Python Programming Lab Record (Course Code: 20CS3352)

by

**G. TARAKA ROHITH (21501A0563)**

**II B.Tech I Semester**

**in**

**Computer Science and Engineering**

****

**Prasad V Potluri Siddhartha Institute of Technology**

Accredited with A+ grade by NAAC, NBA Accredited, and Autonomous ISO 9001:2015 Certified Institute

Permanently Affiliated to JNTUK-Kakinada and approved by AICTE

**Kanuru, Vijayawada-520 007**

**Prasad V. Potluri Siddhartha Institute of Technology**

Accredited with A+ grade by NAAC, NBA Accredited, and Autonomous ISO 9001:2015 Certified Institute

Permanently Affiliated to JNTUK-Kakinada and approved by AICTE

#### Kanuru, Vijayawada-520 007

****

**CERTIFICATE**

This is to certify that Mr. / Ms. G. TARAKA ROHITH bearing Regd. no. 21501A0563 of II B.Tech I semester **Computer Science and Engineering** branch has satisfactorily completed **Python Programming Laboratory (Course Code: 20CS3352)** during the academic year 2023-24.

Signature of Faculty Signature of HOD

Signature of Internal Examiner Signature of External Examiner

**DECLARATION**

I declare that the **Python Programming Laboratory** (**Course Code: 20CS3352)** report is my own original work and does not involve plagiarism or unauthorized collusion, except where due credit is given to the work of others. The report is based on results and spectra obtained by me during my laboratory session. This laboratory report has not previously been submitted for assessment in this or any other course.

Student Name: G. TARAKA ROHITH Regd.No : 21501A0563

II B.Tech I Sem CSE Section-1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PVPSIT – CSE Python Programming Lab Record (Course Code: 20CS3352)  **Index** | | | | | | | |
|  | **S.No** | **Date** | **Title of the Experiment** | **Page No.** | **Day to**  **Day Marks** | **Record Marks** | **Date**  **with Sign** |
|  |  | Introduction to python programming |  |  |  |  |
| 1. | 19. 08. 2023 | and demonstrate the difference |
|  |  | between interactive mode and IDEs. |
| 2. | 19. 08. 2023 | Demonstrate programs using basic constructs. |  |  |  |  |
| 3. | 19. 08. 2023 | Programs to demonstrate Programming constructs. |  |  |  |  |
| 4. | 26. 08. 2023 | Programs to demonstrate decision making and branching. (Selection) |  |  |  |  |
| 5. | 02. 09. 2023 | Programs to demonstrate iterative statements. |
| 6. | 02. 09. 2023 | Build modular programs using functions. |  |  |  |  |
|  |  | Programs to perform operations on |  |  |  |  |
| 7. | 16. 09. 2023 | strings, regular expressions with built |
|  |  | – in functions. |
| 8. | 16. 09. 2023 | Implement programs using various |
| data structures. |
| 9. | 23. 09. 2023 | Programs to demonstrate access |  |  |  |  |
| specifiers. |
| 10. | 23. 09. 2023 | Programs to demonstrate types of Inheritance, polymorphism. |  |  |  |  |
| 11. | 07. 10. 2023 | Python programming to demonstrate Exception handling. |  |  |  |  |
| 12. | 07. 10. 2023 | Programs to demonstrate Files. |  |  |  |  |
|  |  | Installing, importing accessing and |  |  |  |  |
| 13. | 28. 10. 2023 | computations on a dataset using |
|  |  | Pandas library. |
|  |  | Installing, importing accessing and |  |  |  |  |
| 14. | 04. 11. 2023 | computations on a dataset using |
|  |  | Numpy library. |
|  |  | Installing, importing accessing and |  |  |  |  |
| 15. | 18. 11. 2023 | computations on a dataset using |
|  |  | MatplotLib library. |
| IV | | | | | | | |

# EXP.NO-1: Introduction to Python Programming and demonstrate the difference between interactive mode and IDEs.

### Introduction to Python Programming

* + 1. What is Python

Python is a widely-used, interpreted, object-oriented, and high- level programming language with dynamic semantics, used for general- purpose programming. It’s everywhere, and people use numerous Python-powered devices on a daily basis, whether they realize it or not.

