Part A
1. Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number as an argument. 2. Python function that takes a list and returns a new list with unique elements of the first list. 3. Python program of recursion list sum. 4. Python program to get the sum of digits of a non-negative integer. 5. Python program to demonstrate any 5 string and List operations. 6. Create an output tuple that converts the words to uppercase from the input tuple of words. 7. Python program to demonstrate any 5 operations performed on dictionary. 8. Python program to create a module Calculation.py that contains functions to perform basic arithmetic operations. Demonstrate importing the module.
Part B
1. Python program to demonstrate modification of an existing table data from MySQL database. 2. Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle. 3. Python class named Rectangle constructed by a length and width and a method which will compute the area and perimeter of rectangle. Inherit a class Box that contains additional method volume. Override the perimeter method to compute perimeter of a Box. 4. Python program to show use of Regular expressions with match(), search(), findall(), sub() and split(). 5. python program to demonstrate Exception handling using 'try', 'except', 'finally' and 'else block. 6. Python program to read a file line by line store it into an array. 7. Python GUI program to design Student Registration Form using any 5 widgets.

Part-A

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

```python
def factorial(n):
    if not isinstance(n, int):
        raise TypeError("Input must be an integer.")
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers.")
    if n == 0 or n == 1:
        return 1
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result
if __name__ == "__main__":
    print("Testing factorial function:")
    try:
        print("Factorial of 0: {}".format(factorial(0)))    # Expected: 1
        print("Factorial of 1: {}".format(factorial(1)))    # Expected: 1
        print("Factorial of 5: {}".format(factorial(5)))    # Expected: 120 (5*4*3*2*1)
        print("Factorial of 7: {}".format(factorial(7)))    # Expected: 5040
    except (ValueError, TypeError) as e:
        print("Error during valid input test: {}".format(e))
    print("\nTesting with invalid inputs:")
    try:
        print("Factorial of -3: {}".format(factorial(-3)))
    except ValueError as e:
        print("Caught expected error for -3: {}".format(e))
    try:
        print("Factorial of 4.5: {}".format(factorial(4.5)))
    except TypeError as e:
        print("Caught expected error for 4.5: {}".format(e))
    try:
        print("Factorial of 'abc': {}".format(factorial('abc')))
    except TypeError as e:
        print("Caught expected error for 'abc': {}".format(e))
```

```
<terminated> F:\eclipse-python-32-64\64biteclipse\BCA3python\1.py
Testing factorial function:
Factorial of 0: 1
Factorial of 1: 1
Factorial of 5: 120
Factorial of 7: 5040

Testing with invalid inputs:
Caught expected error for -3: Factorial is not defined for negative numbers.
Caught expected error for 4.5: Input must be an integer.
Caught expected error for 'abc': Input must be an integer.
```

2. **Python function that takes a list and returns a new list with unique elements of the first list.**

```python
def get_unique_elements(input_list):
    seen = set()
    unique_list = []

    for item in input_list:
        if item not in seen:
            unique_list.append(item)
            seen.add(item)

    return unique_list
if __name__ == "__main__":
    print("Testing get_unique_elements function:")
    list1 = [1, 2, 2, 3, 4, 4, 5, 1]
    unique1 = get_unique_elements(list1)
    print("Original List 1: {}".format(list1))
    print("Unique Elements 1: {}".format(unique1)) # Expected: [1, 2, 3, 4, 5]

    print("\n" + "="*30 + "\n")
    list2 = ['apple', 'banana', 'orange', 'apple', 'grape', 'banana', 10, 20, 10]
    unique2 = get_unique_elements(list2)
    print("Original List 2: {}".format(list2))
    print("Unique Elements 2: {}".format(unique2)) # Expected: ['apple', 'banana', 'orange', 'grape

    print("\n" + "="*30 + "\n")
    list3 = [100, 200, 300]
    unique3 = get_unique_elements(list3)
    print("Original List 3: {}".format(list3))
    print("Unique Elements 3: {}".format(unique3)) # Expected: [100, 200, 300]

    print("\n" + "="*30 + "\n")
    list4 = []
    unique4 = get_unique_elements(list4)
    print("Original List 4 (empty): {}".format(list4))
    print("Unique Elements 4: {}".format(unique4)) # Expected: []

    print("\n" + "="*30 + "\n")
    list5 = [7, 7, 7, 7]
    unique5 = get_unique_elements(list5)
    print("Original List 5: {}".format(list5))
    print("Unique Elements 5: {}".format(unique5)) # Expected: [7]
```

```
Console

<terminated> F:\eclipse-python-32-64\64biteclipse\BCA3python\2.py
Testing get_unique_elements function:
Original List 1: [1, 2, 2, 3, 4, 4, 5, 1]
Unique Elements 1: [1, 2, 3, 4, 5]

==============================

Original List 2: ['apple', 'banana', 'orange', 'apple', 'grape', 'banana', 10, 20, 10]
Unique Elements 2: ['apple', 'banana', 'orange', 'grape', 10, 20]

==============================

Original List 3: [100, 200, 300]
Unique Elements 3: [100, 200, 300]

==============================

Original List 4 (empty): []
Unique Elements 4: []

==============================

Original List 5: [7, 7, 7, 7]
Unique Elements 5: [7]
```

3. Python program of recursion list sum.

```python
def recursive_list_sum(data_list):
    if not data_list:
        return 0
    else:
        return data_list[0] + recursive_list_sum(data_list[1:])
list_a = [1, 2, 3, 4, 5]
sum_a = recursive_list_sum(list_a)
print("Input List:", list_a)
print("Sum of elements:", sum_a)
print("-" * 25)

# Input 2: A list with a single element
list_b = [99]
sum_b = recursive_list_sum(list_b)
print("Input List:", list_b)
print("Sum of elements:", sum_b)
print("-" * 25)

# Input 3: An empty list
list_c = []
sum_c = recursive_list_sum(list_c)
print("Input List:", list_c)
print("Sum of elements:", sum_c)
print("-" * 25)

# Input 4: A list with negative numbers
list_d = [-10, 5, -2, 8]
sum_d = recursive_list_sum(list_d)
print("Input List:", list_d)
print("Sum of elements:", sum_d)
print("-" * 25)
```

Console

```
<terminated> F:\eclipse-python-32-64\64biteclipse\BCA3python\3.py
Input List: [1, 2, 3, 4, 5]
Sum of elements: 15
-------------------------
Input List: [99]
Sum of elements: 99
-------------------------
Input List: []
Sum of elements: 0
-------------------------
Input List: [-10, 5, -2, 8]
Sum of elements: 1
-------------------------
```

4. Python program to get the sum of digits of a non-negative integer.

```python
def sum_of_digits(n):
    if not isinstance(n, int) or n < 0:
        return "Error: Input must be a non-negative integer."

    if n == 0:
        return 0

    s = str(n)
    digit_sum = 0
    for digit_char in s:
        digit_sum += int(digit_char)
    return digit_sum
number_1 = 12345
result_1 = sum_of_digits(number_1)
print("The sum of the digits of {} is: {}".format(number_1, result_1))
print("-" * 35)
number_2 = 7
result_2 = sum_of_digits(number_2)
print("The sum of the digits of {} is: {}".format(number_2, result_2))
print("-" * 35)
number_3 = 0
result_3 = sum_of_digits(number_3)
print("The sum of the digits of {} is: {}".format(number_3, result_3))
print("-" * 35)
number_4 = -456
result_4 = sum_of_digits(number_4)
print("The sum of the digits of {} is: {}".format(number_4, result_4))
print("-" * 35)
number_5 = 12.5
result_5 = sum_of_digits(number_5)
print("The sum of the digits of {} is: {}".format(number_5, result_5))
print("-" * 35)
```

