# SREE NARAYANA GURUKULAM COLLEGE OF ENGINEERING

## KADAYIRUPPU, KOLENCHERY 682 311

**(Affiliated to APJ Abdul Kalam Technological University)**

**ACADEMIC YEAR 2021-22**

## 20 MCA 132 PROGRAMMING LABORATORY RECORD

***Submitted by***

**SREERAG T V**

**REG NO: SNG21MCA-2036**

*in partial fulfillment for the award of the degree in*

***MASTER OF COMPUTER APPLICATIONS***

# SREE NARAYANA GURUKULAM COLLEGE OF ENGINEERING

## KADAYIRUPPU, KOLENCHERY 682 311

**(Affiliated to APJ Abdul Kalam Technological University)**

## 20 MCA 132 PROGRAMMING LABORATORY RECORD

*Certified that this is a Bonafide record of practical work done by* ***SREERAG T V*** *to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree in Master of Computer Applications of Sree Narayana Gurukulam College of Engineering done during the Academic year 2021-22.*

Kadayiruppu

Date: …….  Course Instructor

Head of the Department

**Prof.Dr. SANDHYA R**

Submitted for University Practical Examination

**Reg No: SNG21MCA-2036 on**

External Examiner  Internal Examiner

| SL NO. | DATE | NAME OF EXPERIMENT | PAGE NO. | REMARK |
|---|---|---|---|---|
| I | CO1 | | | |
| 1 | 24/11/21 | Familiarizing Text Editor, IDE, Code Analysis Tools etc | 1 | |
| 2 | 24/11/21 | Leap Year | 3 | |
| 3 | 24/11/21 | List comprehensions | 4 | |
| 4 | 24/11/21 | occurrences of each word | 5 | |
| 5 | 24/11/21 | Prompt the user for a list of integers. | 6 | |
| 6 | 24/11/21 | Store a list of first names. | 7 | |
| 7 | 24/11/21 | Checking list are of same length,sums to same value,any value occur in both | 8 | |
| 8 | 24/11/21 | Get a string from an input string and replacing a character | 9 | |
| 9 | 24/11/21 | Create a string from given string where first and last characters exchanged. | 10 | |
| 10 | 24/11/21 | Accept the radius from user and find area of circle | 11 | |
| 11 | 29/11/21 | Find biggest of 3 numbers entered | 12 | |
| 12 | 29/11/21 | Accept a file name from user and print extension of that | 13 | |
| 13 | 29/11/21 | Create a list of colors,Display first and last colors. | 14 | |
| 14 | 29/11/21 | Accept an integer n and compute n+nn+nnn | 15 | |
| 15 | 29/11/21 | Print out all colors from color-list1 not contained in color-list2 | 16 | |
| 16 | 29/11/21 | Create a single string separated with space from two strings by swapping the character at position 1. | 17 | |
| 17 | 29/11/21 | Sort dictionary in ascending and descending order | 18 | |
| 18 | 29/11/21 | Merge two dictionaries | 19 | |

| | | | | |
|---|---|---|---|---|
| **19** | **29/11/21** | **Find gcd of 2 numbers.** | **20** | |
| **20** | **29/11/21** | **From a list of integers, create a list removing even numbers**. | **21** | |
| **II** | **CO2** | | | |
| **1** | **1/12/21** | **Program to find the factorial of a number** | **22** | |
| **2** | **1/12/21** | **Generate Fibonacci series of N terms** | **23** | |
| **3** | **1/12/21** | **Find the sum of all items in a list** | **24** | |
| **4** | **1/12/21** | **Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.** | **25** | |
| **5** | **1/12/21** | **Display the given pyramid with step number accepted from user** | **26** | |
| **6** | **1/12/21** | **Count the number of characters (character frequency) in a string** | **27** | |
| **7** | **8/12/21** | **Add 'ing' at the end of a given string. If it already ends with 'ing', then add 'ly'** | **28** | |
| **8** | **8/12/21** | **Accept a list of words and return length of longest word** | **29** | |
| **9** | **8/12/21** | **Construct pattern using nested loop** | **30** | |
| **10** | **8/12/21** | **Generate all factors of a number. def print_factors(x):** | **31** | |
| **11** | **8/12/21** | **Write lambda functions to find area of square, rectangle and triangle.** | **32** | |
| **III** | **CO3** | | | |
| **1** | **15/12/21** | **Work with built-in packages** | **33** | |
| **2** | **15/12/21** | **Creation of packages** | **39** | |
| **IV** | **CO4** | | | |
| **1** | **9/1/22** | **Compare two Rectangle objects by their area** | **41** | |

| | | | | |
|---|---|---|---|---|
| **2** | **9/1/22** | **Create a Bank account with members account number, name, type of account and balance.** | **42** | |
| **3** | **9/1/22** | **Overload '<' operator to compare the area of 2 rectangles.** | **43** | |
| **4** | **9/1/22** | **Overload '+' operator to find sum of 2 time** | **44** | |
| **5** | **9/1/22** | **Use base class constructor invocation and method overriding.** | **46** | |
| **V** | **CO5** | | | |
| **1** | **30/1/22** | **Write a Python program to read a file line by line and store it into a list.** | **47** | |
| **2** | **30/1/22** | **Python program to copy odd lines of one file to other** | **48** | |
| **3** | **30/1/22** | **Write a Python program to read each row from a given csv file and print a list of strings.** | **49** | |
| **4** | **30/1/22** | **Write a Python program to read specific columns of a given CSV file** | **50** | |
| **5** | **30/1/22** | **Write a Python program to write a Python dictionary to a csv file.** | **51** | |

# I. COURSE OUTCOME 1(CO1)

**PROGRAM NO: 1**

**DATE:24/11/2021**

**AIM: Familiarizing Text Editor, IDE, Code Analysis Tools etc // Use any IDE**

**IDE**

An IDE (or Integrated Development Environment) is a program dedicated to software development. As the name implies, IDEs integrate several tools specifically designed for software development. These tools usually include:

- An editor designed to handle code (with, for example, syntax highlighting and auto-completion)
- Build, execution, and debugging tools
- Some form of source control

Most IDEs support many different programming languages and contain many more features. They can, therefore, be large and take time to download and install. You may also need advanced knowledge to use them properly.