* + 1. Python History

Python was created by **Guido van Rossum** who is a Dutch programmer. He was also known as the "Benevolent dictator for life" (BDFL) for python, until he stepped down from the position in July 2018. He Worked at the Centrum Wiskunde & Informatica (CWI) and contributed a glob() routine to BSD Unix during 1986 and helped develop the ABC programming language. But later during December 1989, he was looking for a hobby project around the time in Christmas and started working on a new interpreter for a new language. He later named this language as Python after choosing the word form a TV serial named Monty Python's Flying Circus. He admits that ABC language is the predecessor of Python language. The first release of the language was done in 1991.

* + 1. Python Version List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Latest micro version** | **Release date** | **End of full support** | **End of security fixes** |
| 0.9 | 0.9.9 | 1991-02-20 | 1993-07-29 | |
| 1.0 | 1.0.4 | 1994-01-26 | 1994-02-15 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Latest micro version** | **Release date** | **End of full support** | **End of security fixes** |
| 1.1 | 1.1.1 | 1994-10-11 | 1994-11-10 | |
| 1.2 |  | 1995-04-13 | Unsupported | |
| 1.3 |  | 1995-10-13 | Unsupported | |
| 1.4 |  | 1996-10-25 | Unsupported | |
| 1.5 | 1.5.2 | 1998-01-03 | 1999-04-13 | |
| 1.6 | 1.6.1 | 2000-09-05 | 2000-09 | |
| 2.0 | 2.0.1 | 2000-10-16 | 2001-06-22 | |
| 2.1 | 2.1.3 | 2001-04-15 | 2002-04-09 | |
| 2.2 | 2.2.3 | 2001-12-21 | 2003-05-30 | |
| 2.3 | 2.3.7 | 2003-06-29 | 2008-03-11 | |
| 2.4 | 2.4.6 | 2004-11-30 | 2008-12-19 | |
| 2.5 | 2.5.6 | 2006-09-19 | 2011-05-26 | |
| 2.6 | 2.6.9 | 2008-10-01 | 2010-08-24 | 2013-10-29 |
| 2.7 | 2.7.18 | 2010-07-03 | 2020-01-01 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Latest micro version** | **Release date** | **End of full support** | **End of security fixes** |
| 3.0 | 3.0.1 | 2008-12-03 | 2009-06-27 | |
| 3.1 | 3.1.5 | 2009-06-27 | 2011-06-12 | 2012-04-06 |
| 3.2 | 3.2.6 | 2011-02-20 | 2013-05-13 | 2016-02-20 |
| 3.3 | 3.3.7 | 2012-09-29 | 2014-03-08 | 2017-09-29 |
| 3.4 | 3.4.10 | 2014-03-16 | 2017-08-09 | 2019-03-18 |
| 3.5 | 3.5.10 | 2015-09-13 | 2017-08-08 | 2020-09-30 |
| 3.6 | 3.6.15 | 2016-12-23 | 2018-12-24 | 2021-12-23 |
| 3.7 | 3.7.15 | 2018-06-27 | 2020-06-27 | 2023-06-27 |
| 3.8 | 3.8.15 | 2019-10-14 | 2021-05-03 | 2024-10 |
| 3.9 | 3.9.15 | 2020-10-05 | 2022-05-17 | 2025-10 |
| 3.10 | 3.10.8 | 2021-10-04 | 2023-05 | 2026-10 |
| 3.11 | 3.11.0 | 2022-10-24 | 2024-05 | 2027-10 |
| 3.12 |  | 2023-10 | 2025-05 | 2028-10 |

* + 1. Key features of Python
       1. Easy to learn and Readable language
       2. Interpreted Language
       3. Dynamically Typed Language
       4. Open Source and Free
       5. Large Standard Library
       6. High-Level Language
       7. Object Oriented Programming Language
       8. Large Community Support
       9. Platform Independent
       10. Extensible and Embeddable
       11. Graphical User Interface (GUI) Support
    2. Application Areas

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non- programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