Console

```
<terminated> F:\eclipse-python-32-64\64biteclipse\BCA3python\4.py
The sum of the digits of 12345 is: 15
-----------------------------------
The sum of the digits of 7 is: 7
-----------------------------------
The sum of the digits of 0 is: 0
-----------------------------------
The sum of the digits of -456 is: Error: Input must be a non-negative integer.
-----------------------------------
The sum of the digits of 12.5 is: Error: Input must be a non-negative integer.
-----------------------------------
```

5. Python program to demonstrate any 5 string and List operations.

5

```python
my_string = "hello world"
my_list = [10, 20, 30, 40, 50, 20]

print("Original String:", my_string)
print("Original List:", my_list)
upper_string = my_string.upper()
print("Uppercase String:", upper_string)
my_list.append(60)
print("List after append:", my_list)
ends_with_world = my_string.endswith("world")
print("Does string end with 'world'?:", ends_with_world)
my_list.remove(20)
print("List after removing first 20:", my_list)
count_l = my_string.count('l')
print("Count of 'l' in string:", count_l)
my_list.insert(2, 25)
print("List after inserting 25 at index 2:", my_list)
split_string = my_string.split(' ')
print("String split into a list:", split_string)
my_list.sort()
print("List after sorting:", my_list)

# String Operation 5: Check if the string is alphanumeric
is_alpha_num = my_string.isalnum()
print("Is string alphanumeric?:", is_alpha_num)

# List Operation 5: Extend the list with elements from another list
my_list.extend([70, 80])
print("List after extend:", my_list)
```

Console

```
<terminated> F:\eclipse-python-32-64\64biteclipse\BCA3python\5.py
Original String: hello world
Original List: [10, 20, 30, 40, 50, 20]
Uppercase String: HELLO WORLD
List after append: [10, 20, 30, 40, 50, 20, 60]
Does string end with 'world'?: True
List after removing first 20: [10, 30, 40, 50, 20, 60]
Count of 'l' in string: 3
List after inserting 25 at index 2: [10, 30, 25, 40, 50, 20, 60]
String split into a list: ['hello', 'world']
List after sorting: [10, 20, 25, 30, 40, 50, 60]
Is string alphanumeric?: False
List after extend: [10, 20, 25, 30, 40, 50, 60, 70, 80]
```

6. **Create an output tuple that converts the words to uppercase from the input tuple of words.**

```python
def convert_to_uppercase(word_tuple):

    return tuple(word.upper() for word in word_tuple)
input_tuple_1 = ('hello', 'world', 'python')
output_tuple_1 = convert_to_uppercase(input_tuple_1)

print("Original Tuple 1:", input_tuple_1)
print("Uppercase Tuple 1:", output_tuple_1)
print("-" * 30)
input_tuple_2 = ('Data', 'Science', 'a', 'b', 'c')
output_tuple_2 = convert_to_uppercase(input_tuple_2)

print("Original Tuple 2:", input_tuple_2)
print("Uppercase Tuple 2:", output_tuple_2)
print("-" * 30)
input_tuple_3 = ('apple', '', '123', 'banana')
output_tuple_3 = convert_to_uppercase(input_tuple_3)

print("Original Tuple 3:", input_tuple_3)
print("Uppercase Tuple 3:", output_tuple_3)
print("-" * 30)

# Input 4: An empty tuple
input_tuple_4 = ()
output_tuple_4 = convert_to_uppercase(input_tuple_4)

print("Original Tuple 4 (empty):", input_tuple_4)
print("Uppercase Tuple 4:", output_tuple_4)
```

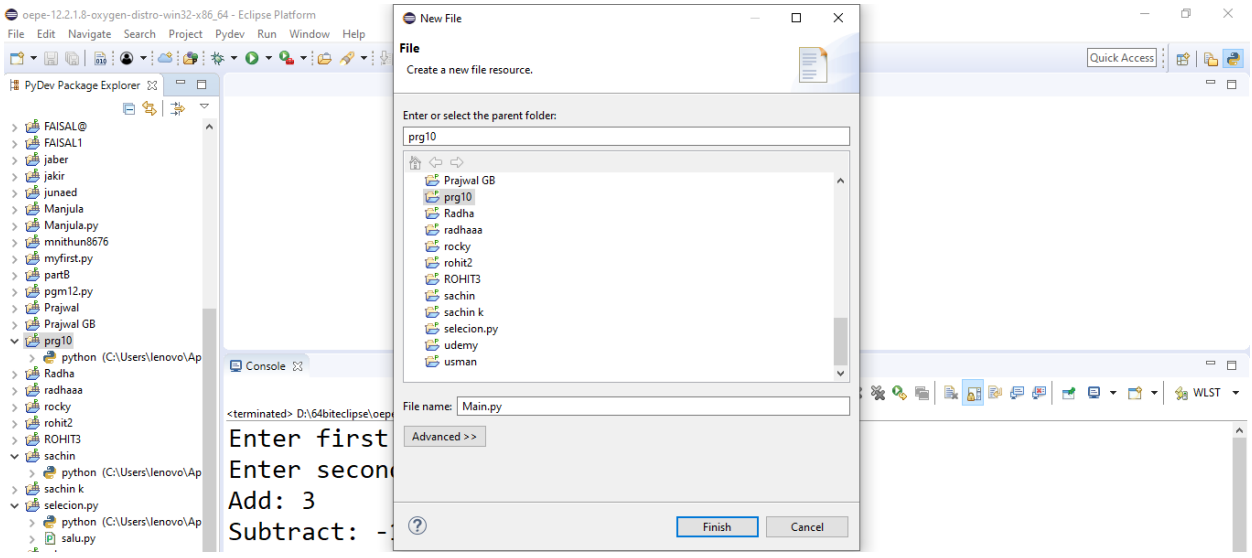
```
Console

<terminated> F:\eclipse-python-32-64\64biteclipse\BCA3python\6.py
Original Tuple 1: ('hello', 'world', 'python')
Uppercase Tuple 1: ('HELLO', 'WORLD', 'PYTHON')
------------------------------
Original Tuple 2: ('Data', 'Science', 'a', 'b', 'c')
Uppercase Tuple 2: ('DATA', 'SCIENCE', 'A', 'B', 'C')
------------------------------
Original Tuple 3: ('apple', '', '123', 'banana')
Uppercase Tuple 3: ('APPLE', '', '123', 'BANANA')
------------------------------
Original Tuple 4 (empty): ()
Uppercase Tuple 4: ()
```