In contrast, a dedicated code editor can be as simple as a text editor with syntax highlighting and code formatting capabilities. Most good code editors can execute code and control a debugger. The very best ones interact with source control systems as well. Compared to an IDE, a good dedicated code editor is usually smaller and quicker, but often less feature rich.

**General Editors and IDEs with Python Support**

**1.Visual Studio**

Built by Microsoft, Visual Studio is a full-featured IDE, in many ways comparable to Eclipse. Built for Windows and Mac OS only, VS comes in both free (Community) and paid (Professional and Enterprise) versions. Visual Studio enables development for a variety of platforms and comes with its own marketplace for extensions.

Python Tools for Visual Studio (aka PTVS) enables Python coding in Visual Studio, as well as Intellisense for Python, debugging, and other tools.

Not to be confused with full Visual Studio, Visual Studio Code (aka VS Code) is a full-featured code editor available for Linux, Mac OS X, and Windows platforms. Small and light-weight, but full-featured, VS Code is open-source, extensible and configurable for almost any task. Like Atom, VS Code is built on Electron, so it has the same advantages and disadvantages that brings.

Installing Python support in VS Code is very accessible: the Marketplace is a quick button click away. Search for Python, click Install, and restart if necessary. VS Code will recognize your Python installation and libraries automatically.

**Pros:** Thanks to Electron, VS Code is available on every platform, surprisingly full-featured despite having a small footprint, and open-source.

**Cons:** Electron means VS Code is not a native app. Plus, some people may have principled reasons to not use Microsoft resources.

**2.Pycharm**

One of the best (and only) full-featured, dedicated IDEs for Python is PyCharm. Available in both paid (Professional) and free open-source (Community) editions, PyCharm installs quickly and easily on Windows, Mac OS X, and Linux platforms.

Out of the box, PyCharm supports Python development directly. You can just open a new file and start writing code. You can run and debug Python directly inside PyCharm, and it has support for source control and projects.

**Pros:** It's the de facto Python IDE environment, with tons of support and a supportive community. It edits, runs, and debugs Python out of the box.

**Cons:** PyCharm can be slow to load, and the default settings may need tweaking for existing projects.

***3.Thonny***

A recent addition to the Python IDE family, Thonny is billed as an IDE for beginners. Written and maintained by the Institute of Computer Science at the University of Tartu in Estonia, Thonny is available for all major platforms, with installation instructions on the site.

By default, Thonny installs with its own bundled version of Python, so you don't need to install anything else new. More experienced users may need to tweak this setting so already installed libraries are found and used.

**Pros:** You're a beginning Python user, and want an IDE that's ready to roll.

**Cons:** More experienced Python developers will find Thonny too basic for most uses, and the built-in interpreter is something to work around, not with. Plus, as a new tool, there may be issues you find which may not have immediate solutions.

**4.Spyder**

Spyder is an open-source Python IDE that's optimized for data science workflows. Spyder comes included with the Anaconda package manager distribution, so depending on your setup you may already have it installed on your machine.

What's interesting about Spyder is that it's target audience is data scientists using Python. You'll notice this throughout. For example, Spyder integrates well with common Python data science libraries like SciPy, NumPy, and Matplotlib.

Spyder features most of the "common IDE features" you might expect, such as a code editor with robust syntax highlighting, Python code completion, and even an integrated documentation browser.

A special feature that I haven't seen in other Python editing environments is Spyder's "variable explorer" that allows you to display data using a table-based layout right inside your IDE.

Overall, I'd say that Spyder feels more basic than other IDEs. I like to view it more as a special purpose tool rather than something I use as my primary editing environment every day. What is nice about this Python IDE is that it is available for free on Windows, macOS, and Linux and that it is fully open-source software.

**Pros:** You're a data scientist using the Anaconda Python distribution.

**Cons:** More experienced Python developers might find Spyder too basic to work with on a daily basis and instead opt for a more complete IDE or customized editor solution.