### Python Installation

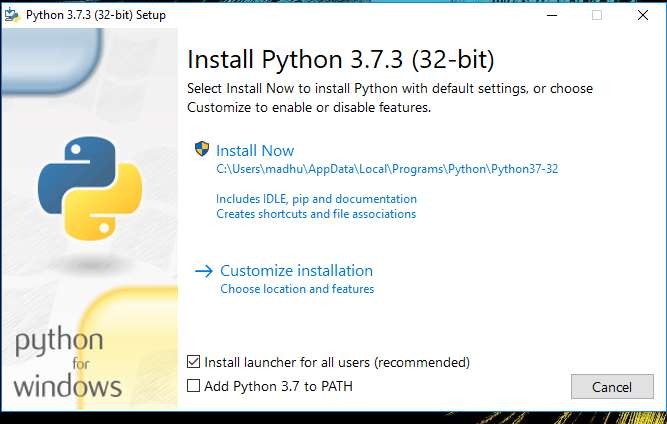
#### Step 1: Download the Python Installer binaries

1. Open the official Python website in your web browser. Navigate to the Downloads tab for Windows.
2. Choose the latest Python 3 release. In our example, we choose the latest Python 3.7.3 version.
3. Click on the link to download Windows x86 executable installer if you are using a 32-bit installer. In case your Windows installation is a 64-bit system, then download Windows x86-64 executable installer.



#### Step 2: Run the Executable Installer

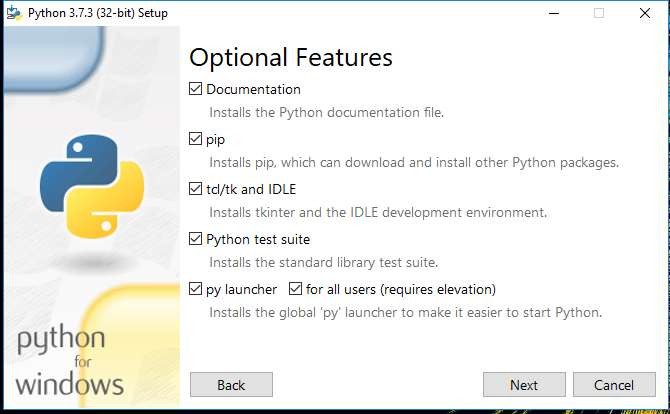
1. Once the installer is downloaded, run the Python installer.
2. Check the Install launcher for all users check box. Further, you may check the Add Python 3.7 to path check box to include the interpreter in the execution path.



1. Select Customize installation. Choose the optional features by checking the

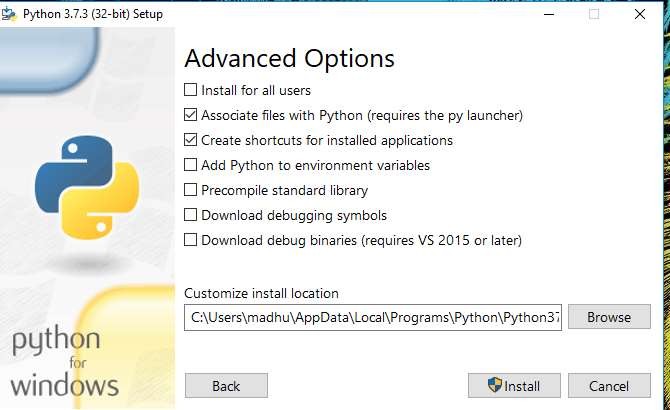
following check boxes:

1. Documentation
2. pip
3. tcl/tk and IDLE (to install tkinter and IDLE)
4. Python test suite (to install the standard library test suite of Python)
5. Install the global launcher for `.py` files. This makes it easier to start Python
6. Install for all users

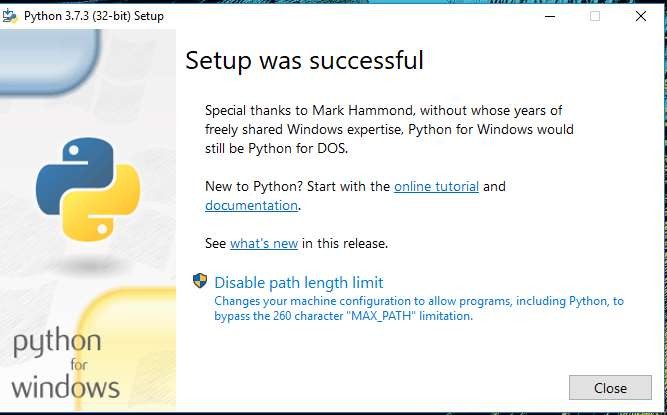


.