7. **Python program to demonstrate any 5 operations performed on dictionary.**

```python
def demonstrate_dict_operations(input_dict, description):
    print("--- Demonstrating operations for:", description, "---")
    print("Original dictionary:", input_dict)

    input_dict['language'] = 'Python'
    print("After adding a new key 'language':", input_dict)

    input_dict['author'] = 'Jane Doe'
    print("After updating the value of 'author':", input_dict)

    favorite_color = input_dict.get('color', 'No color specified')
    print("Accessing the value of 'color' using .get():", favorite_color)

    if 'year' in input_dict:
        removed_year = input_dict.pop('year')
        print("After removing 'year', the dictionary is:", input_dict)
        print("The value removed was:", removed_year)

    print("All keys in the final dictionary:", list(input_dict.keys()))
    print("All values in the final dictionary:", list(input_dict.values()))
    print("\n" + "="*50 + "\n")


my_dict_1 = {'title': 'The Great Gatsby', 'author': 'F. Scott Fitzgerald', 'year': 1925}
demonstrate_dict_operations(my_dict_1, "Dictionary 1 (Strings and Integers)")

my_dict_2 = {1: 'apple', 2: 'banana', 3: 'cherry'}
demonstrate_dict_operations(my_dict_2, "Dictionary 2 (Integer Keys)")

my_dict_3 = {}
demonstrate_dict_operations(my_dict_3, "Dictionary 3 (Empty Dictionary)")
```

```
Console
<terminated> F:\eclipse-python-32-64\64biteclipse\BCA3python\7.py
--- Demonstrating operations for: Dictionary 1 (Strings and Integers) ---
Original dictionary: {'title': 'The Great Gatsby', 'author': 'F. Scott Fitzgerald', 'year': 1925}
After adding a new key 'language': {'title': 'The Great Gatsby', 'author': 'F. Scott Fitzgerald', 'year': 1925, 'language': 'Python'}
After updating the value of 'author': {'title': 'The Great Gatsby', 'author': 'Jane Doe', 'year': 1925, 'language': 'Python'}
Accessing the value of 'color' using .get(): No color specified
After removing 'year', the dictionary is: {'title': 'The Great Gatsby', 'author': 'Jane Doe', 'language': 'Python'}
The value removed was: 1925
All keys in the final dictionary: ['title', 'author', 'language']
All values in the final dictionary: ['The Great Gatsby', 'Jane Doe', 'Python']

==================================================

--- Demonstrating operations for: Dictionary 2 (Integer Keys) ---
Original dictionary: {1: 'apple', 2: 'banana', 3: 'cherry'}
After adding a new key 'language': {1: 'apple', 2: 'banana', 3: 'cherry', 'language': 'Python'}
After updating the value of 'author': {1: 'apple', 2: 'banana', 3: 'cherry', 'language': 'Python', 'author': 'Jane Doe'}
Accessing the value of 'color' using .get(): No color specified
All keys in the final dictionary: [1, 2, 3, 'language', 'author']
All values in the final dictionary: ['apple', 'banana', 'cherry', 'Python', 'Jane Doe']

==================================================

--- Demonstrating operations for: Dictionary 3 (Empty Dictionary) ---
Original dictionary: {}
After adding a new key 'language': {'language': 'Python'}
After updating the value of 'author': {'language': 'Python', 'author': 'Jane Doe'}
```

8. Python program to create a module Calculation.py that contains functions to perform basic arithmetic operations. Demonstrate importing the module.

