# COURSE OUTCOME 1

**DATE: 26/09/2024**

1. Familiarizing Integrated Development Environment (IDE), Code Analysis Tools

An integrated development environment (IDE) refers to a software application that offers computer programmers with extensive software development abilities. IDEs most often consist of a source code editor, build automation tools, and a debugger. Most modern IDEs have intelligent code completion. An IDE enables programmers to combine the different aspects of writing a computer program and increase programmer productivity by introducing features like editing source code, building executable, and debugging. IDEs are usually more feature-rich and include tools for debugging, building and deploying code. An IDE typically includes:

- A source code editor
- A compiler or interpreter
- An integrated debugger
- A graphical user interface (GUI)

A code editor is a text editor program designed specifically for editing source code. It typically includes features that help in code development, such as syntax highlighting, code completion, and debugging. The main difference between an IDE and a code editor is that an IDE has a graphical user interface (GUI) while a code editor does not. An IDE also has features such as code completion, syntax highlighting, and debugging, which are not found in a code editor. Code editors are generally simpler than IDEs, as they do not include many other IDE components. As such, code editors are typically used by experienced developers who prefer to configure their development environment manually. Some IDEs are given below:

1. IDLE

IDLE (Integrated Development and Learning Environment) is a default editor that accompanies Python. This IDE is suitable for beginner-level developers. The IDLE tool can be used on Mac OS, Windows, and Linux. The most notable features of IDLE include:

- Ability to search for multiple files
- Interactive interpreter with syntax highlighting, and error and i/o messages
- Smart indenting, along with basic text editor features
- A very capable debugger
- A great Python IDE for Windows

2. PyCharm

PyCharm is a widely used Python IDE created by JetBrains This IDE is suitable for professional developers and facilitates the development of large Python projects
The most notable features of PyCharm include:

- Support for JavaScript, CSS, and TypeScript
- Smart code navigation
- Quick and safe code refactoring
- Support features like accessing databases directly from the IDE

3. Visual Studio Code

Visual Studio Code (VS Code) is an open-source (and free) IDE created by Microsoft. It finds great use in Python development. VS Code is lightweight and comes with powerful features that only some of the paid IDEs offer. The most notable features of Visual Studio Code include Git integration and Code debugging within the editor.

4. Sublime Text 3

Sublime Text is a very popular code editor. It supports many languages, including Python. It is highly customizable and also offers fast development speeds and reliability. The most notable features of Sublime Text 3 include:

- Syntax highlighting
- Custom user commands for using the IDE
- Efficient project directory management
- It supports additional packages for the web and scientific Python development

5. Atom

Atom is an open-source code editor by GitHub and supports Python development. Atom is similar to Sublime Text and provides almost the same features emphasis on speed and usability. The most notable features of Atom include:

- Support for a large number of plugins
- Smart autocompletion
- Supports custom commands for the user to interact with the editor
- Support for cross-platform development

6. Jupyter

Jupyter is widely used in the field of data science. It is easy to use, interactive and allows live code sharing and visualization. The most notable features of Jupyter include:

- Supports for the numerical calculations and machine learning workflow
- Combine code, text, and images for greater user experience
- Intergeneration of data science libraries like NumPy, Pandas, and Matplotlib

7. Spyder

Spyder is an open-source IDE most commonly used for scientific development. Spyder comes with Anaconda distribution, which is popular for data science and machine learning. The most notable features of Spyder include:

- Support for automatic code completion and splitting
- Supports plotting different types of charts and data manipulation
- Integration of data science libraries like NumPy, Pandas, and Matplotlib

Code Analysis Tools

Source code analysis tools, also known as Static Application Security Testing (SAST) Tools, can help analyse source code or compiled versions of code to help find security flaws. SAST tools can be added into IDE. Such tools can help to detect issues during software development. Static code analysis techniques are used to identify potential problems in code before it is deployed, allowing developers to make changes and improve the quality of the software. Three techniques include syntax analysis, data and control flow analysis, and security analysis. SonarQube (Community Edition) is an open source static + dynamic code analysis platform developed by SonarSource for continuous inspection of code quality to perform fully automated code reviews / analysis to detect code smells, bugs, performance enhancements and security vulnerabilities.

**DATE:8-10-2024**

2.Display future leap years from current year to a final year entered by user.