**PROGRAM NO: 2**

**DATE:24/11/2021**

**AIM: Display future leap years from current year to a final year entered by user**.

```
s=int(input("enter start year:"))
e=int(input("enter end year:"))
if(s<e):
   print("leap years are:",end=" ")
for i in range(s,e):
   if i%4==0 and i%100!=0:
      print(i, end=" ")
else:
    print(“Invalid”)
```

**OUTPUT:**

```
enter start year:2021
enter end year:2050
leap years are: 2024 2028 2032 2036 2040 2044 2048
```

**PROGRAM NO: 3**

**DATE:24/11/2021**

**AIM: List comprehensions:**

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

```
list =[-10,20,35,-67,70]
for i in list:
   if(i>0):
      print(i)
```

**OUTPUT:**

```
20
35
70
```

**(b) Square of N number**

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

**OUTPUT:**

```
Enter the limit:5
1
4
9
16
25
```

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

```
word =input("Enter the word :")
print("The original string is : "+word)
print("The vowels are :")
for i in word:
   if i in "aeiouAEIOU":
      print([i])
```

**OUTPUT**

```
Enter the word :sreerag
The original string is : sreerag
The vowels are :
['e']
['e']
['a']
```

**PROGRAM NO: 4**

**DATE:24/11/2021**

**AIM: Count the occurrences of each word in a line of text.**

```
str1 = input("Enter a string : ")
wordlist = str1.split()
count= []
for w in wordlist:
   count.append(wordlist.count(w))
print("count of the occurrence:" + str(list(zip(wordlist, count))))
```

**OUTPUT**

```
Enter a string : Python is a programming language python
count of the occurrence:[('Python', 1), ('is', 1), ('a', 1), ('programming', 1), ('language', 1), ('python',
1)]
```

**PROGRAM NO: 5**

**DATE:24/11/2021**

**AIM: Prompt the user for a list of integers. For all values greater than 100, store 'over' instead**

```
n=[]
s=int(input("Enter a limit:"))
print("Enter {s} values :")
for i in range(0,s):
   n.append(int(input()))
   print("\nThe list after assinging:\n")
for i in range(0,len(n)):
  if n[i]>=100:
     print("over!!")
  else:print(n[i])
```

**OUTPUT**

```
Enter a limit:2
Enter {s} values
24
199
The list after assinging:
24
over
```

**PROGRAM NO: 6**

**DATE:24/11/2021**

**AIM: Store a list of first names. Count the occurrences of ‘a’ within the list**

```
lst = ["a","b","c","a"]
occ = lst.count("a")
print("Occurrences of 'a' :",occ)
```

**OUTPUT**

```
Occurrences of 'a' : 2
```

**PROGRAM NO: 7**

**DATE:24/11/2021**

**AIM: 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**

```
lst=[1,3,5,7,9,11,34]
lst1=[5,13,45,7,20,65,1]
s=int(0)
c=int(0)
if len(lst)==len(lst1):
  print("Lists are of same length")
else:
  print("Lists have different length")

for i in range(0,len(lst) and len(lst1)):
  s=s+lst[i]
  c=c+lst1[i]
if(s==c):
  print("equal sum")
else:
  print("not same sum")

print("Elements that matched are:")
l=[]
for i in range(0,len(lst)):
  for j in range(0,len(lst1)):
    if lst[i]==lst1[j]:
       l.append(lst[i] and lst1[j])
    else:
      continue
print(l)
```

**OUTPUT**

```
Lists are of same length
not same sum
Elements that matched are:
[1, 5, 7]
```

**PROGRAM NO: 8**

**DATE:24/11/2021**

**AIM: Get a string from an input string where all occurrences of first character replaced with '$', except first character. [eg: onion -> oni$n]**

```
str = "onion"
char = str[0]
str = str.replace(char, '$')
str = char + str[1:]
print(str)
```

**OUTPUT**

oni$n

**PROGRAM NO: 9**

**DATE:24/11/2021**

**AIM: Create a string from given string where first and last characters exchanged.**
**[eg: python -> nythop]**

```
str = input("Enter a string :")
newstr = str[-1:] + str[1:-1] + str[:1]
print("New string :",newstr)
```

**OUTPUT**

```
Enter a string : python
New string :  nythop
```

**PROGRAM NO: 10**

**DATE:24/11/2021**

**AIM: Accept the radius from user and find area of circle.**

```
pi = 3.14
r = float(input("Enter the radius of circle :"))
area = pi*r**2
print("Area of circle :", area)
```

**OUTPUT**

```
Enter the radius of circle :2
Area of circle : 12.56
```

**PROGRAM NO: 11**

**DATE:29/11/2021**

**AIM: Find the biggest of three numbers entered**

```
a = int(input("Enter First No:"))
b = int(input("Enter Second No:"))
c = int(input("Enter Third No:"))

if(a > b and a>c):
   print(a,"is largest")
elif(b > c):
   print(b,"is largest")

elif(c > a):
   print(c,"is largest")
```

**OUTPUT**

```
Enter First No:4
Enter Second No:9
Enter Third No:1
9 is largest
```

**PROGRAM NO: 12**

**DATE:29/11/2021**

**AIM: Accept a file name from user and print extension of that.**

```
file =  input("Enter file name :")
f = file.split(".")
print("Extension of file is :",f[-1])
```

**OUTPUT**

```
Enter file name :sample.java
Extension of file is : java
```

**PROGRAM NO: 13**

**DATE:29/11/2021**

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

```
a=[]
for i in range(3):
   b=input("enter the color:")
   a.append(b)
print(a)
print(a[0])
print(a[2])
```

**OUTPUT**

```
enter the color:Black
enter the color:Blue
enter the color:White
['Black', 'Blue', 'White']
Black
White
```

**PROGRAM NO: 14**

**DATE:29/11/2021**

**AIM: Accept an integer n and compute n+nn+nnn**

```
n = int(input("Enter a number :"))
x = int("%s" % n)
y = int("%s%s" % (n,n))
z = int("%s%s%s" % (n,n,n))
print("n + nn + nnn :", x+y+z)
```

**OUTPUT**

```
Enter a number :5
n + nn + nnn : 615
```

**PROGRAM NO: 15**

**DATE:29/11/2021**

**AIM: Print out all colors from color-list1 not contained in color-list2.**

```
lst1 = set(["White", "Pink", "Red", "Blue"])
lst2 = set(["Red", "Green", "Pink"])
print(lst1.difference(lst2))
```

