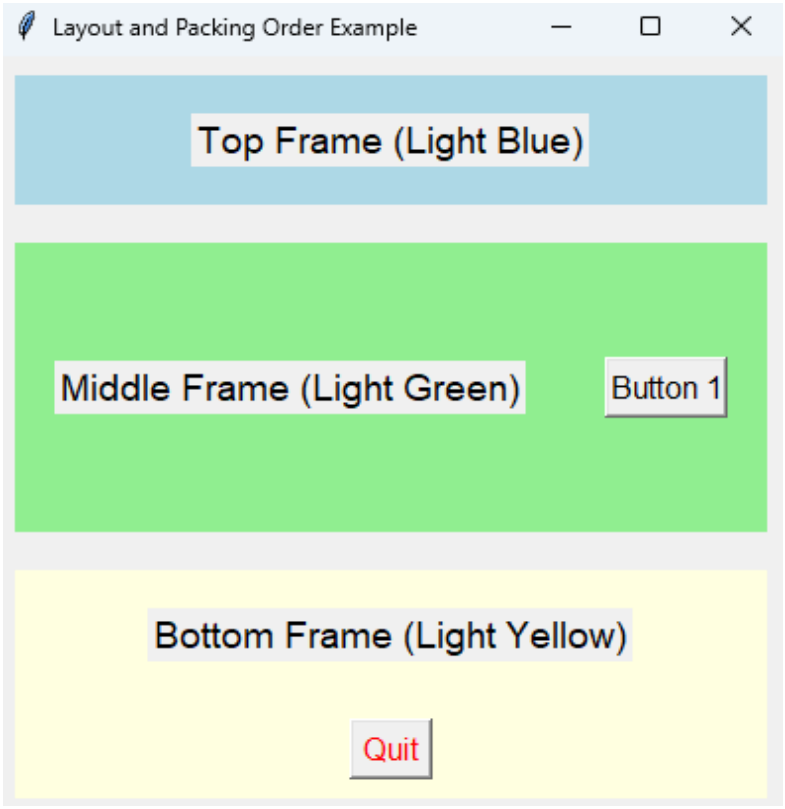
```
button2 = tk.Button(frame2, text="Button 2",
font=("Arial", 12))button2.pack(side="right",padx=20)

# Step 4: Add Widgets to Frame 3 (Bottom)
label3 = tk.Label(frame3, text="Bottom Frame(Light
Yellow)",font=("Arial", 14))
label3.pack(pady=20)

quit_button = tk.Button(frame3, text="Quit", font=("Arial", 12),
command=root.quit, fg="red")
quit_button.pack(pady=10)

# Start the GUI event loop
root.mainloop()
```

Output:



17. Implement a program based on Text Processing, Searching for Files.

```
import  os
import  fnmatch

def search_files(directory, pattern):
   """Search for files in the directory matching the given pattern."""
   matches = []  # List to store the matched file paths

   # Walk through the directory
   for root, dirs, files in os.walk(directory):
      for filename in fnmatch.filter(files, pattern):  # Match files against the
pattern
         matches.append(os.path.join(root, filename))  # Add the full file
path to matches

   return matches

if __name__ == "__main__":
   # Specify the directory and file pattern
   search_directory = input("Enter the directory to search in: ")
   file_pattern = input("Enter the file pattern (e.g., *.txt): ")

   # Search for files
   found_files = search_files(search_directory, file_pattern)

   # Display results
   if found_files:
      print("Found files:")
      for file in found_files:
         print(file)
   else:
      print("No files found matching the pattern.")
```

Output:

Enter the directory to search
in:demo1 Enter the file pattern
(e.g., *.txt): ABC.txtFound files:
demo1\ABC.txt

18. Implement a program based on HTML Parsing.

First step: pip install beautifulsoup4

```
from bs4 import BeautifulSoup

html_content = """
<html>
<head>
<title>Simple HTML Page</title>
</head>
<body>
<h1>Main Heading</h1>
<p>This is the first paragraph of the page.</p>
<p>This is the second paragraph with <a href="#">a link</a>.</p>
</body>
</html>"""

# Parse the HTML content with BeautifulSoup
soup = BeautifulSoup(html_content, 'html.parser')
# Extract and print the title
title = soup.title.string
print(f"Title: {title}")
# Extract and print the heading (h1)
heading = soup.h1.string
print(f"Heading: {heading}")
# Extract and print all paragraphs
paragraphs = soup.find_all('p')
for i, paragraph in enumerate(paragraphs, 1):
print(f"Paragraph {i}: {paragraph.text}")
```

Output:

Title: Simple HTML Page
Heading: Main Heading
Paragraph 1: This is the first paragraph of the page.
Paragraph 2: This is the second paragraph with a link.