**PROGRAM:**

```
a=int(input("Enter the Current year:\n"))
b=int(input("Enter the final Year:\n"))

print("Future leap year from ",a,"to",b,"are:")
for i in range(a,b+1):
        if i%4==0 and (i%4==0 or i%400==0):
                print(i)
```

**OUTPUT:**

```
Enter the Current year:
2000
Enter the final Year:
2020
Future leap year from  2000 to 2020 are:
2000
2004
2008
2012
2016
2020
```

**DATE:8-10-2024**

3. List comprehensions:

   (a) Generate positive list of numbers from a given list of integers

   (b) Square of N numbers

   (c) Form a list of vowels selected from a given word

   (d) List ordinal value of each element of a word (Hint: use ord() to get ordinal values)

**PROGRAM (a)**

```
l=[int(i) for i in input("Enter list of integers : ").split()]
p=[i for i in l if i>=0]
print("Positive Integers : ",p)
```

**OUTPUT**

```
Enter list of integers : 2 4 -8 -9
Positive Integers : [2, 4]

Enter List of integers : -6 -3 9 1
Positive Integers : [9, 1]
```

**PROGRAM (b)**

```
l=[int(i) for i in input("Enter List : ").split()]
l1=[i*i for i in l]
print(l1)
```

**OUTPUT**

Enter List : 1 3 5
[1, 9, 25]

Enter List :  6 7 8
[ 36, 49, 64]

**PROGRAM (c)**

```
word=input("Enter word : ")
vowels = "aeiouAEIOU"
vowel_list = [i for i in word if i in vowels]
print("Vowels in ",word," : ",vowel_list)
```

**OUTPUT**

Enter word : Midhun
Vowels in Programming : [ ‘i’,‘u’]

Enter word : Hello
Vowels in Hello : [‘e’, ‘o’]

**PROGRAM (d)**

```
word=input("Enter a word : ")
ordinal_values = [ord(char) for char in word]
print(f"The ordinal values of the characters are : {ordinal_values}")
```

**OUTPUT**

Enter a word : Python
The ordinal values of the characters are : [80, 121, 116, 104, 111, 110]

Enter word : Hello
The ordinal values of the characters are : [72, 101, 108, 108, 111]

**DATE:8-10-2024**

4.Count the occurrences of each word in a line of text.

**PROGRAM:**

```
a= input("Enter a line of text: ")
words = a.split()
word_count = {}

for word in words:
    word = word.strip('.,!?";:')
    if word in word_count:
        word_count[word] += 1
    else:
        word_count[word] = 1


print(word_count)
```

**OUTPUT:**

```
Enter a line of text: word count is are is
{'word': 1, 'count': 1, 'is': 2, 'are': 1}
```

**DATE:8-10-2024**

5.Prompt the user for a list of integers. For all values greater than 100, store ‘over’ instead.

**PROGRAM:**

```
numbers = input("Enter a list of integers separated by spaces: ").split()

list = ['over' if int(num) > 100 else num for num in numbers]

print("NEW list:", list)
```

**OUTPUT:**

```
Enter a list of integers separated by spaces: 12 38 109 23 98 1002 6
NEW list: ['12', '38', 'over', '23', '98', 'over', '6']
```

**DATE:8-10-2024**

6.Store a list of first names. Count the occurrences of ‘a’ within the list

**PROGRAM:**

```
names = input("Enter the names:")
count= sum(name.lower().count('a') for name in names)
print("Total occurrences of 'a':", count)
```

**OUTPUT:**

```
Enter the names:MIDHUN ASWIN CHRISTINA
Total occurrences of 'a': 2
```

**DATE:8-10-2024**

7.Enter 2 lists of integers. Check (a) Whether list are of same length (b) whether list sums to same value (c) whether any value occur in both

**PROGRAM:**

```
list1 = list(map(int, input("Enter the first list of Integers:").split()))
list2 = list(map(int, input("Enter the second list of Integers:").split()))

a = len(list1) == len(list2)
b=sum(list1) == sum(list2)
c=[value for value in list1 if value in list2]


print("Are the lists of the same length?",a)
print("Do the lists sum to the same value?",b)
print("Common values in both lists:", c)
```

**OUTPUT:**

```
Enter the first list of Integers:2 6 7 8
Enter the second list of Integers:6 7 8 9
Are the lists of the same length? True
Do the lists sum to the same value? False
Common values in both lists: [6, 7, 8]
```

**DATE:8-10-2024**

8 Get a string from an input string where all occurrences of first character replaced with ‘$’, except first character.

**PROGRAM:**

```
a=input("Enter a string: ")
if a:
   first_char = a[0]
   newstring = first_char + a[1:].replace(first_char, '$')
else:
   newstring = ""


print("Modified string:", newstring)
```

**OUTPUT:**

```
Enter a string: AMAL KRISHNA
Modified string: AM$L KRISHN$
```

**DATE:10/10/2024**

9.Create a string from given string where first and last characters exchanged.