Click Next.This takes you to Advanced Options available while installing Python. Here, select the Install for all users and Add Python to environment variables check boxes. Optionally, you can select the Associate files with Python, Create shortcuts for installed applications and other advanced options. Make note of the python installation directory displayed in this step. You would need it for the next step. After selecting the Advanced options, click Install to start installation.



1. Once the installation is over, you will see a Python Setup Successful window.

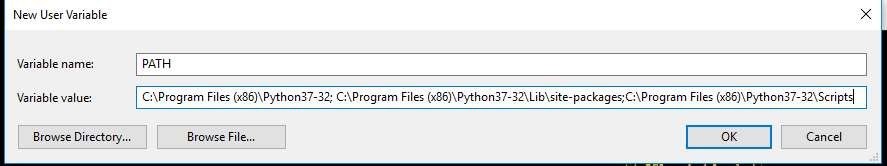
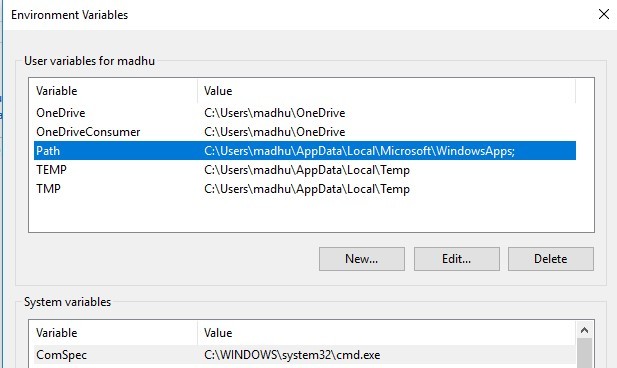


#### Step 3: Add Python to environmental variables

The last (optional) step in the installation process is to add Python Path to the System Environment variables. This step is done to access Python through the command line. In case you have added Python to environment variables while setting the Advanced options during the installation procedure, you can avoid this step. Else, this step is done manually as follows. In the Start menu, search for “advanced system settings”. Select “View advanced system settings”. In the “System Properties” window, click on the “Advanced” tab and then click on the “Environment Variables” button. Locate the Python installation directory on your system. If you followed the steps exactly as above, python will be installed in below locations:

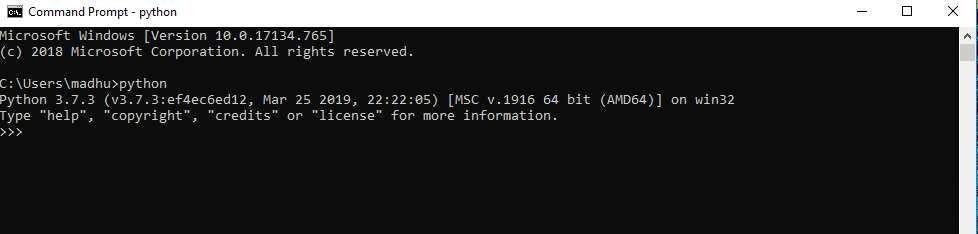
* + C:\Program Files (x86)\Python37-32: for 32-bit installation
  + C:\Program Files\Python37-32: for 64-bit installation

The folder name may be different from “Python37-32” if you installed a different version. Look for a folder whose name starts with Python. Append the following entries to PATH variable as shown below:

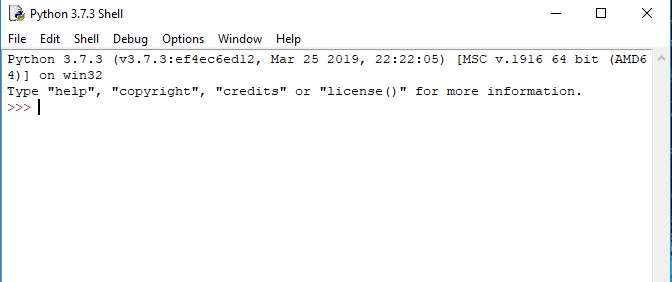


#### Step 4: Verify the Python Installation

You have now successfully installed Python 3.7.3 on Windows 10. You can verify if the Python installation is successful either through the command line or through the IDLE app that gets installed along with the installation. Search for the command prompt and type “python”. You can see that Python 3.7.3 is successfully installed.