**OUTPUT**

```
{'Blue', 'White'}
```

**PROGRAM NO: 16**

**DATE:29/11/2021**

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

```
a = "Python"
b = "Java"
p1 = a[0]
p2 = b[0]
c = b[0] + a[1:len(a)]+" "+a[0] + b[1:len(b)]
print(c)
```

**OUTPUT**

Jython Pava

**PROGRAM NO: 17**

**DATE:29/11/2021**

**AIM: Sort dictionary in ascending and descending order.**

```
import operator
d = {1: 2, 3: 4, 4: 3, 2: 1, 0: 0}
print('Original dictionary : ',d)
sorted_d = sorted(d.items(), key=operator.itemgetter(1))
print('Dictionary in ascending order : ',sorted_d)
sorted_d = dict( sorted(d.items(), key=operator.itemgetter(1),reverse=True))
print('Dictionary in descending order : ',sorted_d)
```

**OUTPUT**

```
Original dictionary :  {1: 2, 3: 4, 4: 3, 2: 1, 0: 0}
Dictionary in ascending order :  [(0, 0), (2, 1), (1, 2), (4, 3), (3, 4)]
Dictionary in descending order :  {3: 4, 4: 3, 1: 2, 2: 1, 0: 0}
```

**PROGRAM NO: 18**

**DATE:29/11/2021**

**AIM: Merge two dictionaries**

```
d1 ={ 'a': 100, 'b': 200}
d2 ={'x' : 300, 'y': 200}
print ("Dict ionary 1 :", d1)
print ("Dictionary 2 : ", d2)
d =d1. copy ()
d.update (d2)
print ("Merged Dictionary : ", d)
```

**OUTPUT**

```
Dictionary 1 : {'a': 100, 'b': 200}
Dictionary 2 :  {'x': 300, 'y': 200}
Merged Dictionary :  {'a': 100, 'b': 200, 'x': 300, 'y': 200}
```

**PROGRAM NO: 19**

**DATE:29/11/2021**

**AIM: Find the gcd of 2 numbers**

```
x= int(input("Enter 1st number: "))
y= int(input("Enter 2nd number: "))
i = 1
while(i <= x and i <= y):
   if(x % i == 0 and y% i == 0):
      gcd = i
   i = i + 1
print("GCD :", gcd)
```

**OUTPUT**

```
Enter 1st number: 120
Enter 2nd number: 5
GCD : 5
```

**PROGRAM NO: 20**

**DATE:29/11/2021**

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

```
num = [1,2,3,4,5,6,7,8,9,10]
print( "Original list:",num)
num = [x for x in num if x%2!=0]
print("list after removing Even numbers:",num)
```

**OUTPUT**

```
Original list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
list after removing Even numbers: [1, 3, 5, 7, 9]
```

## II. COURSE OUTCOME 2(CO2)

**PROGRAM NO: 1**

**DATE:01/12/2021**

**AIM: Program to find the factorial of a number**

```
n=int(input('Enter a number : '))
f=1
for i in range(1,n+1):
    f=f*i
print ('Factorial of',n, '=',f)
```

**OUTPUT**

```
Enter a number : 5
Factorial of 5 = 120
```

**PROGRAM NO: 2**

**DATE:01/12/2021**

**AIM: Generate Fibonacci series of N terms**

```
n = int(input("Enter the limit : "))
a = 0
b = 1
sum = 0
count = 1
print("Fibonacci Series :",end= " ")
while(count <= n):
    print(sum, end = " ")
    count += 1
    a = b
    b = sum
    sum = a + b
```

**OUTPUT**

Enter the limit : 5
Fibonacci Series : 0 1 1 2 3

**PROGRAM NO: 3**

**DATE:01/12/2021**

**AIM: Find the sum of all items in a list**

```
list = [10, 15, 20, 25, 30]
total = sum(list)
print("Sum of list : ",total)
```

**OUTPUT**

```
Sum of list :  100
```

**PROGRAM NO: 4**

**DATE:01/12/2021**

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

```
from math import sqrt as s
for i in range(1000,10000):
   if s(i)==int(s(i)) and i%2==0:
      print(i,end=" ")
```

**OUTPUT**

1024 1156 1296 1444 1600 1764 1936 2116 2304 2500 2704 2916 3136 3364 3600 3844 4096 4356 4624 4900 5184 5476 5776 6084 6400 6724 7056 7396 7744 8100 8464 8836 9216 9604

**PROGRAM NO: 5**

**DATE:01/12/2021**

**AIM: Display the given pyramid with step number accepted from user.**

```
rows = int(input("Enter the number of rows: "))
for i in range(1, rows+1):
    for j in range(1,i+1):
        print(i * j, end=' ')
    print()
```

**OUTPUT**

```
Enter the number of rows: 3
1
2 4
3 6 9
```

**PROGRAM NO: 6**

**DATE:01/12/2021**

**AIM: Count the number of characters (character frequency) in a string**

```
test_str=str(input("Enter the string : "))
freq = {}
for i in test_str:
   if i in freq:
      freq[i] += 1
   else:
      freq[i] = 1
print ("Count of all characters : "+ str(freq))
```

**OUTPUT**

```
Enter the string : ENGLISH
Count of all characters : {'E': 1, 'N': 1, 'G': 1, 'L': 1, 'I': 1, 'S': 1, 'H': 1}
```