**PROGRAM:**

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

if len(a) > 1:

    new_string = ''.join([a[-1]] + list(a)[1:-1] + [a[0]])
else:
    new_string = a

print("Updated string:", new_string)
```

**OUTPUT:**

```
Enter a string: MIDHUN
Updated string: NIDHUM
```

**DATE:10/10/2024**

10.Accept the radius from user and find area of circle.

**PROGRAM:**

```
r=float(input("Enter radius of circle: \n"))
area=3.14*r*r
print("Area of circle is: \n",area)
```

**OUTPUT:**

```
Enter radius of circle:
5
Area of circle is:
78.5
```

**DATE:10/10/2024**

11.Find biggest of 3 numbers entered.

**PROGRAM:**

```
a=int(input("Enter first number: \n"))
b=int(input("Enter second number: \n"))
c=int(input("Enter third number: \n"))
if a>b and a>c:
 print(a,"is greater than",b,c)
elif b>a and b>c:
 print(b,"is greater than",a,c)
elif c>a and c>b:
 print(c,"is greater than",a,b)
else:
 print("All are equal")
```

**OUTPUT:**

```
Enter first number:
5
Enter second number:
6
Enter third number:
9
9 is greater than 5 6
```

**DATE:10/10/2024**

12.Accept a file name from user and print extension of that.

**PROGRAM:**

```
file=input("Enter File Name : ")
temp=file.split(".")
ext= temp[-1]
if len(temp) > 1:
   print("Extension : ",ext)
else :
   print("")
```

**OUTPUT:**

```
Enter File Name : img.jpg
Extension :  jpg

Enter File Name : py.txt
Extension :  txt
```

**DATE:10/10/2024**

13. Create a list of colors from comma-separated color names entered by user. Display first and last colors.

**PROGRAM:**

```
list=[i for i in input("enter the colors:")].split(",")
print(“First colour entered is:”,list[0])
print(“Last colour entered is:”,list[-1])
```

**OUTPUT:**

```
enter the colors: red,blue,green
First colour entered is:red
Last colour entered is:green

enter the colors: black,grey,orange
First colour entered is:black
Last colour entered is:orange
```

**DATE:15/10/2024**

14.Accept an integer n and compute n+nn+nnn(value of n should be less than 10).

**PROGRAM:**

```
n=input("Enter an input")
s=int(n)+int(n*2)+int(n*3)
print("n+nn+nnn=",s)
```

**OUTPUT:**

```
Enter an input:
8
n+nn+nnn= 984
```

**DATE:15/10/2024**

15.Print out all colors from color-list1 not contained in color-list2.

**PROGRAM:**

```
l1=[i for i in input("enter the colos in list 1:").split()]
l2=[i for i in input("enter the colos in list 2:").split()]
print(l1)
print(l2)
result=set(l1)-set(l2)
print("colors in list1 not in list2",result)
```

**OUTPUT:**

```
enter the colos in list 1: red green yellow
enter the colos in list 2:blue red black
['red', 'green', 'yellow']
['blue', 'red', 'black']
colors in list1 not in list2 {'yellow', 'green'}
```

**DATE:15/10/2024**

16. Create a single string separated with space from two strings by swapping the character at position 1.

**PROGRAM:**

```
s1=input("Enter String 1 :")
s2=input("Enter String 2 :")
new1=s1[0]+s2[1]+s1[2:]
new2=s2[0]+s1[1]+s2[2:]
print("New String : “,new1,” ”,new2)
```

**OUTPUT:**

```
Enter String 1 : Midhun
Enter String 2 : Babu
New String : Madhun Bibu
```

```
Enter String 1 :Hey
Enter String 2 : Python
New String :Hyy Pethon
```

**DATE:22/10/2024**

17. Sort dictionary in ascending and descending order.

**PROGRAM:**

```
d={"apple":1 ,"pineapple":3,"kiwi":4,"banana":2}
print(" original dictionary is",d)
aresult=dict(sorted(d.items()))
print("dictionary in ascending order",aresult)
dresult=dict(sorted(d.items(),reverse=True))
print("dictionary in decending order",dresult)
```

**OUTPUT:**

```
original dictionary is {'apple': 1, 'pineapple': 3, 'kiwi': 4, 'banana': 2}
dictionary in ascending order {'apple': 1, 'banana': 2, 'kiwi': 4, 'pineapple': 3}
dictionary in decending order {'pineapple': 3, 'kiwi': 4, 'banana': 2, 'apple': 1}
```

**DATE:22/10/2024**

18.Merge two dictionaries.

**PROGRAM:**