An alternate way to reach python is to search for “Python” in the start menu and clicking on IDLE (Python 3.7 64-bit). You can start coding in Python using the Integrated Development Environment(IDLE).



### Difference between interactive mode and IDEs

|  |  |
| --- | --- |
| **Interactive Mode** | **Script Mode** |
| It is a way of executing a Python program in which statements are written in command prompt and result is obtained on the same. | In the script mode, the Python program is written in a file. Python interpreter reads the file and then executes it and provides the desired result. The program is compiled  in the command prompt, |
| The interactive mode is more suitable for  writing very short programs. | Script mode is more suitable for writing  long programs. |
| Editing of code can be done but it is a  tedious task. | Editing of code can be easily done in script  mode. |
| We get output for every single line of code in interactive mode i.e. result is obtained after execution of each line of  code. | In script mode entire program is first compiled and then executed. |
| Code cannot be saved and used in the  future. | Code can be saved and can be used in the  future. |
| It is more preferred by beginners. | It is more preferred by experts. |

1.4 Viva Questions:

* What kind of programming language is Python?

Ans: Python is a high-level object- oriented programming language.

* What is syntax for converting data type? float (input (“enter the data))
* Why python is preferred to all other programming languages?

Because it is easy to write the code in python. Here we need not to follow complex syntax

* What are the Optional features in Python?

Documentation, pip, tcl/tk and IDLE, Python test suite (to install the standard library test suite of Python), Install the global launcher for `.py` files.

# EXP.NO-2: Demonstrate Programs using Basic Constructs

## Task – 2.1:

**Problem Statement:**

The perimeter of your class room is P meters. If the length of the class is l meters, then write a python program to find its width (w) and the area of your same class room

A in square feet.

(Hint: P=125 meters, so P = 2(length + width))

## Program:

length = float (input ()) P = float (input ()) width = (P/2) – length

area = length \* breadth \* 10.7639 print (width)

print (area)

## Output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | 1.6 | 0.4 | 0.4 | Pass |
|  | 4.0 | 6.888896 | 6.888896 |  |
| 2 | 10 | 10 | 10 | Pass |
|  | 40 | 1076.39 | 1076.39 |  |
| 3 | 4.2 | 5.8 | 5.8 | Pass |
|  | 20 | 262.208604 | 262.208604 |  |
| 4 | 6.2 | 4.6 | 4.6 | Pass |
|  | 21.6 | 306.986428 | 306.986428 |  |

## Task – 2.2:

**Problem Statement:**

When the diameter of the Cricket ground is D cm. Write a python program to find the radius (r), area (A), and perimeter (P) of the cricket ground.

(Hint: d=15000, so, perimeter P=2\*pi\*r)

## Program:

D = float (input ()) pi = 3.14

r = D / 2

A = pi \* r \*r P = 2 \* pi \* r print (r) print (A) print (P)

## Output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | 42 | 21 | 21 | Pass |
|  |  | 1384.74 | 1384.74 |  |
|  |  | 18.212 | 18.212 |  |
| 2 | 5.8 | 21 | 21 | Pass |
|  |  | 1384.74 | 1384.74 |  |
|  |  | 18.212 | 18.212 |  |
| 3 | 49.62 | 24.81 | 24.81 | Pass |
|  |  | 1932.783354 | 1932.783354 |  |
|  |  | 155.8068 | 155.8068 |  |
| 4 | 12 cm | 6 | VALUE ERROR | Fail |
|  |  | 113.04 |  |  |
|  |  | 37.68 |  |  |

## Task – 2.3:

**Problem Statement:**

Amount of Rs. 12800 was invested by Mr. Daniel dividing it into two different investment schemes A and B at a simple interest rate of 11% and 14%. Write a python program to find out what was the amount in plan B if the amount of interest earned in two years was Rs. 3508.