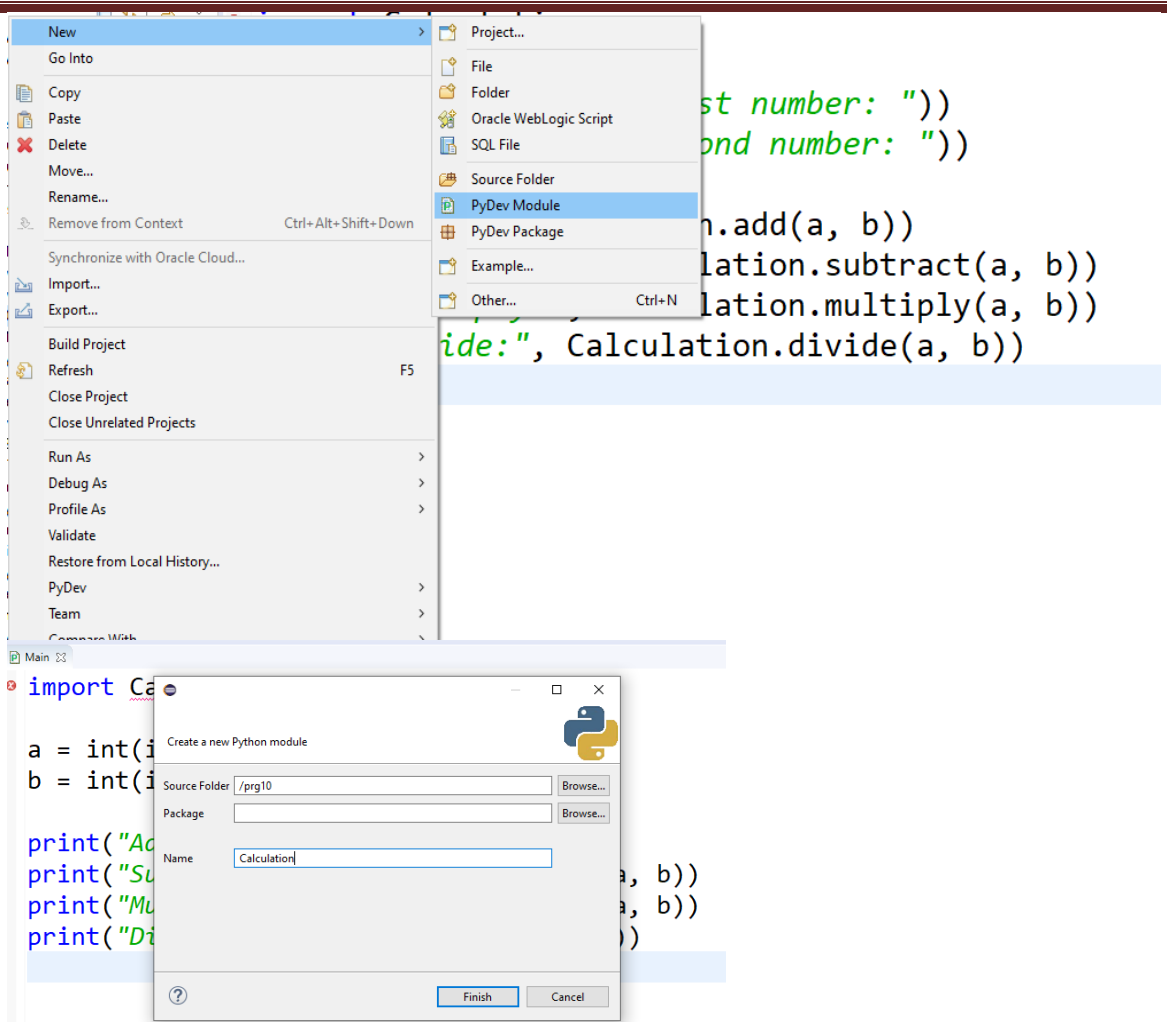


Create Main.py and type the code as shown below

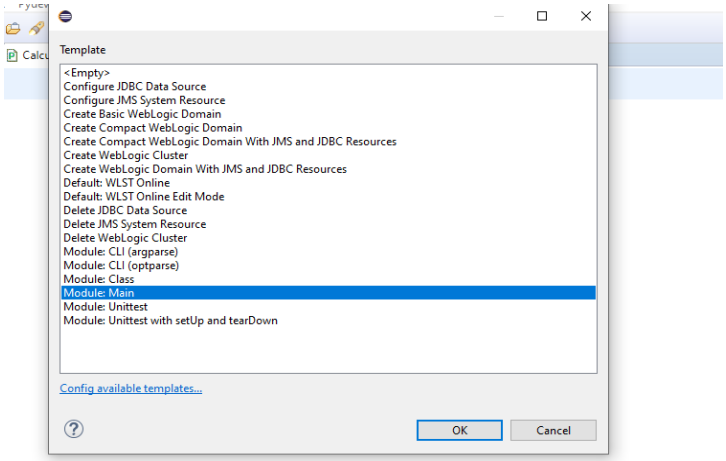
Main

```python
import Calculation

a = 15
b = 5

c = 100
d = 10

e = 7
f = 0

print("Operations with", a, "and", b)
print("Addition:", Calculation.add(a, b))
print("Subtraction:", Calculation.subtract(a, b))
print("Multiplication:", Calculation.multiply(a, b))
print("Division:", Calculation.divide(a, b))
print("-" * 30)

print("Operations with", c, "and", d)
print("Addition:", Calculation.add(c, d))
print("Subtraction:", Calculation.subtract(c, d))
print("Multiplication:", Calculation.multiply(c, d))
print("Division:", Calculation.divide(c, d))
print("-" * 30)

print("Operations with", e, "and", f)
print("Addition:", Calculation.add(e, f))
print("Subtraction:", Calculation.subtract(e, f))
print("Multiplication:", Calculation.multiply(e, f))
print("Division:", Calculation.divide(e, f))
```

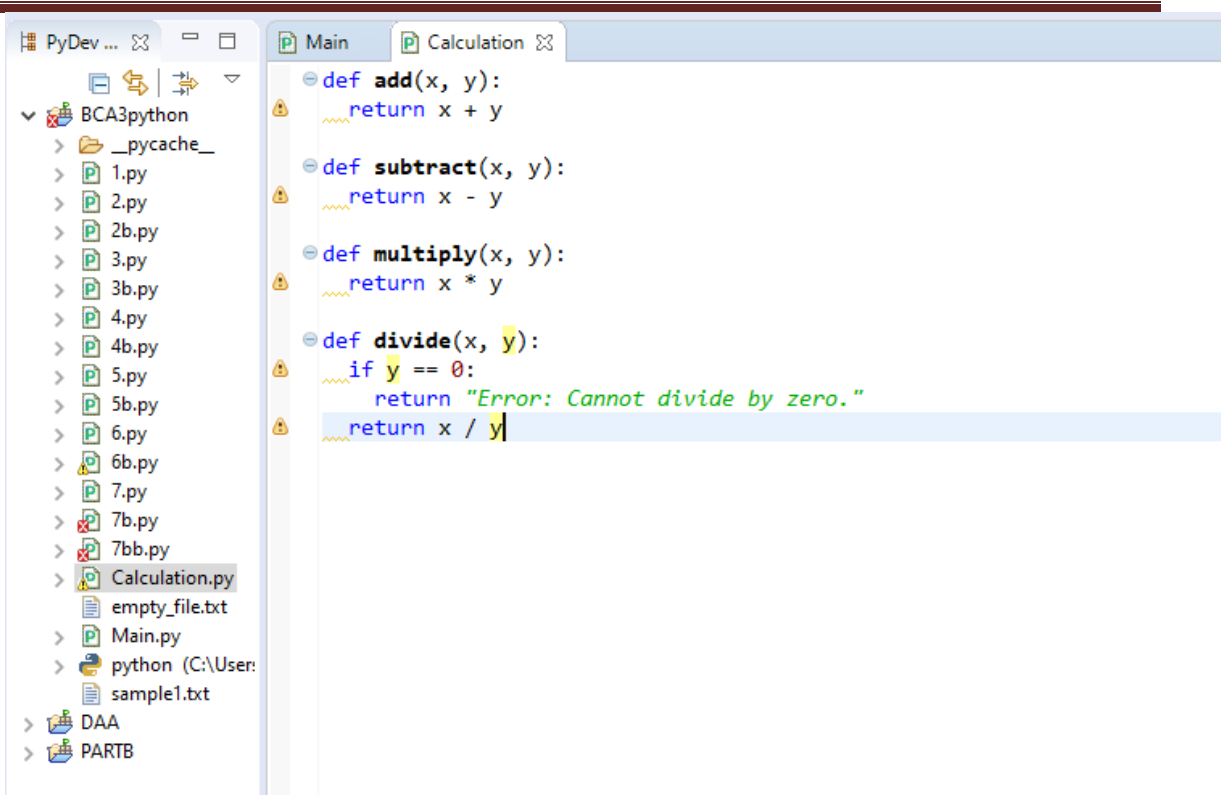
Now create Calculation.py module right click on prg10 as shown below and create the Calculation.py module in the same folderof Main.py file means both files should be in same folder.