```
d1={"name":"midhun","age":21,"gender":"M"}
d2={"course":"MCA","semester":"s1"}
print("dictionary 1",d1)
print("dictionary 2"d2)
d1.update(d2)
print(d1)
```

**OUTPUT:**

```
{'name': 'midhun', 'age': 21, 'gender': 'M'}
{'course': 'MCA', 'semester': 's1'}
{'name': 'anagha', 'age': 21, 'gender': 'f', 'course': 'MCA', 'semester': 's1'}
```

**DATE:22/10/2024**

19. Find gcd of 2 numbers.

**PROGRAM:**

```
import math
x=int(input("Enter the first number:"))
y=int(input("Enter the second number:"))
print("Gcd of 2 nummbers",math.gcd(x,y))
```

**OUTPUT:**

```
Enter the first number:42
Enter the second number:7
Gcd of 2 nummbers 7
```

**DATE:22/10/2024**

**20 From a list of integers, create a list removing even numbers.**

**PROGRAM:**

```
ol=[int(i) for i in input("Enter the numbers: \n").split()]
nl=[i for i in ol if i%2!=0]
print("New list =",nl)
```

**OUTPUT:**

```
Enter the numbers:
2 4 5 7 6 13 27 8
New list = [5, 7, 13, 27]
```

## COURSE OUTCOME 2

**DATE:22/10/2024**

1.Program to find the factorial of a number

**PROGRAM:**

```
num=int(input("Enter a number:\n"))
factorial = 1
for i in range(1, num + 1):
    factorial *= i
print(f"The factorial of {num} is {factorial}")
```

**OUTPUT:**

```
Enter a number:
6
The factorial of 6 is 720
```

**DATE:22/10/2024**

2.Generate Fibonacci series of N terms

**PROGRAM:**

```
n = int(input("Enter the limit: "))
a = 0
b = 1
for i in range(n):
    print(a)
    c = a + b
    a = b
    b = c
```

**OUTPUT:**

```
Enter the limit: 7
0
1
1
2
3
5
8
```

**DATE: 24/10/2024**

3.Find the sum of all items in a list.

**PROGRAM:**

```
li=[int(i) for i in input("Enter the list elements").split()]
sum_li=sum(li)
print(“SUM OF LIST:”,sum_li)
```

**OUTPUT:**

```
Enter the list elements2 3 4
SUM OF LIST:9

Enter the list elements 10 5 4 2
SUM OF LIST:21
```

**DATE: 24/10/2024**

4. Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.

**PROGRAM**

```
import math

def is_all_digits_even(num):
    while num > 0:
        digit = num % 10
        if digit % 2 != 0:
            return False
        num //= 10
    return True


def even_digit_perfect_squares(start, end):
    even_digit_squares = []
    for num in range(start, end + 1):
        if num >= 1000 and num <= 9999 and is_all_digits_even(num):
            root = int(math.sqrt(num))
            if root * root == num:
                even_digit_squares.append(num)
    return even_digit_squares
```

```
start = 1000

end = 9999


even_digit_squares = even_digit_perfect_squares(start, end)

print(even_digit_squares)
```

**OUTPUT**

```
[4624, 6084, 6400, 8464]
```

**DATE:24/10/2024**

5. Display the given pyramid with step number accepted from user. Eg: N=4

1

2 4

3 6 9

4 8 12 16

**PROGRAM:**

```
n=int(input("enter a limit "))
for i in range (1,n+1):
        for j in range(1,i+1):
                print(i*j,end=' ')
        print()
```

**OUTPUT:**

enter a limit 4

1

2 4

3 6 9

4 8 12 16

**DATE: 24/10/2024**

6. Count the number of characters (character frequency) in a string.

**PROGRAM:**

```
def character_frequency(s):
    frequency = {}
    for char in s:
        if char in frequency:
            frequency[char] += 1
        else:
            frequency[char] = 1
    return frequency

a = input("Enter a String:")
result = character_frequency(a)
print(result)
```

**OUTPUT:**

```
Enter a String:midhun babu
{'m': 1, 'i': 1, 'd': 1, 'h': 1, 'u': 2, 'n': 1, ' ': 1, 'b': 2, 'a': 1}
```

**DATE: 24/10/2024**

7. Add ‘ing’ at the end of a given string. If it already ends with ‘ing’, then add ‘ly’.

**PROGRAM:**

```
a=input("Enter a string :")
b=a[-3:]
if b=='ing':
    print(a+'ly')
else:
    print(a+'ing')
```

**OUTPUT:**

```
Enter a string : Swimming
Swimmingily
```

```
Enter a string : Swim
Swiming
```

**DATE: 29/10/2024**

8. Accept a list of words and return length of longest word.

**PROGRAM:**