(Hint: Let the sum invested in Scheme B be Rs. x and that in Scheme A be Rs. (12800 – x), then, [x \* 14 \* 2]/100 + [(12800 – x) \* 11 \* 2]/100 = 3508)

## Program:

P = int (input ()) I = int (input ())

x = (100\*I - 22\*P) / 6

print(x)

## Output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | 12800  3508 | 11533.33333334 | 11533.33333334 | Pass |
| 2 | 42464  9500 | 2632.0 | 2632.0 | Pass |
| 3 | 95862  21500 | 6839.33333334 | 6839.33333334 | Pass |
| 4 | 563  128 | 69.0 | 69.0 | Pass |

## Task – 2.4:

**Problem Statement:**

A town has a population of P. The population increases by 10% per year. Write a python program to find out the population after 2 years?

(Hint: P=20,000, r=10/100, T=2, so the population after two years will be P[1+(r/100)] ^ T)

## Program:

P = int (input ()) r = 10

T = 2

Print (P \* (1 + (r / 100)) \*\* T)

## Output:

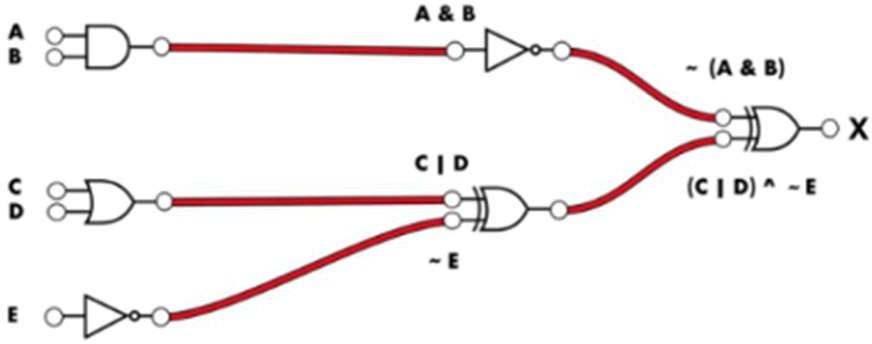
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | 20000 | 24200 | 24200 | Pass |
| 2 | 12345 | 14937.45 | 14937.45 | Pass |
| 3 | 42469 | 51387.49 | 51387.49 | Pass |
| 4 | 95862 | 115993.02 | 115993.02 | Pass |

## Task – 2.5:

**Problem Statement:**

Write a python program to find the value of X and print the output at each level of the given digital logic circuit. Also perform shift left and shift right operations.

(Hint: A = 12, B = 10, C = 8, D = 4, E = 5, print the output in digital form)



## Program:

A = int (input ())

B = int (input ())

C = int (input ())

D = int (input ())

E = int (input ()) A1 = A & B

B1 = C | D C1 = ~E A2 = ~A1

B2 = B1 ^ C1 X = A2 ^ B2

print(X)

## Output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | 12 | 1 | 1 | Pass |
|  | 10 |  |  |  |
|  | 8 |  |  |  |
|  | 4 |  |  |  |
|  | 5 |  |  |  |
| 2 | 62 | 47 | 47 | Pass |
|  | 58 |  |  |  |
|  | 49 |  |  |  |
|  | 46 |  |  |  |
|  | 42 |  |  |  |

# EXP.NO-3: Programs to demonstrate Programming constructs.

## Task – 3.1:

**Problem Statement:**

Write a python program to check whether you are eligible to vote or not?

Your program should get the age of the voter from the user and if their age is 18 and above, let them vote otherwise deny them from voting.

Hint: Use if-else, Enter voters age:17 Sorry! You are not eligible to vote

## Program:

n = int(input("")) if n >= 18:

print("Eligible for vote")

else:

## Output:

print("Sorry! You are not eligible to vote!")

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | 17 | Sorry! You are not  eligible to vote! | Sorry! You are not  eligible to vote! | Pass |
| 2 | 42 | Eligible for vote | Eligible for vote | Pass |
| 3 | 49.46 | Eligible for vote | Value Error | Fail |
| 4 | -58 | Value error | Sorry! You are not  eligible to vote! | Fail |

## Task – 3.2:

**Problem Statement:**

Write a python program that will check for the following conditions: If the light is green – Car is allowed to go

If the light is yellow – Car has to wait If the light is red – Car has to stop

Other signal – unrecognized signal. Example black, blue, etc

Hint: Use If, elif, and else; What is a traffic signal?: red Stop your car!