**PROGRAM NO: 7**

**DATE:08/12/2021**

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

```
str=input("enter a string:")
print("Input string is:",str)
if(str.endswith("ing")):
   str=str+'ly'
else:
   str=str+'ing'
print("the formated string is:",str)
```

**OUTPUT**

```
enter a string:play
Input string is: play
the formated string is: playing

enter a string:playing
Input string is: playing
the formated string is: playingly
```

**PROGRAM NO: 8**

**DATE:08/12/2021**

**AIM: Accept a list of words and return length of longest word.**

```
a=[]
n= int(input("Enter the number of elements in list:"))
for x in range(0,n):
    element=input("Enter element "+str(x+1)+":")
    a.append(element)
    max1=len(a[0])
    temp=a[0]
for i in a:
    if(len(i)>max1):
        max1=len(i)
        temp=i
        print("Longest Word : ",temp)
        print("Length of longest word : ",max1)
```

**OUTPUT**

```
Enter the number of elements in list:4
Enter element 1:python
Enter element 2:programming
Enter element 3:is
Enter element 4:simple
Longest Word :  programming
Length of longest word :  11
```

**PROGRAM NO: 9**

**DATE:08/12/2021**

**AIM: Construct the following pattern using nested loop**

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

```
n= int(input("Enter the limit:"))
for i in range(n):
    for j in range(i):
        print('* ', end="")
    print('')
for i in range(n,0,-1):
    for j in range(i):
        print('* ', end="")
    print('')
```

**OUTPUT**

```
Enter the limit:3


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

**PROGRAM NO: 10**

**DATE:08/12/2021**

**AIM: Generate all factors of a number. def print_factors(x):**

```
def factors(x):
    print("The factors of",x,"are:")
    for i in range(1, x + 1):
        if x % i == 0:
            print(i)
n=int(input("Enter a number:"))
factors(n)
```

**OUTPUT**

```
Enter a number:5
The factors of 5 are:
1
5
```

**PROGRAM NO: 11**

**DATE:08/12/2021**

**AIM: Write lambda functions to find area of square, rectangle and triangle.**

```
import math
t_area = lambda b,h : 1/2*b*h
r_area = lambda l,b : l*b
s_area = lambda a : a*a

print("Area of Triangle :", t_area(10,20))
print("Area of Rectangle:", r_area(30,20))
print("Area of Square :", s_area(15))
```

**OUTPUT**

```
Area of Triangle : 100.0
Area of Rectangle: 600
Area of Square : 225
```

# III. COURSE OUTCOME 3(CO3)

**PROGRAM NO: 1**

**DATE:15/12/2021**

**AIM: Design modules and packages – builtin and user defined packages.**

**MATH PROGRAM**

```
import math
print("The value of pi:",math.pi)
import math as m
print("The value of pi is :", m.pi)
from math import pi,sqrt
print("The value of pi is : ", pi)
print("The square root of 4 is : ", sqrt(4))
print("Value of sin(90): ",math.cos(90))
print("Value of cos(90): ",math.sin(90))
print("Value of tan(90): ",math.tan(90))
```

**OUTPUT**

```
The value of pi: 3.141592653589793
The value of pi is : 3.141592653589793
The value of pi is :  3.141592653589793
The square root of 4 is :  2.0
Value of cos(90):  -0.4480736161291701
Value of sin(90):  0.8939966636005579
Value of tan(90):  -1.995200412208242
```

## CALENDAR PROGRAM

```
import calendar
mm = int(input("Enter Month : "))
yy = int(input("Enter Year : "))
print(calendar.month(yy,mm))
print(calendar.calendar(1999))
```

## OUTPUT

```
Enter Month : 12
Enter Year : 1999
   December 1999
Mo Tu We Th Fr Sa Su
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

                                  1999

      January               February                March
Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
             1  2  3    1  2  3  4  5  6  7    1  2  3  4  5  6  7
 4  5  6  7  8  9 10    8  9 10 11 12 13 14    8  9 10 11 12 13 14
11 12 13 14 15 16 17   15 16 17 18 19 20 21   15 16 17 18 19 20 21
18 19 20 21 22 23 24   22 23 24 25 26 27 28   22 23 24 25 26 27 28
25 26 27 28 29 30 31                          29 30 31

       April                  May                   June
Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
          1  2  3  4                1  2           1  2  3  4  5  6
 5  6  7  8  9 10 11    3  4  5  6  7  8  9    7  8  9 10 11 12 13
12 13 14 15 16 17 18   10 11 12 13 14 15 16   14 15 16 17 18 19 20
19 20 21 22 23 24 25   17 18 19 20 21 22 23   21 22 23 24 25 26 27
26 27 28 29 30         24 25 26 27 28 29 30   28 29 30
                       31

        July                 August                September
Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
          1  2  3  4                      1              1  2  3  4  5
 5  6  7  8  9 10 11    2  3  4  5  6  7  8    6  7  8  9 10 11 12
12 13 14 15 16 17 18    9 10 11 12 13 14 15   13 14 15 16 17 18 19
19 20 21 22 23 24 25   16 17 18 19 20 21 22   20 21 22 23 24 25 26
26 27 28 29 30 31      23 24 25 26 27 28 29   27 28 29 30
                       30 31