```
s=[i for i in input("Enter some words :").split()]
print(len(max(s, key=len)))
```

**OUTPUT:**

```
Enter some words :Midhun Babu
6

Enter some words : Masters in Computer Application
11
```

**DATE: 29/10/2024**

9. Construct following pattern using nested loop

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

**PROGRAM:**

```
n = int(input("Enter the limit: "))
for i in range(n):
    for j in range(i + 1):
        print("*", end=" ")
    print()
for i in range(n):
    for j in range(n-i-1):
        print("*", end=" ")
    print()
```

**OUTPUT:**

Enter the limit : 5

```
*

* *

* * *

* * * *

* * * * *

* * * *

* * *

* *

*
```

**DATE: 29/10/2024**

10. Generate all factors of a number.

**PROGRAM:**

```
def factors(m):
    for i in range(1,a+1):
        if a%i == 0:
            print(i)

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

**OUTPUT:**

```
Enter a number : 4
1
2
4
```

**DATE:29/10/2024**

11.Write lambda functions to find area of square, rectangle and triangle.

**PROGRAM:**

```
area1=lambda a:a*a

area2=lambda l,b:l*b

area3=lambda b,h:0.5*b*h


s=int(input("enter the side of the square"))

print("area of a square is",area1(s))


l=int(input("enter the length of a rectangle"))

b=int(input("enter the breadth of a rectangle  "))

print("area of a square is",area2(l,b))


h=int(input("enter the height of a triangle"))

w=int(input("enter the breadth of a triangle  "))

print("area of a square is",area3(h,w))
```

**OUTPUT:**

enter the side of the square5

area of a square is 25

enter the length of a rectangle3

enter the breadth of a rectangle  4

area of a square is 12

enter the height of a triangle3

enter the breadth of a triangle  4

area of a square is 6.0

enter the side of the square6

area of a square is 36

enter the length of a rectangle6

enter the breadth of a rectangle  3

area of a square is 18

enter the height of a triangle2

enter the breadth of a triangle  4

area of a square is 4.0

# COURSE OUTCOME 3

**DATE: 13/11/2024**

1.Work with built-in packages.

**BUILT-IN PACKAGES IN PYTHON**

Python comes with a comprehensive standard library that includes a wide range of built-in packages and modules. These modules provide functionality for tasks ranging from file I/O to web development. Here are some commonly used built-in packages in Python:

1. os :Operating system interface, provides a way of using operating system-dependent functionality like reading or writing to the file system.

```
import os
```

2. sys : Provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

```
import sys
```

3. math : Mathematical functions such as basic arithmetic operations, logarithms, trigonometric functions, etc.

```
import math
```

4. datetime : Date and time handling.

```
import datetime
```

5. json : JSON encoder and decoder.

```
import json
```

6. urllib : URL handling modules, including parsing, quoting, and fetching.

```
from urllib import request, parse
```

7. random : Generate pseudo-random numbers.

```
import random
```

8. re : Regular expression operations.

```
import re
```

9. collections : Implements specialized container datatypes.

```
from collections import Counter, defaultdict
```

10. sqlite3 : SQLite database interface.

```
import sqlite3
```

11. csv : CSV file reading and writing.

```
import csv
```

12. gzip : Support for gzip files.

```
import gzip
```

13. socket : Low-level networking interface.

```
import socket
```

14. argparse : Command-line argument parsing.

```
import argparse
```

**DATE:20/11/2024**

2.create a package graphics with modules rectangle, circle and sub-package 3D-graphics with modules cuboid and sphere. Include methods to find area and perimeter of respective figures in each module. Write programs that finds area and perimeter of figures by different importing statements. (Include selective import of modules and import * statements)

Path- home/Midhun/Python/graphics/threeD

**PROGRAM:**

**main.py**

```
# main.py includes importing modules and calling functions to calculate area and perimeter of shapes.

from graphics import rectangle,circle
from graphics.threed import cuboid,sphere

l=int(input("Enter the length of rectangle:\t"))
b=int(input("Enter the width of rectangle:\t"))
print("Area of rectangle is",rectangle.area(l,b))
print("Perimeter of rectangle is",rectangle.perimeter(l,b))

r=int(input("Enter the radius of circle:"))
print("Area of circle",circle.area(r))
print("Perimeter of rectangle",circle.perimeter(r))

l=int(input("Enter the length of cuboid:"))
b=int(input("Enter the width of cuboid:"))
h=int(input("Enter the height of cuboid:"))
print("Surface area of cuboid",cuboid.surfacearea(l,b,h))
print("Volume of cuboid",cuboid.volume(l,b,h))

r=int(input("Enter the radius of sphere:"))
print("Surface area of sphere is",sphere.surfacearea(r))
print("Volume of sphere is",sphere.volume(r))
```