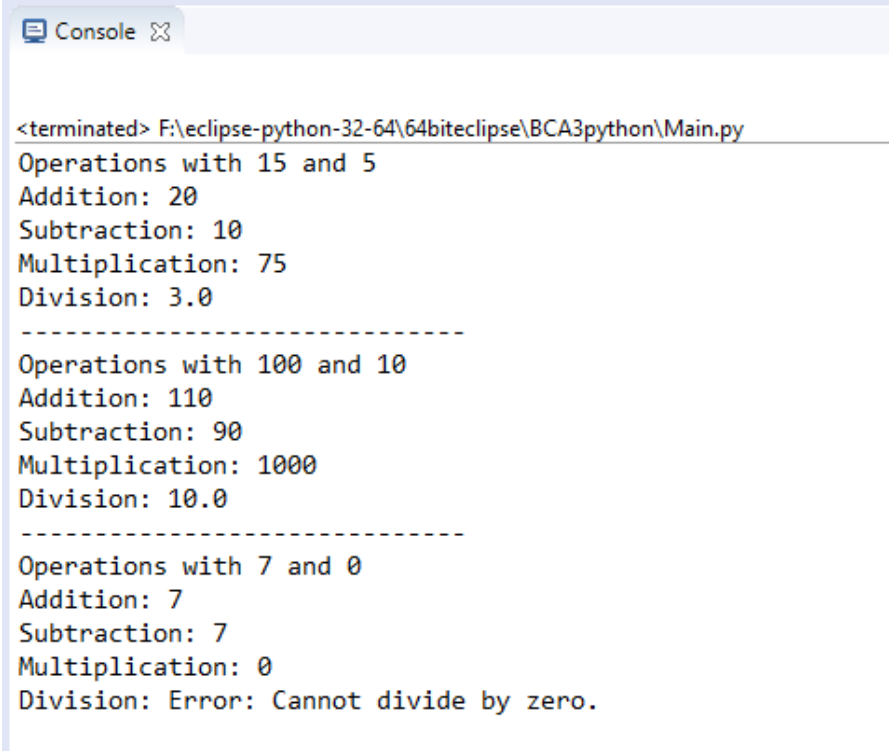
Select Module Main- click ok



Now type the code of Calculation.py



Now Run the Main.py file



End of Part –A

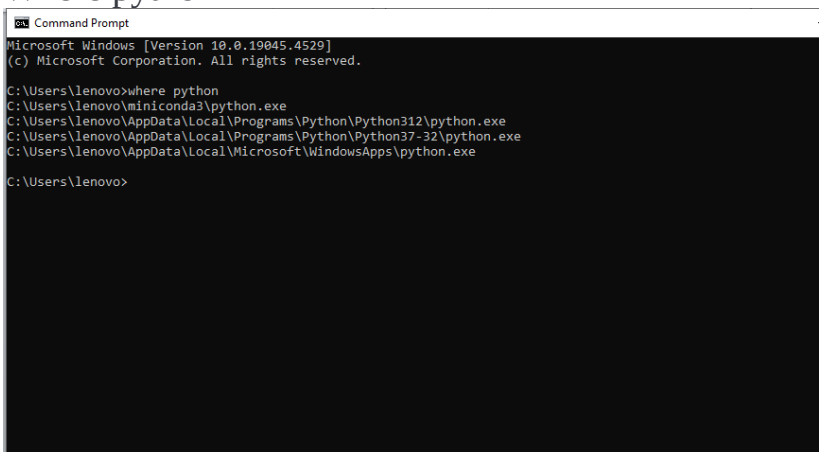
Part-B

1. Python program to demonstrate modification of an existing table data from MySQL database

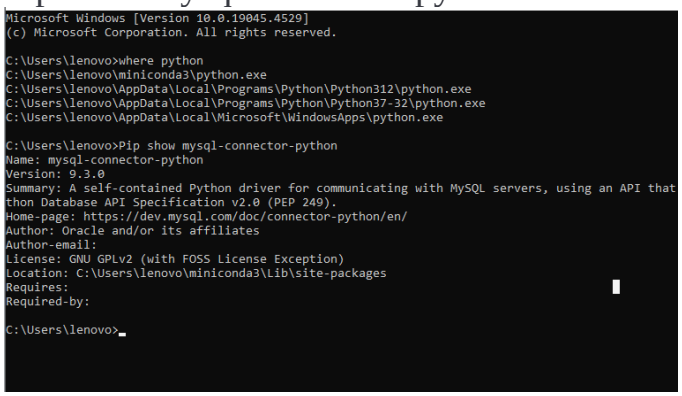
- Python programs using the mysql.connectormodule to:
- Create a MySQL database and table, insert 3 records, and display them.
- Modify an existing record in the table and show the updated data.

Prerequisites:

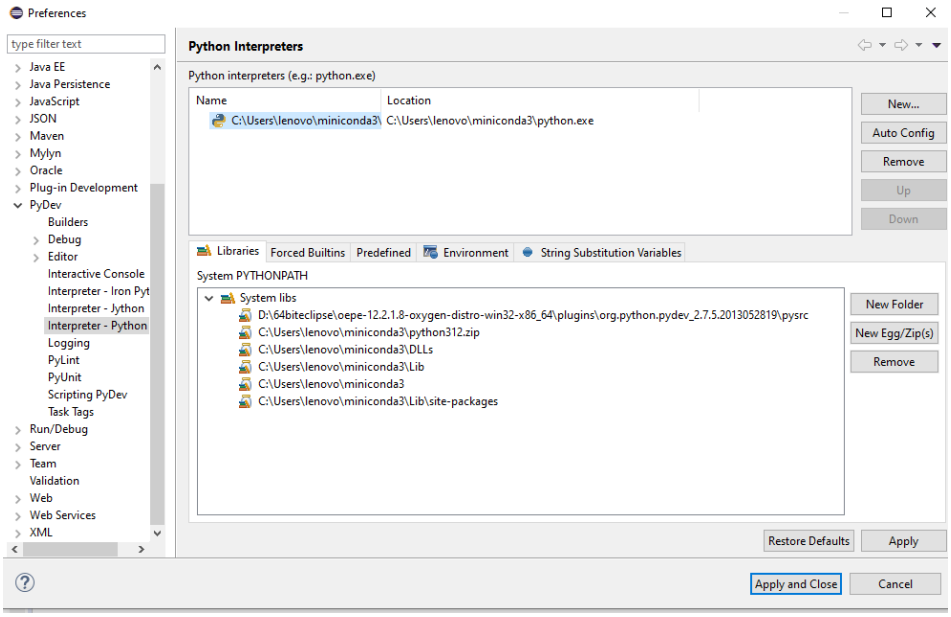
- Install the MySQL connector module if not already installed:
  pip install mysql-connector-python
- Install XAMPP and start apache and MySQL server
- Check the python path from command line
  Where python



- Then check where the module mysql-connector-python is installed
  Pip show mysql-connector-python



- Look at the location field in the output. it should match the python path you are using in eclipse
- Check the python path inside eclipse(PyDev)



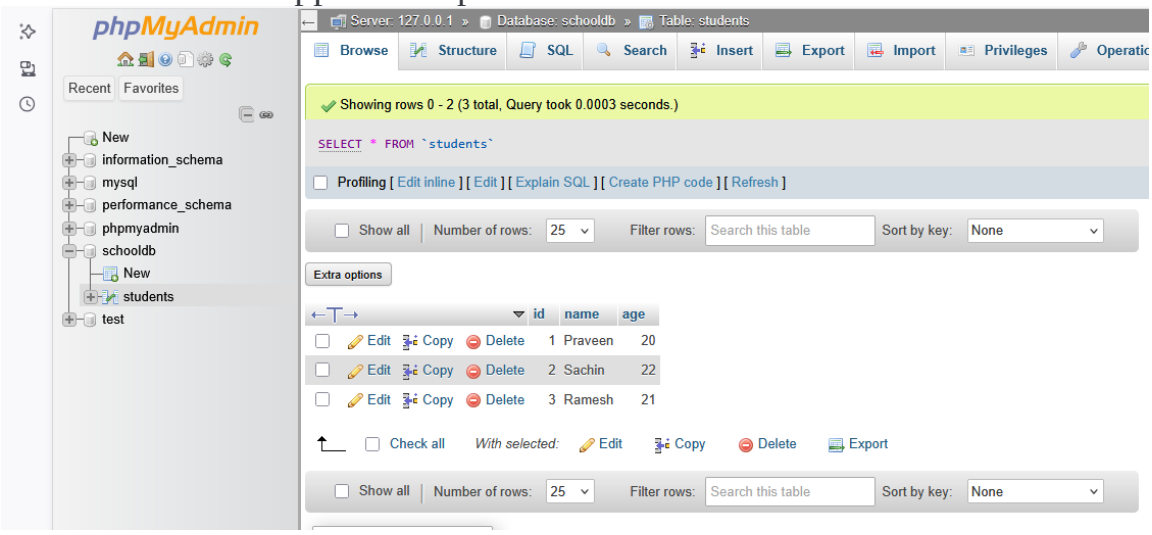
- Lastly from the above picture the Python interpreter is in location **C:\Users\lenovo\miniconda3\python.exe**
- And the above picture Mysql module is also in location**C:\Users\lenovo\miniconda3\Lib\site-packages**
- Make sure your MySQL server is running, and you know theusername and password host="localhost",user="root",password=""

Program 1: Create Database, Table, Insert & ShowRecords

```
import mysql.connector
conn = mysql.connector.connect(host="localhost",user="root",password="")
cursor = conn.cursor()
# Create database
cursor.execute("CREATE DATABASE IF NOT EXISTS SchoolDB")
cursor.execute("USE SchoolDB")
# Create table
cursor.execute("""
CREATE TABLE IF NOT EXISTS Students (id INT PRIMARY KEY, name VARCHAR(50),age INT)""")
# Insert records
cursor.execute("INSERT INTO Students VALUES (1, 'Praveen', 20)")
cursor.execute("INSERT INTO Students VALUES (2, 'Sachin', 22)")
cursor.execute("INSERT INTO Students VALUES (3, 'Ramesh', 21)")
conn.commit()
# Show records
cursor.execute("SELECT * FROM Students")
rows = cursor.fetchall()

print("Initial Records:")
for row in rows:
    print(row)
cursor.close()
conn.close()
```

```
<terminated> D:\64biteclipse\oepe-12.2.1.8-oxygen-distro-win32-x86_64\datbase\2.py
Initial Records:
(1, 'Praveen', 20)
(2, 'Sachin', 22)
(3, 'Ramesh', 21)
```