20. Demonstrate a program using DBM - Creating and Accessing Persistent Dictionaries.

```
import dbm
# Creating and storing key-value pairs in a DBM database
def create_db():
   with dbm.open('example_db', 'c') as db: # 'c' mode means create if it doesn't exist
      # Add key-value pairs to the database
      db[b'name'] = b'John Doe'
      db[b'age'] = b'30'
      db[b'city'] = b'New York'
      db[b'occupation'] = b'Software Developer'
      print("Data has been written to the DBM database.")
   # Accessing and retrieving key-value pairs from the DBM database
def access_db():
   with dbm.open('example_db', 'r') as db: # 'r' mode means read-only
      # Access and print values from the database
      name = db.get(b'name', b'Unknown').decode('utf-8')
      age = db.get(b'age', b'Unknown').decode('utf-8')
      city = db.get(b'city', b'Unknown').decode('utf-8')
      occupation = db.get(b'occupation', b'Unknown').decode('utf-8')
      print(f"Name: {name}")
      print(f"Age: {age}")
      print(f"City: {city}")
      print(f"Occupation: {occupation}")
      # Delete a key from the DBM database
def delete_from_db():
```

```
    with dbm.open('example_db', 'w') as db: # 'w' mode means
read/write (must exist)
        del db[b'age'] # Delete the 'age' key
        print("Key 'age' has been deleted.")
# Run the functions
create_db() # Create the DB and store data
access_db() # Access and print the stored data
delete_from_db() # Delete an entry
access_db() # Access the data again to check deletion
```

Output:

```
Data has been written to the DBM database.
Name: John Doe
Age: 30
City: New York
Occupation: Software Developer
Key 'age' has been deleted.
Name: John Doe
Age: Unknown
City: New York
Occupation: Software Developer
```

21. Demonstrate a program using Relational Database - Writing SQL Statements, Defining Tables, Setting Up a Database.(Transactions and Committing the Results.)

```
import sqlite3
# Connect to SQLite database (it will create the database if it doesn't exist)
def create_database():
    conn = sqlite3.connect('company.db') # 'company.db' is the database file
    cursor = conn.cursor()
    # Create a table (DDL statement)
    cursor.execute('''CREATE TABLE IF NOT EXISTS employees (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    department TEXT NOT NULL,
    salary REAL NOT NULL,
    hire_date TEXT NOT NULL
    )''')
    print("Table 'employees' created successfully.")
    conn.commit() # Commit the DDL changes
    conn.close() # Close the connection
    # Insert multiple employee records in a transaction
def insert_data():
    conn = sqlite3.connect('company.db')
    cursor = conn.cursor()
    try:
        # Begin transaction
        cursor.execute("BEGIN TRANSACTION;")
        # Insert some data into employees table (DML statement)
        cursor.execute("INSERT INTO employees (name, department, salary, hire_date) VALUES (?, ?, ?, ?)",
        ('John Doe', 'HR', 50000, '2020-01-15'))
```

```
        cursor.execute("INSERT INTO employees (name, department,
salary, hire_date) VALUES (?, ?, ?, ?)",
        ('Jane Smith', 'IT', 70000, '2019-07-01'))
        cursor.execute("INSERT INTO employees (name, department,
salary, hire_date) VALUES (?, ?, ?, ?)",
        ('Mike Johnson', 'Finance', 60000, '2021-05-12'))
        # Commit the transaction
        conn.commit()
        print("Transaction committed successfully.")
    except sqlite3.Error as e:
        # Rollback in case of any error
        conn.rollback()
        print(f"An error occurred: {e}")
    finally:
        # Close the connection
        conn.close()
        # Fetch data from the employees table and display it
def fetch_data():
    conn = sqlite3.connect('company.db')
    cursor = conn.cursor()
    # Query to fetch all records from the employees table
    cursor.execute("SELECT * FROM employees")
    # Fetch all rows and print them
    rows = cursor.fetchall()
    for row in rows:
        print(row)
        conn.close()
        # Main function to execute all steps
if __name__ == '__main__':
    create_database() # Step 1: Set up the database and define the table
    insert_data() # Step 2: Insert data with transaction and commit
```

```
fetch_data() # Step 3: Fetch and display the data
```

output:

```
Table 'employees' created successfully.
Transaction committed successfully.
(1, 'John Doe', 'HR', 50000.0, '2020-01-15')
(2, 'Jane Smith', 'IT', 70000.0, '2019-07-01')
(3, 'Mike Johnson', 'Finance', 60000.0, '2021-05-12')
```