**graphics**

#graphics is a package that includes modules rectangle.py and circle.py

**circle.py**

```
import math
def area(r):
    return(math.pi*r**2)

def perimeter(r):
    return(2*math.pi*r)
```

**rectangle.py**

```
def area(l,b):
    return(l*b)

def perimeter(l,b):
    return(2*(l+b))
```

**threeD**

#threeD is a sub-package inside of graphics that includes modules cuboid.py and sphere.py

**sphere.py**

```
import math

def surfacearea(r):
    return(4*math.pi*r**2)

def volume(r):
    return((4/3)*math.pi*r**3)
```

**cuboid.py**

```
def surfacearea(l,b,h):
    return(2*(l*b+b*h+h*l))
def volume(l,b,h):
    return(l*b*h)
```

**OUTPUT:**

```
Enter length of rectangle:3
Enter width of rectangle:4
Area of a rectangle is 12
Perimeter of a rectangle is 14
Enter radius of circle:5
Area of a circle is 78.57142857142857
Perimeter of a circle is 31.42857142857143
Enter length of cuboid:3
Enter width of cuboid:4
Enter height of cuboid:5
Surface area of a cuboid is 94
Volume of a cuboid is 60
Enter radius of sphere:6
Surface area of a sphere is 452.5714285714286
Volume of sphere is 905.1428571428572
```

# COURSE OUTCOME 4

**DATE:20/11/2024**

1.Create Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area.

**PROGRAM:**

```
class Rectangle:
    def __init__(self,length,breadth):
    self.l=length
    self.b=breadth

    def area(self):
    return self.l * self.b

    def perimeter(self):
    return 2 * (self.l + self.b)

    def compare_area(self, other):

            if self.area() > other.area():
            return "First rectangle is larger."

            elif self.area() < other.area():

            return "Second rectangle is larger."

            else:
            return "Both rectangles have the same area."
```

```
rect=Rectangle(int(input("Enter length of the First rectangle:")),int(input("Enter width of the First rectangle:")))

rect1=Rectangle(int(input("Enter length of the Second rectangle:")),int(input("Enter width of the Second rectangle:")))

print("\nDetails of the first rectangle:")
print("Area of rectangle is:", rect.area())
print("Perimeter of rectangle is:", rect.perimeter())

print("\nDetails of the Second rectangle:")
print("Area of rectangle is:", rect1.area())
print("Perimeter of rectangle is:", rect1.perimeter())
print("\nComparison of Two rectangle:" )

result=rect.compare_area(rect1)
print(result)
```

**OUTPUT:**

```
Enter length of the First rectangle:5
Enter width of the First rectangle:2
Enter length of the Second rectangle:6
Enter width of the Second rectangle:1

Details of the first rectangle:
Area of rectangle is: 10
Perimeter of rectangle is: 14

Details of the Second rectangle:
Area of rectangle is: 6
Perimeter of rectangle is: 14

Comparison of Two rectangle:
First rectangle is larger.
```

**DATE:5/12/2024**

2.Create a Bank account with members account number, name, type of account and balance. Write constructor and methods to deposit at the bank and withdraw an amount from the bank.

**PROGRAM:**

```
class Bank:
    def __init__(self,acc_no,name,acc_type,balance):
    self.ano=acc_no
    self.anname=name
    self.atype=acc_type
    self.abalance=balance

    def deposit(self,amt):
    if amt>0:
            self.abalance=self.abalance+amt
            print("Amount deposited successfully")
    else:
            print("Invalid amount")

    def withdraw(self,amt):
    if amt > self.abalance:
            print("Insufficient balance.")
    else:
            print("Amount succesfully withdrawn")
            self.abalance -= amount

    def displaydetails(self):
    print("Name:", self.anname)
    print("Account No:", self.ano)
    print("Account Type:", self.atype)
    print("Balance Amount:", self.abalance)

account=int(input("Enter account number:"))
name=input("Enter name:")
acc_type=input("Enter account type(Fixed or savings):")
```

```
balance=int(input("Enter current savings:"))
account = Bank(account,name,acc_type, balance)
while(True):
    print("\n.......MENU.......\n1.Deposit\n2.Withdraw\n3.Display Account Info\n4.Exit")
    choice = int(input("Enter your choice: "))

    if choice == 1:
            amount = int(input("Enter amount to deposit: "))
            account.deposit(amount)
    elif choice == 2:
            amount = int(input("Enter amount to withdraw: "))
            account.withdraw(amount)
    elif choice == 3:
            account.displaydetails()
    elif choice == 4:
            print("Exiting the program.")
            break
    else:
            print("Invalid choice. Please try again.")
```