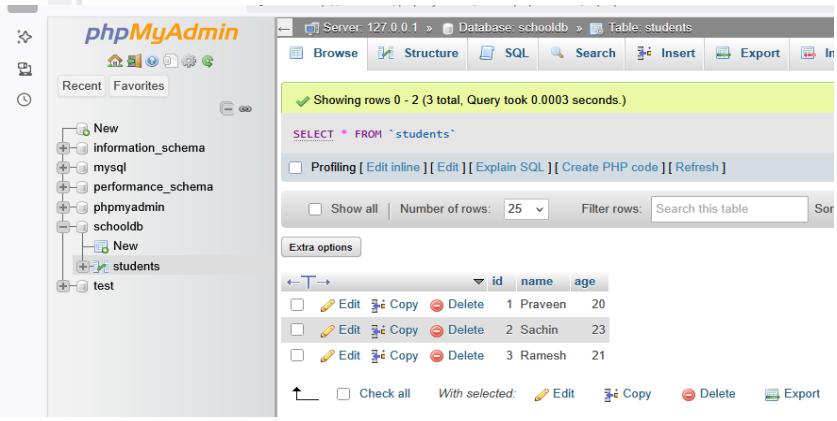
You can check in Xampp control panel as show below



Program 2: Modify a Record & ShowUpdated Table

```python
import mysql.connector
conn = mysql.connector.connect(host="Localhost",user="root",password="",database="SchoolDB")
cursor = conn.cursor()
print("Before Modification:")
cursor.execute("SELECT * FROM Students")
for row in cursor.fetchall():
    print(row)
# Modify student age where id=2
cursor.execute("UPDATE Students SET age = 23 WHERE id = 2")
conn.commit()
# Show updated records
print("\nAfter Modification:")
cursor.execute("SELECT * FROM Students")
for row in cursor.fetchall():
    print(row)
cursor.close()
conn.close()
```

```
<terminated> D:\64biteclipse\oepe-12.2.1.8-oxygen-distro-win32-x86_64\datbase\3.py
Before Modification:
(1, 'Praveen', 20)
(2, 'Sachin', 22)
(3, 'Ramesh', 21)

After Modification:
(1, 'Praveen', 20)
(2, 'Sachin', 23)
(3, 'Ramesh', 21)
```



2. **Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.**

2b

```python
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius**2

    def perimeter(self):
        return 2 * math.pi * self.radius
circle1 = Circle(5)
print("Circle 1 with radius:", circle1.radius)
print("Area:", circle1.area())
print("Perimeter:", circle1.perimeter())
print("-" * 30)

# Input 2: A circle with a larger radius
circle2 = Circle(12.5)
print("Circle 2 with radius:", circle2.radius)
print("Area:", circle2.area())
print("Perimeter:", circle2.perimeter())
print("-" * 30)

# Input 3: A circle with a radius of zero
circle3 = Circle(0)
print("Circle 3 with radius:", circle3.radius)
print("Area:", circle3.area())
print("Perimeter:", circle3.perimeter())
print("-" * 30)
```

Console

```
<terminated> F:\eclipse-python-32-64\64biteclipse\BCA3python\2b.py
Circle 1 with radius: 5
Area: 78.53981633974483
Perimeter: 31.41592653589793
------------------------------
Circle 2 with radius: 12.5
Area: 490.8738521234052
Perimeter: 78.53981633974483
------------------------------
Circle 3 with radius: 0
Area: 0.0
Perimeter: 0.0
------------------------------
```

3. **Python class named Rectangle constructed by a length and width and a method which will compute the area and perimeter of rectangle.**

Inherit a class Box that contains additional method volume. Override the perimeter method to compute perimeter of a Box.

3b

```python
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width
    def area(self):
        return self.length * self.width
    def perimeter(self):
        return 2 * (self.length + self.width)
class Box(Rectangle):
    def __init__(self, length, width, height):
        super().__init__(length, width)
        self.height = height
    def volume(self):
        return self.length * self.width * self.height
    def perimeter(self):
        return 4 * (self.length + self.width + self.height)
rect1 = Rectangle(10, 5)
print("Rectangle 1:")
print("Length:", rect1.length, "Width:", rect1.width)
print("Area:", rect1.area())
print("Perimeter:", rect1.perimeter())
print("-" * 30)
rect2 = Rectangle(7, 7)
print("Rectangle 2 (Square):")
print("Length:", rect2.length, "Width:", rect2.width)
print("Area:", rect2.area())
print("Perimeter:", rect2.perimeter())
print("-" * 30)
box1 = Box(10, 5, 3)
print("Box 1:")
print("Length:", box1.length, "Width:", box1.width, "Height:", box1.height)
print("Area of base (from parent class):", box1.area())
print("Perimeter of all edges (overridden method):", box1.perimeter())
print("Volume:", box1.volume())
print("-" * 30)
box2 = Box(4, 6, 8)
print("Box 2:")
print("Length:", box2.length, "Width:", box2.width, "Height:", box2.height)
print("Area of base:", box2.area())
print("Perimeter of all edges:", box2.perimeter())
print("Volume:", box2.volume())
```

```
Console

<terminated> F:\eclipse-python-32-64\64biteclipse\BCA3python\3b.py
Rectangle 1:
Length: 10 Width: 5
Area: 50
Perimeter: 30
------------------------------
Rectangle 2 (Square):
Length: 7 Width: 7
Area: 49
Perimeter: 28
------------------------------
Box 1:
Length: 10 Width: 5 Height: 3
Area of base (from parent class): 50
Perimeter of all edges (overridden method): 72
Volume: 150
------------------------------
Box 2:
Length: 4 Width: 6 Height: 8
Area of base: 24
Perimeter of all edges: 72
Volume: 192
```