      October              November              December
Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
             1  2  3    1  2  3  4  5  6  7          1  2  3  4  5
 4  5  6  7  8  9 10    8  9 10 11 12 13 14    6  7  8  9 10 11 12
11 12 13 14 15 16 17   15 16 17 18 19 20 21   13 14 15 16 17 18 19
18 19 20 21 22 23 24   22 23 24 25 26 27 28   20 21 22 23 24 25 26
25 26 27 28 29 30 31   29 30                 27 28 29 30 31
```

**TIME PROGRAM**

```
import time
print("Current time in sec : ",time.time())
print("Current time : ",time.ctime())
print("Current time after 30 sec : ",time.time()+30)
t = time.localtime()
print("Time : ", t)
print("Current Year :", t.tm_year)
print("Current Month :", t.tm_mon)
print("Current Day :", t.tm_mday)
print("Current Hour :", t.tm_hour)
print("Current Weekday :", t.tm_wday)
print("Day of year :", t.tm_yday)
```

**OUTPUT**

```
Current time in sec :  1644039881.7853935
Current time :  Sat Feb  5 11:14:41 2022
Current time after 30 sec :  1644039911.7853935
Time :  time.struct_time(tm_year=2022, tm_mon=2, tm_mday=5, tm_hour=11, tm_min=14, tm_sec=41,
tm_wday=5, tm_yday=36, tm_isdst=0)
Current Year : 2022
Current Month : 2
Current Day : 5
Current Hour : 11
Current Weekday : 5
Day of year : 36
```

**DATETIME PROGRAM**

```
import datetime
t=datetime.time(22,56,44)
print(t)
print("Hour : ", t.hour)
print("Minute : ", t.minute)
print("Second : ", t.second
print("============================")
d = datetime.date.today()
print(d)
td = datetime.timedelta(days=2)
print(td)

d2 = d+td
print("After adding two days :",d2)
print("d2-d",d2-d)
print("d2>d",d2>d)

d1 = datetime.date.today()
t1 = datetime.time(12,44,56)
print("Date and Time : ",d1, t1)
```

**OUTPUT**

```
22:56:44
Hour :  22
Minute :  56
Second :  44
============================
2022-02-05
2 days, 0:00:00
After adding two days : 2022-02-07
d2-d 2 days, 0:00:00
d2>d True
Date and Time :  2022-02-05 12:44:56
```

**STATISTICS PROGRAM**

```
import statistics
# Calculate average values
print("Mean : ",statistics.mean([1, 3, 5, 7, 9, 11, 13]))
print("Mean : ",statistics.mean([1, 3, 5, 7, 9, 11]))
print("Mean : ",statistics.mean([-11, 5.5, -3.4, 7.1, -9, 22]))
print("==============================")
# Calculate middle values
print("Median : ",statistics.median([1, 3, 5, 7, 9, 11, 13]))
print("Median : ",statistics.median([1, 3, 5, 7, 9, 11]))
print("Median : ",statistics.median([-11, 5.5, -3.4, 7.1, -9, 22]))
print("==============================")
# Calculate the mode
print("Mode :",statistics.mode([1, 3, 3, 3, 5, 7, 9, 11]))
print("Mode :",statistics.mode([1, 1, 3, -5, 7, -9, 11]))
print("Mode :",statistics.mode(['red', 'green', 'blue', 'red']))
print("==============================")
# Calculate the variance from a sample of data
print("Varience :",([1, 3, 5, 7, 9, 11]))
print("Varience :",statistics.variance([2, 2.5, 1.25, 3.1, 1.75, 2.8]))
print("Varience :",statistics.variance([-11, 5.5, -3.4, 7.1]))
print("Varience :",statistics.variance([1, 30, 50, 100]))
print("==============================")
# Calculate harmonic mean
print("Hermonic mean",statistics.harmonic_mean([40, 60, 80]))
print("Hermonic mean",statistics.harmonic_mean([10, 30, 50, 70, 90]))
print("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-")
```

**OUTPUT**

```
Mean :  7
Mean :  6
Mean :  1.8666666666666667
==============================
Median :  7
Median :  6.0
Median :  1.05
==============================
Mode : 3
Mode : 1
Mode : red
==============================
Varience : [1, 3, 5, 7, 9, 11]
Varience : 0.4796666666666667
Varience : 70.80333333333334
Varience : 1736.9166666666667
==============================
```

Hermonic mean 55.38461538461538
Hermonic mean 27.97513321492007
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=- -=--=--=-

**RANDOM PROGRAM**

```
import random
print(random.random())
print("==============================")
mylist = ["apple", "banana", "cherry"]
random.shuffle(mylist)
print(mylist)
print("==============================")
random.seed(10)
print(random.random())
print("==============================")
mylist = ["apple", "banana", "cherry"]
print(random.choice(mylist))
print("==============================")
print(random.randrange(3, 9))
```

**OUTPUT**

```
0.42752636832484947
==============================
['apple', 'banana', 'cherry']
==============================
0.5714025946899135
==============================
banana
==============================
6
```

**PROGRAM NO: 2**

**DATE:15/12/2021**

**AIM: 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)**

**Graphics package:**

**rectangle.py**

```
def rectangle(l,b):
    print("Area of rectangle :",l*b)
    print("Perimeter of rectangle :",2*(l+b))
    print("----------------------------------")
```

**circle.py**

```
def circle(r):
    print("Area of Circle :",3.14*r*r)
    print("Perimeter of rectangle :",2*3.14*r)
    print("----------------------------------")
```

**cuboid.py**

```
def cuboid(l,b,h):
    print("Area of cuboid :",(2*l*b)+(2*l*h)+(2*l*b))
    print("Perimeter of cuboid :",4*(l+b+h))
    print("----------------------------------")
```

**sphere.py**