**OUTPUT:**

```
Enter account number:21345632156
Enter name:Midhun Babu
Enter account type(Fixed or savings):savings
Enter current savings:10000

.......MENU.......
1.Deposit
2.Withdraw
3.Display Account Info
4.Exit
Enter your choice: 1
Enter amount to deposit: 750
Amount deposited successfully
```

.....MENU.....
1.Deposit
2.Withdraw
3.Display Account Info
4.Exit
Enter your choice: 3
Name: Midhun Babu
Account No: 21345632156
Account Type: savings
Balance Amount: 10750

.....MENU.....
1.Deposit
2.Withdraw
3.Display Account Info
4.Exit
Enter your choice: 2
Enter amount to withdraw: 350
Amount succesfully withdrawn

.....MENU.....
1.Deposit
2.Withdraw
3.Display Account Info
4.Exit
Enter your choice: 3
Name: Midhun Babu
Account No: 21345632156
Account Type: savings
Balance Amount: 10400
.....MENU.....
1.Deposit
2.Withdraw
3.Display Account Info
4.Exit
Enter your choice: 4
Exiting the program.

**DATE:5/12/2024**

3.Create a class Rectangle with private attributes length and width. Overload '<' operator to compare the area of 2 rectangles.

**PROGRAM:**

```
class Rectangle:
    def __init__(self,length,width):
    self.length=length
    self.width=width
    def area(self):
    return self.length*self.width
    def __lt__(self,other):
    return self.area() < other.area()

rect1=Rectangle(int(input("Enter length of the First rectangle:")),int(input("Enter width of the First rectangle:")))
rect2=Rectangle(int(input("Enter length of the Second rectangle:")),int(input("Enter width of the Second rectangle:")))
print("Area of rectangle 1 is:", rect1.area())
print("Area of rectangle 2 is:", rect2.area())

if rect1<rect2:
    print("Area of rectangle 1 is lesser than area of rectangle 2.")
elif rect1>rect2:
    print("Area of rectangle 1 is greater than area of rectangle 2.")
else:
    print("Both rectangles have same area.")
```

**OUTPUT:**

```
Enter length of the First rectangle:7
Enter width of the First rectangle:4
Enter length of the Second rectangle:5
Enter width of the Second rectangle:3
Area of rectangle 1 is: 28
Area of rectangle 2 is: 15
Area of rectangle 1 is greater than area of rectangle 2.
```

**DATE:5/12/2024**

4.Create a class Time with private attributes hour, minute and second. Overload '+' operator to find sum of 2 time.

**PROGRAM:**

```
class Time:
    def __init__(self, hour, minute, second):
    self.hour = hour
    self.minute = minute
    self.second = second

    def __add__(self, other):
    second = self.second + other.second
    minute = self.minute + other.minute + second // 60
    hour = self.hour + other.hour + minute // 60
    return Time(hour % 24, minute % 60, second % 60)

    def display(self):
    print(self.hour,"Hour",self.minute,"minute",self.second,"second")

a,b,c=map(int,input("Enter first time in format of hh mm ss:").split())
x,y,z=map(int,input("Enter second time in format of hh mm ss:").split())
time1 = Time(a, b, c)
time2 = Time(x, y,z)
result = time1 + time2
result.display()
```

**OUTPUT:**

```
Enter first time in format of hh mm ss:18 26 22
Enter second time in format of hh mm ss:2 14 44
20 Hour 41 minute 6 second

Enter first time in format of hh mm ss:16 5 22
Enter second time in format of hh mm ss:9 55 11
2 Hour 0 minute 33 second
```

**DATE:5/12/2024**

5.Create a class Publisher (name). Derive class Book from Publisher with attributes title and author. Derive class Python from Book with attributes price and no_of_pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding.

**PROGRAM:**

```
class Publisher:
    def __init__(self,name):
    self.name=name
    def display():
    pass


class Book(Publisher):
    def __init__(self,name,title,author):
    super().__init__(name)
    self.title=title
    self.author=author
    def display():
    pass


class Python(Book):
    def __init__(self,name,title,author,price,no_of_pages):
    super().__init__(name,title,author)
    self.price=price
    self.no_of_pages = no_of_pages

    def display(self):
    print("Book details:")
    print("Title of Book :",self.title)
    print("Name of author:",self.author)
    print("Name of publisher:",self.name)
    print("Price of boooks:",self.price)
    print("Number of pages:",self.no_of_pages)
```

```
book1=Python(input("Enter the title of book:"),input("Enter name of
Author:"),input("Enter name of publisher:"),int(input("Enter price of
Book:")),int(input("Number of pages in book:")))

book1.display()
```

**OUTPUT:**

```
Enter the title of book:Effective Python
Enter name of Author: Brett Slatkin
Enter name of publisher:Addison-Wesley Professional
Enter price of Book:3795
Number of pages in book:480
Book details:
Title of Book :  Brett Slatkin
Name of author: Addison-Wesley Professional
Name of publisher: Effective Python
Price of boooks: 3795
Number of pages: 480
```

# COURSE OUTCOME 5

**DATE:20/11/2024**

1.Write a Python program to read a file line by line and store it into a list.