4. Python program to show use of Regular expressions with match(), search(), findall(), sub() and split().

```python
import re
string1 = "Python is a popular programming language."
string2 = "I use Python for data analysis."
string3 = "The numbers are 123 and 456, and also 789."
string4 = "Hello, world! How are you today?"
print("--- Using re.match() ---")
match_result1 = re.match(r"Python", string1)
if match_result1:
    print("Match found at start of string:", match_result1.group())
else:
    print("No match found for 'Python' at start of string 1.")

match_result2 = re.match(r"use", string2)
if match_result2:
    print("Match found at start of string:", match_result2.group())
else:
    print("No match found for 'use' at start of string 2.")

print("-" * 30)
print("--- Using re.search() ---")
search_result1 = re.search(r"language", string1)
if search_result1:
    print("Search found:", search_result1.group())
else:
    print("Search did not find 'language' in string 1.")

search_result2 = re.search(r"C++", string2)
if search_result2:
    print("Search found:", search_result2.group())
```

```
else:
    print("Search did not find 'C++' in string 2.")

print("-" * 30)
print("--- Using re.findall() ---")
findall_result1 = re.findall(r"\d+", string3)
print("Original string:", string3)
print("Found all numbers:", findall_result1)

findall_result2 = re.findall(r"p\w+", string1)
print("Original string:", string1)
print("Found all words starting with 'p':", findall_result2)

print("-" * 30)
print("--- Using re.sub() ---")
sub_result1 = re.sub(r"Python", "Java", string1)
print("Original string:", string1)
print("After replacing 'Python':", sub_result1)

sub_result2 = re.sub(r"is", "was", string2)
print("Original string:", string2)
print("After replacing 'is':", sub_result2)

print("-" * 30)
print("--- Using re.split() ---")
split_result1 = re.split(r" ", string4)
print("Original string:", string4)
print("Split by space:", split_result1)

split_result2 = re.split(r"[ ,!]+", string4)
print("Original string:", string4)
print("Split by multiple delimiters:", split_result2)
```

```
<terminated> F:\eclipse-python-32-64\64biteclipse\BCA3python\4b.py
--- Using re.match() ---
Match found at start of string: Python
No match found for 'use' at start of string 2.
------------------------------
--- Using re.search() ---
Search found: language
Search did not find 'C++' in string 2.
------------------------------
--- Using re.findall() ---
Original string: The numbers are 123 and 456, and also 789.
Found all numbers: ['123', '456', '789']
Original string: Python is a popular programming language.
Found all words starting with 'p': ['popular', 'programming']
------------------------------
--- Using re.sub() ---
Original string: Python is a popular programming language.
After replacing 'Python': Java is a popular programming language.
Original string: I use Python for data analysis.
After replacing 'is': I use Python for data analyswas.
------------------------------
--- Using re.split() ---
Original string: Hello, world! How are you today?
Split by space: ['Hello,', 'world!', 'How', 'are', 'you', 'today?']
Original string: Hello, world! How are you today?
Split by multiple delimiters: ['Hello', 'world', 'How', 'are', 'you', 'today?']
```

5. Python program to demonstrate Exception handling using 'try', 'except', 'finally' and 'else block.

```python
# --- Program to demonstrate exception handling ---

def demonstrate_exception_handling(input_value):
    print("Testing with input: '{}'".format(input_value))
    try:
        # The code that might raise an exception is placed here.
        number = int(input_value)
        result = 100 / number
    except ValueError:
        # This block runs if a ValueError occurs (e.g., non-numeric input).
        print("Except block: Invalid input. Please enter a number.")
    except ZeroDivisionError:
        # This block runs if a ZeroDivisionError occurs (e.g., dividing by zero).
        print("Except block: Cannot divide by zero.")
    except Exception as e:
        # A general except block to catch any other unexpected errors.
        print("Except block: An unexpected error occurred: {}".format(e))
    else:
        # This block runs ONLY if the code in the try block completes without any exceptions.
        print("Else block: The result of the division is: {}".format(result))
    finally:
        # This block ALWAYS runs, regardless of whether an exception occurred or not.
        print("Finally block: Execution of this block is complete.")
    print("-" * 35)
demonstrate_exception_handling("10")
demonstrate_exception_handling("abc")
demonstrate_exception_handling("0")
demonstrate_exception_handling("50")
demonstrate_exception_handling("")
```

```
<terminated> F:\eclipse-python-32-64\64biteclipse\BCA3python\5b.py
Testing with input: '10'
Else block: The result of the division is: 10.0
Finally block: Execution of this block is complete.
-----------------------------------
Testing with input: 'abc'
Except block: Invalid input. Please enter a number.
Finally block: Execution of this block is complete.
-----------------------------------
Testing with input: '0'
Except block: Cannot divide by zero.
Finally block: Execution of this block is complete.
-----------------------------------
Testing with input: '50'
Else block: The result of the division is: 2.0
Finally block: Execution of this block is complete.
-----------------------------------
Testing with input: ''
Except block: Invalid input. Please enter a number.
Finally block: Execution of this block is complete.
-----------------------------------
```

6. Python program to read a file line by line store it into an array.

```python
def read_file_to_list(filename):
    try:
        with open(filename, 'r') as file:
            lines_list = [line.strip() for line in file]
        return lines_list
    except FileNotFoundError:
        print("Error: The file '{}' was not found.".format(filename))
        return None
    except Exception as e:
        print("An unexpected error occurred: {}".format(e))
        return None
file_name_1 = "sample1.txt"
print("--- Reading File: {} ---".format(file_name_1))
result1 = read_file_to_list(file_name_1)
if result1 is not None:
    print("The content stored in the list is:")
    print(result1)
print("=" * 30)
file_name_2 = "empty_file.txt"
print("--- Reading File: {} ---".format(file_name_2))
result2 = read_file_to_list(file_name_2)
if result2 is not None:
    print("The content stored in the list is:")
    print(result2)
print("=" * 30)
file_name_3 = "non_existent_file.txt"
print("--- Reading File: {} ---".format(file_name_3))
result3 = read_file_to_list(file_name_3)
if result3 is not None:
    print("The content stored in the list is:")
    print(result3)
print("=" * 30)
```

```
<terminated> F:\eclipse-python-32-64\64biteclipse\BCA3python\6b.py
--- Reading File: sample1.txt ---
The content stored in the list is:
['This is my home work.', 'This is the second line.', '', 'This is the Darbar college.']
==============================
--- Reading File: empty_file.txt ---
The content stored in the list is:
[]
==============================
--- Reading File: non_existent_file.txt ---
Error: The file 'non_existent_file.txt' was not found.
==============================
```

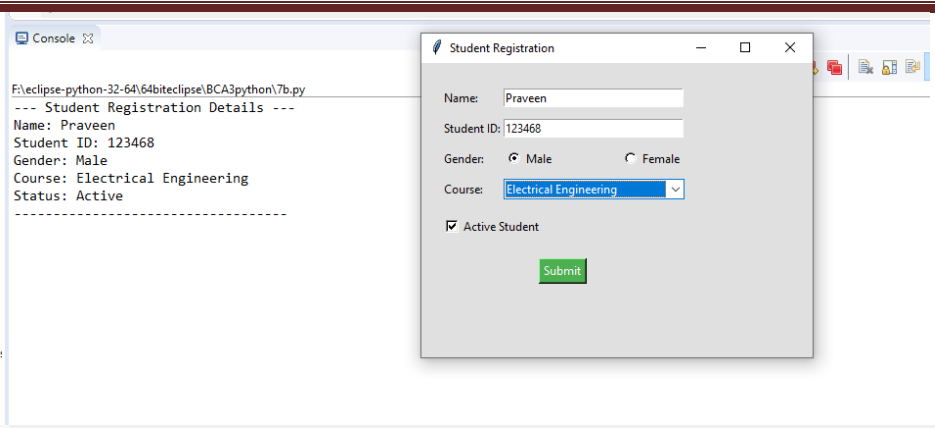
7. Python GUI program to design Student Registration Form using any 5 widgets.Widgets Used:

```python
import tkinter as tk
from tkinter import ttk
def submit_form():
    name = name_entry.get()
```