## Program:

colour = str(input(""))

if (colour == "red" or colour == "Red"): print("Car has to stop")

elif (colour == "yellow" or colour == "Yellow"): print("Car has to wait")

elif (colour == "green" or colour == "Green"): print("Car is allowed to go")

else:

## Output:

print("Unrecognized colour")

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | Red | Car has to stop | Car has to stop | Pass |
| 2 | green | Car is allowed to  go | Car is allowed to  go | Pass |
| 3 | Orange | Unrecognized  colour | Unrecognized  colour | Pass |
| 4 | 4246495862 | NO OUTPUT | Unrecognized  colour | Fail |

## Task – 3.3:

**Problem Statement:**

Your program should fulfill the following conditions: Grade A – Outstanding

Grade B – Excellent Grade C – Very Good Grade D – Good Grade E – Satisfactory others – Unrecognized

Hint: Use If, elif, and else; Enter student name: Raja Enter class: II B.Tech I sem Enter section:s1

Enter students grade:B Name: Raja

Class: II B.Tech I sem Section: S1

Grade: Excellent!

## Program:

name = str(input("Enter Student Name: ")) student\_class = str(input("Enter Class: ")) section = str(input("Enter Section: ")) grade = str(input("Enter Grade: ")) print("Name: ", name)

print("Class: ", student\_class) print("Section: ", section)

if (grade == 'a' or 'A'):

print("Grade: Out Standing") elif (grade == 'b' or 'B'):

print("Grade: Excellent") elif (grade == 'c' or 'C'):

print("Grade: Very good") elif (grade == 'd' or 'D'):

print("Grade: Good") elif (grade == 'e' or 'E'):

print("Grade: Satisfactory")

else:

print("Unrecognized Grade")

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result (Pass/**  **Fail)** |
| 1 | Enter Student Name: Srakhya  Enter Class: II B. Tech 1st semester  Enter Section: S1  Enter Grade: A | Name: Srakhya Class: II B. Tech 1st semester  Section: S1  Grade: Out Standing | Name: Srakhya Class: II B. Tech 1st semester  Section: S1  Grade: Out Standing | Pass |
| 2 | Enter Student Name: Navya  Enter Class: II B. Tech 2nd semester  Enter Section: S2  Enter Grade: b | Name: Navya Class: II B. Tech 2nd semester  Section: S2 Grade: Excellent | Name: Navya  Class: II B. Tech 2nd semester  Section: S2 Grade: Excellent | Pass |
| 3 | Enter Student Name: Dhanush  Enter Class: IV B. Tech 1st semester  Enter Section: S1  Enter Grade: F | Name: Dhanush Class: IV B. Tech 1st semester Section: S1  Grade: Satisfactory | Name: Srakhya Class: IV B. Tech 1st semester  Section: S1  Grade: Satisfactory | Pass |
| 4 | Enter Student Name: Likitha Ram  Enter Class: III B. Tech 1st semester  Enter Section: S1  Enter Grade: 4.2 | NO OUTPUT | Name: Likitha Ram Class: III B. Tech 1st semester  Section: S1  Grade: Unrecognized | Fail |

# EXP.NO-4: Programs to demonstrate decision making and branching (Selection).

## Task – 4.1:

**Problem Statement:**

Modify the earlier program students’ grades in such a way that they should take in five subject marks. Find the total mark and their percentage. Your program should check for the following conditions:

If the percentage falls below 45, they are considered fail.

If the percentage is between 45 and 60, grade them as pass. If the percentage is between 60 and 75, grade them as good.

If the percentage is between 75 and 85, grade them as very good. If the percentage is between 85 and 100, grade them excellent.

If the percentage is below zero or above 100, it’s an error.

Make sure to print their mark in every statement to prove that the condition is fulfilled.

Moreover, name, class, and section should be also displayed along with the marks and their grade.