**PROGRAM:**

```
f= open("midhun.txt", "r")
l=[i.split() for i in open("midhun.txt")]
print(l)
f.close()
```

**midhun.txt**

Hello! Welcome to midhun.txt
Sachin is a cricketer.
Good Day!

**OUTPUT:**

[[‘Hello’,’welcome’,’to’,’midhun.txt’]],[‘Sachin’,‘is’,‘a’,’cricketer’]
[‘Good’,’Day.’]

**DATE:5/12/2024**

2.Python program to copy odd lines of one file to other

**PROGRAM:**

```
with open("midhun.txt", "r") as x, open("demo.txt","w") as y:
    line_number = 0
    for line in x:
    if line_number % 2 == 0:
            y.write(line)
    line_number += 1
```

**file.txt**

Hello! World
My name is Midhun Babu
Dream High

**OUTPUT**

Hello! World
Dream High

**DATE:5/12/2024**

3.Write a Python program to read each row from a given csv file and print a list of strings.

**PROGRAM:**

```
import csv
with open("student.csv",mode="r")as file:
    csvr=csv.reader(file)
    for row in csvr:

    print(row)
```

**student.csv**

```
roll,name,age,course
101,Jude,21,,mca
102,Midhun,21,mca
103,Rasim,21,mca
104,Aswin,24,mca
105,Amal,21,mca
```

**OUTPUT**

```
['roll', 'name', 'age', 'course']
[‘101’,‘Jude’,‘21’,‘mca’]
[‘102’,‘Midhun’,‘21’,‘mca’]
[‘103’,‘Rasim’,‘21’,‘mca’]
[‘104’,‘Aswin’,‘24’,‘mca’]
[‘105’,‘Amal’,‘21’,‘mca’]
```

**DATE:5/12/2024**

4.Write a Python program to read specific columns of a given CSV file and print the content of the columns.

**PROGRAM:**

```
import csv

CI=int(input("Enter coloumn:"))

with open("student.csv",mode="r") as file:
    csvr=csv.reader(file)

    rows=list(csvr)

    for row in rows:
    print(row)

    for row in rows:
    if len(row)>CI:
            print(row[CI])
```

**student.csv**

```
 rollno,name,age,course
 101,Midhun,21,MCA
 102,Rasim,21,Physics
 103,Mariyam,21,MCA
 104,Naaji,21,MCA
 105,Anurag,21,BCA
```

**OUTPUT:**

```
Enter coloumn:1
[' rollno', 'name', 'age', 'course']
[' 101', 'Midhun', '21', 'MCA']
[' 102', 'Rasim', '21', 'Physics']
[' 103', 'Mariyam', '21', 'MCA']
[' 104', 'Naaji', '21', 'MCA']
[' 105', 'Anurag', '21', 'BCA']
name
Midhun
Rasim
Mariyam
Naaji
Anurag
```

**DATE:5/12/2024**

5.Write a Python program to write a Python dictionary to a csv file. After writing the CSV file read the CSV file and display the content.

**PROGRAM:**

```
import csv
data = {
    'Name': ['Midhun', 'Rasim', 'Mariyam'],
    'Age': [21, 23, 22],
    'City': ['Thrissur', 'Kannur', 'Aroor']
}


with open('output.csv', mode='w', newline='') as file:
    writer = csv.DictWriter(file, fieldnames=data.keys())
    writer.writeheader()
    for i in range(len(next(iter(data.values())))):
    row = {field: data[field][i] for field in data}
    writer.writerow(row)

print("Dictionary written to CSV file 'output.csv'.")


with open('output.csv', mode='r') as file:
    reader = csv.DictReader(file)
    for row in reader:
    print(row)
```

**OUTPUT:**

```
Dictionary written to CSV file 'output.csv'.
{'Name': 'Midhun', 'Age': '21', 'City': 'Thrissur'}
{'Name': 'Rasim', 'Age': '23', 'City': 'Kannur'}
{'Name': 'Mariyam', 'Age': '22', 'City': 'Aroor'}
```