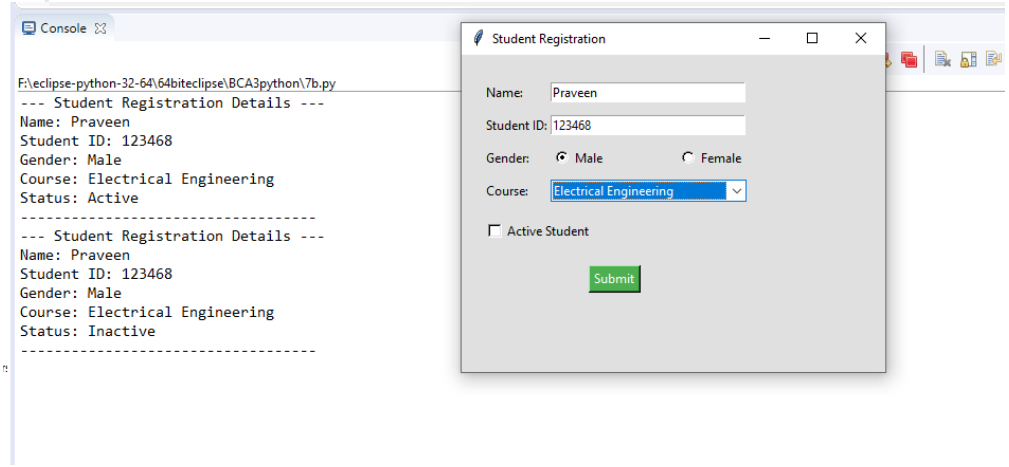
```
    student_id = id_entry.get()
    gender = gender_var.get()
    course = course_combo.get()
    status = "Active" if status_var.get() else "Inactive"
    print("--- Student Registration Details ---")
    print("Name: {}".format(name))
    print("Student ID: {}".format(student_id))
    print("Gender: {}".format(gender))
    print("Course: {}".format(course))
    print("Status: {}".format(status))
    print("-" * 35)
root = tk.Tk()
root.title("Student Registration")
root.geometry("400x300")
root.configure(bg="#e0e0e0")
main_frame = tk.Frame(root, padx=20, pady=20, bg="#e0e0e0")
main_frame.pack(fill="both", expand=True)
name_label = tk.Label(main_frame, text="Name:", bg="#e0e0e0")
name_label.grid(row=0, column=0, sticky="w", pady=5)
name_entry = tk.Entry(main_frame, width=30)
name_entry.grid(row=0, column=1, pady=5)
id_label = tk.Label(main_frame, text="Student ID:", bg="#e0e0e0")
id_label.grid(row=1, column=0, sticky="w", pady=5)
id_entry = tk.Entry(main_frame, width=30)
id_entry.grid(row=1, column=1, pady=5)
gender_label = tk.Label(main_frame, text="Gender:", bg="#e0e0e0")
gender_label.grid(row=2, column=0, sticky="w", pady=5)
gender_var = tk.StringVar(value="Male")  # Default value
gender_male_rb = tk.Radiobutton(main_frame, text="Male",
variable=gender_var, value="Male", bg="#e0e0e0")
gender_female_rb = tk.Radiobutton(main_frame, text="Female",
variable=gender_var, value="Female", bg="#e0e0e0")
gender_male_rb.grid(row=2, column=1, sticky="w")
gender_female_rb.grid(row=2, column=1, sticky="e")
course_label = tk.Label(main_frame, text="Course:", bg="#e0e0e0")
course_label.grid(row=3, column=0, sticky="w", pady=5)
course_options = ["Computer Science", "Electrical Engineering", "Mechanical
Engineering", "Physics"]
course_combo = ttk.Combobox(main_frame, values=course_options,
state="readonly", width=27)
course_combo.grid(row=3, column=1, pady=5)
course_combo.set(course_options[0])  # Set default value
status_var = tk.BooleanVar(value=True) # Default value
status_check = tk.Checkbutton(main_frame, text="Active Student",
variable=status_var, bg="#e0e0e0")
status_check.grid(row=4, columnspan=2, sticky="w", pady=10)
submit_button = tk.Button(main_frame, text="Submit", command=submit_form,
bg="#4CAF50", fg="white")
submit_button.grid(row=5, columnspan=2, pady=10)
root.mainloop()
```

```python
import tkinter as tk
from tkinter import ttk
def submit_form():
    name = name_entry.get()
    student_id = id_entry.get()
    gender = gender_var.get()
    course = course_combo.get()
    status = "Active" if status_var.get() else "Inactive"
    print("--- Student Registration Details ---")
    print("Name: {}".format(name))
    print("Student ID: {}".format(student_id))
    print("Gender: {}".format(gender))
    print("Course: {}".format(course))
    print("Status: {}".format(status))
    print("-" * 35)
root = tk.Tk()
root.title("Student Registration")
root.geometry("400x300")
root.configure(bg="#e0e0e0")
main_frame = tk.Frame(root, padx=20, pady=20, bg="#e0e0e0")
main_frame.pack(fill="both", expand=True)
name_label = tk.Label(main_frame, text="Name:", bg="#e0e0e0")
name_label.grid(row=0, column=0, sticky="w", pady=5)
name_entry = tk.Entry(main_frame, width=30)
name_entry.grid(row=0, column=1, pady=5)
id_label = tk.Label(main_frame, text="Student ID:", bg="#e0e0e0")
id_label.grid(row=1, column=0, sticky="w", pady=5)
id_entry = tk.Entry(main_frame, width=30)
id_entry.grid(row=1, column=1, pady=5)
gender_label = tk.Label(main_frame, text="Gender:", bg="#e0e0e0")
gender_label.grid(row=2, column=0, sticky="w", pady=5)
gender_var = tk.StringVar(value="Male")  # Default value
```

```python
gender_male_rb = tk.Radiobutton(main_frame, text="Male", variable=gender_var, value="Male", bg="#e0e0e0")
gender_female_rb = tk.Radiobutton(main_frame, text="Female", variable=gender_var, value="Female", bg="#e0e0e0")
gender_male_rb.grid(row=2, column=1, sticky="w")
gender_female_rb.grid(row=2, column=1, sticky="e")
course_label = tk.Label(main_frame, text="Course:", bg="#e0e0e0")
course_label.grid(row=3, column=0, sticky="w", pady=5)
course_options = ["Computer Science", "Electrical Engineering", "Mechanical Engineering", "Physics"]
course_combo = ttk.Combobox(main_frame, values=course_options, state="readonly", width=27)
course_combo.grid(row=3, column=1, pady=5)
course_combo.set(course_options[0])  # Set default value
status_var = tk.BooleanVar(value=True) # Default value
status_check = tk.Checkbutton(main_frame, text="Active Student", variable=status_var, bg="#e0e0e0")
status_check.grid(row=4, columnspan=2, sticky="w", pady=10)
submit_button = tk.Button(main_frame, text="Submit", command=submit_form, bg="#4CAF50", fg="white")
submit_button.grid(row=5, columnspan=2, pady=10)
root.mainloop()
```


Console
F:\eclipse-python-32-64\64biteclipse\BCA3python\7b.py
--- Student Registration Details ---
Name: Praveen
Student ID: 123468
Gender: Male
Course: Electrical Engineering
Status: Active

Student Registration
Name: Praveen
Student ID: 123468
Gender: Male Female
Course: Electrical Engineering
Active Student
Submit


Console
F:\eclipse-python-32-64\64biteclipse\BCA3python\7b.py
--- Student Registration Details ---
Name: Praveen
Student ID: 123468
Gender: Male
Course: Electrical Engineering
Status: Active

--- Student Registration Details ---
Name: Praveen
Student ID: 123468
Gender: Male
Course: Electrical Engineering
Status: Inactive

Student Registration
Name: Praveen
Student ID: 123468
Gender: Male Female
Course: Electrical Engineering
Active Student
Submit

END