```
def sphere(r):
    print("Area of sphere :",4*3.14*r*r)
    print("Volume of sphere :",(4/3)*3.14*r*r)
    print("----------------------------------")
```

**appackage_views.py**

```
from Graphics import circle
from Graphics import rectangle
from Graphics import cuboid
from Graphics import sphere

l = int(input("Enter length of rectangle :"))
b = int(input("Enter breadth of rectangle :"))
```

```
rectangle.rectangle(l,b)

r = int(input("Enter Radius of circle :"))
circle.circle(r)

l = int(input("Enter length of cuboid :"))
b = int(input("Enter breadth of cuboid :"))
h = int(input("Enter height of cuboid :"))
cuboid.cuboid(l,h,b)

r = int(input("Enter Radius of sphere :"))
sphere.sphere(r)
```

**OUTPUT**

```
Enter length of rectangle :8
Enter breadth of rectangle :4
Area of rectangle : 32
Perimeter of rectangle : 24
---------------------------------
Enter Radius of circle :5
Area of Circle : 78.5
Perimeter of rectangle : 31.400000000000002
---------------------------------
Enter length of cuboid :6
Enter breadth of cuboid :3
Enter height of cuboid :4
Area of cuboid : 132
Perimeter of cuboid : 52
---------------------------------
Enter Radius of sphere :5
Area of sphere : 314.0
Volume of sphere : 104.66666666666667
---------------------------------
```

## IV. COURSE OUTCOME 4(CO4)

**PROGRAM NO: 1**

**DATE:09/01/2022**

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

```
class Rectangle:
    def __init__(self):
        self.l=int(input("Enter the length : "))
        self.b=int(input("Enter the breadth : "))
        self.area=self.l*self.b
        self.perimeter=2*(self.l+self.b)
    def display(self):
        print("Area of Rectangle : ",self.area)
        print("Perimeter of Rectangle : ",self.perimeter)
print("\nFirst Rectangle")
print("---------------")
p1=Rectangle()
p1.display()
print("\nSecond Rectangle")
print("---------------")
p2=Rectangle()
p2.display()
if p1.area>p2.area:
    print("---------------")
    print("First Rectangle with Area", p1.area, "has larger area.")
else:
    print("---------------")
    print("Second Rectangle with Area",p2.area,"has larger area.")
```

**OUTPUT**

```
First Rectangle
---------------
Enter the length : 8
Enter the breadth : 6
Area of Rectangle :  48
Perimeter of Rectangle :  28

Second Rectangle
---------------
Enter the length : 10
Enter the breadth : 5
Area of Rectangle :  50
Perimeter of Rectangle :  30
---------------
Second Rectangle with Area 50 has larger area.
```

**PROGRAM NO: 2**

**DATE:09/01/2022**

**AIM: 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.**

```
class bank:
    def __init__(self):
        self.balance=0
        name=input("Enter the name of account holder : ")
        acno=int(input("Enter the account no : "))
        print ("\n---The account is created---")
        print("\nName of Account Holder : ",name)
        print("\nAccount no : ",acno)
    def deposit(self):
        amount=int(input("\nEnter the amount to deposit : "))
        self.balance+=amount
    def withdraw(self):
        amount = float(input("Enter amount to be Withdrawn : "))
        if (self.balance>=amount):
            self.balance-=amount
            print("\nYou Withdraw:", amount)
        else:
            print("\nInsufficient balance!!!")
    def display(self):
        print("\nAvailable Balance : ",self.balance)

b=bank()
b.deposit()
b.withdraw()
b.display()
```

**OUTPUT**

```
Enter the name of account holder : SREERAG T V
Enter the account no : 030303

---The account is created---

Name of Account Holder :  SREERAG T V

Account no :  30303
Enter the amount to deposit : 10000000
Enter amount to be Withdrawn : 5000000

You Withdraw: 5000000.0

Available Balance :  5000000.0
```

**PROGRAM NO: 3**

**DATE:09/01/2022**

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

```
class rectangle:
   def __init__(self,length,width):
      self.__length=length
      self.__width=width
   def __lt__(self,a1):
      area1=self.__length*self.__width
      area2=a1.__length*a1.__width
      if(area1<area2):
         return(True)
      else:
         return(False)

a1=int(input("Length of  First Rectangle:"))
b1=int(input("Width of First Rectangle:"))
r1=rectangle(a1,b1)

a2=int(input("Length of Second Rectangle:"))
b2=int(input("Width of Second Rectangle:"))
r2=rectangle(a2,b2)
if(r1<r2):
   print("Second Rectangle is larger.")
else:
   print("First Rectangle is larger.")
```

**OUTPUT**

```
Length of  First Rectangle:80
Width of First Rectangle:40

Length of Second Rectangle:50
Width of Second Rectangle:30

--------------------------------

First Rectangle is larger.
```

**PROGRAM NO: 4**

**DATE:09/01/2022**

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

```
class Time:
    def __init__(self,hour,minute,second):
        self.__hour=hour
        self.__minute=minute
        self.__second=second
    def __add__(self,h):
        second=self.__second+h.__second
        minute=self.__minute+h.__minute
        hour=self.__hour+h.__hour
        if(second>60):
            second=second-60
            minute=minute+1
        if(minute>60):
            minute=minute-60
            hour=hour+1
        if(hour>24):
            hour=hour-24
        return hour,minute,second
print("---Enter First Time---\n")
h1=int(input("Enter The Hour : "))
m1=int(input("Enter The Minute : "))
s1=int(input("Enter The Second : "))

t1=Time(h1,m1,s1)

print("\n---Enter Second Time---\n")
h2=int(input("Enter The Hour : "))
m2=int(input("Enter The Minute : "))
s2=int(input("Enter The Second : "))

t2=Time(h2,m2,s2)

hr,min,sec=t1+t2
print("--------------------")
print(hr,end=":")
print(min,end=":")
print(sec,end=" ")
```

**OUTPUT**

```
---Enter First Time---

Enter The Hour : 24
Enter The Minute : 5
Enter The Second : 5

---Enter Second Time---

Enter The Hour : 3
Enter The Minute : 5
Enter The Second : 5
--------------------
3:10:10
```

**PROGRAM NO: 5**

**DATE:09/01/2022**

**AIM: 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.**

```
class publisher:

  def __init__(self,pname):
    self.pname=pname

  def display(self):
   print("Publisher Name:",self.pname)

class book(publisher):

  def get(self,title,author):
    self.title=title
    self.author=author

  def display(self):
   print("Title Name:",self.title)
   print("Author Name:",self.author)

class python(book):

 def __init__(self,price,nop,pname):
  super().__init__(pname)
  self.price=price
  self.nop=nop
 def details(self):
  print("Price:",self.price)
  print("No of pages:",self.nop)

s1=python(200,180,"A P J Abdul kalam")
s1.get("Wings Of Fire","A P J Abdul kalam")
s1.display()
s1.details()
```

**OUTPUT**

```
Title Name: Wings Of Fire
Author Name: A P J Abdul kalam
Price: 200
No of pages: 180
```

## V. COURSE OUTCOME 5(CO5)

**PROGRAM NO: 1**

**DATE:30/01/2022**

**AIM: Write a Python program to read a file line by line and store it into a list.**

```
f1=open("firstfile.txt","w")
f1.write("This is my first file in python.\nWant to work with files.\nThis is my third line.")
f1.close()
f1=open("firstfile.txt","r")
f1.seek(0,0)
ff=f1.readlines()
for x in range(0,len(ff)):
print(ff[x])
print()
print(ff)
f1.close()
```

**OUTPUT**

```
This is my first file in python.

Want to work with files.

This is my third line.

['This is my first file in python.\n', 'Want to work with files.\n', 'This is my third line.']
```

**PROGRAM NO: 2**

**DATE:30/01/2022**

**AIM: Python program to copy odd lines of one file to other**

```
f1 = open("firstfile.txt","r")
f2=open("odd.txt","w")
for x in f1:
   print(x)
print("---------------")
f1.seek(0,0)
ff=f1.readlines()
for x in range(0,len(ff)):
   if(x%2==0):
      f2.write(ff[x])
      print(ff[x])
print("---------------")
```

**OUTPUT**

```
This is my first file in python.

Want to work with files.

This is my third line.
---------------
This is my first file in python.

This is my third line.
---------------
```

**firstfile.txt**

```
This is my first file in python.
Want to work with files.
This is my third line.
```

**odd.txt**

```
This is my first file in python.
This is my third line.
```

**PROGRAM NO: 3**

**DATE:30/01/2022**

**AIM: Write a Python program to read each row from a given csv file and print a list of strings**

```
import csv
filename = "username.csv"
rows = []
cf=open(filename, 'r')
csvreader = csv.reader(cf)
for r in csvreader:
    rows.append(r)
print(rows)
cf.close()
```

**OUTPUT**

```
[['Username; Identifier;Firstname;Lastname'], ['booker12;9012;Rachel;Booker'],
['grey07;2070;Laura;Grey'], ['johnson81;4081;Craig;Johnson'], ['jenkins46;9346;Mary;Jenkins'],
['smith79;5079;Jamie;Smith']]
```

**PROGRAM NO: 4**

**DATE:30/01/2022**

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

```
import csv
filename = "Names1.csv"
cf=open(filename, 'r')
 #csvreader = csv.reader(cf)
data = csv.DictReader(cf)
print("No Company")
for r in data:
    print(r['No'], r['Company'])
```

**OUTPUT**

```
No Company
1 Ferrari
2 Porsche
3 Bugatti
4 Rolls Royce
5 BMW
```

**PROGRAM NO: 5**

**DATE:30/01/2022**

**AIM: 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**

```
import csv

field_names = ['No', 'Company', 'Car Model']

cars = [
{'No': 1, 'Company': 'Ferrari', 'Car Model': '488 GTB'},
{'No': 2, 'Company': 'Porsche', 'Car Model': '918 Spyder'},
{'No': 3, 'Company': 'Bugatti', 'Car Model': 'La Voiture Noire'},
{'No': 4, 'Company': 'Rolls Royce', 'Car Model': 'Phantom'},
{'No': 5, 'Company': 'BMW', 'Car Model': 'BMW X7'},
]

with open('Names1.csv', 'w') as csvfile:
   writer = csv.DictWriter(csvfile, fieldnames = field_names)
   writer.writeheader()
   writer.writerows(cars)

filename = "Names1.csv"

cf=open("Names1.csv", 'r')
rows=[]
csvreader = csv.reader(cf)
for r in csvreader:
  rows.append(r)
for r in rows:
     print(*r)
```

**OUTPUT**

```
No Company Car Model

1 Ferrari 488 GTB

2 Porsche 918 Spyder

3 Bugatti La Voiture Noire

4 Rolls Royce Phantom

5 BMW BMW X7
```