## Program:

nameame = str(input("Enter student name : ")) student\_class = str(input("Enter Class : ")) section = str(input("Enter Section : "))

s1 = int(input("Enter Maths marks :")) s2 = int(input("Enter English marks :")) s3 = int(input("Enter Physics marks : "))

s4 = int(input("Enter Chemistry marks : ")) s5 = int(input("Enter Sanskrit marks : ")) marks = s1+s2+s3+s4+s5

percentage = (Total/5) print("Name : ",name) print("Class : ",student\_class) print("Section : ",section) print("percentage:-", percentage) if percentage < 45:

print("Status: Fail")

else:

print("Status: Pass")

if (percentage >= 45) and (percentage <= 60): print("Remarks : Pass")

elif (percentage > 60) and (percentage <= 75): print("Remarks : Good")

elif (percentage > 75) and (percentage <= 85): print("Remarks : Very good")

elif (percentage > 85) and (percentage <= 100): print("Remarks : Excellent")

else:

print("Error")

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result (Pass/**  **Fail)** |
| 1 | Enter Student Name: | Name: Srakhya | Name: Srakhya | Pass |
|  | Srakhya | Class: II B.Tech 1st | Class: II B.Tech 1st |  |
|  | Enter Class: II B.Tech 1st | semester | semester |  |
|  | semester | Section: S1 | Section: S1 |  |
|  | Enter Section: S1 | Percentage: 96.8 % | Percentage: 96.8 % |  |
|  | Enter Maths mark: 100 | Status: Pass | Status: Pass |  |
|  | Enter English mark: 95 | Remark: Excellent | Remark: Excellent |  |
|  | Enter Physics mark: 98 |  |  |  |
|  | Enter Chemistry mark: 92 |  |  |  |
|  | Enter Sanskrit mark: 99 |  |  |  |
| 2 | Enter Student Name: | Name: Navya | Name: Navya | Pass |
|  | Navya | Class: II B.Tech 1st | Class: II B.Tech 1st |  |
|  | Enter Class: II B.Tech 1st | semester | semester |  |
|  | semester | Section: S1 | Section: S1 |  |
|  | Enter Section: S1 | Percentage: 94.4 % | Percentage: 94.4 % |  |
|  | Enter Maths mark: 99 | Status: Pass | Status: Pass |  |
|  | Enter English mark: 92 | Remark: Excellent | Remark: Excellent |  |
|  | Enter Physics mark: 96 |  |  |  |
|  | Enter Chemistry mark: 90 |  |  |  |
|  | Enter Sanskrit mark: 95 |  |  |  |
| 3 | Enter Student Name: | Name: Dhanush | Name: Dhanush | Pass |
|  | Dhanush | Class: II B.Tech 1st | Class: II B.Tech 1st |  |
|  | Enter Class: II B.Tech 1st | semester | semester |  |
|  | semester | Section: S1 | Section: S1 |  |
|  | Enter Section: S1 | Percentage: 84.2 % | Percentage: 84.2 % |  |
|  | Enter Maths mark: 90 | Status: Pass | Status: Pass |  |
|  | Enter English mark: 90 | Remark: Very | Remark: Very |  |
|  | Enter Physics mark: 85 | Good | Good |  |
|  | Enter Chemistry mark: 80 |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Enter Sanskrit mark: 76 |  |  |  |
| 4 | Enter Student Name: | Name: Likitha | Value Error | Fail |
|  | Likitha | Class: II B.Tech 1st |  |  |
|  | Enter Class: II B.Tech 1st | semester |  |  |
|  | semester | Section: S1 |  |  |
|  | Enter Section: S1 | Percentage: 84.3 % |  |  |
|  | Enter Maths mark: 80.5 | Status: Pass |  |  |
|  | Enter English mark: 93 | Remark: Very |  |  |
|  | Enter Physics mark: 80 | Good |  |  |
|  | Enter Chemistry mark: 86 |  |  |  |
|  | Enter Sanskrit mark:82 |  |  |  |

**Task – 4.2:**

**Problem Statement:**

Anil has reached the finals of the NASA contest. For the finals, students were asked to prepare 20 topics. However, Anil was only able to prepare three topics, numbered A, B and C, he is totally blank about the other topics. This means Anil can only win the contest if he gets the topics A, B or C to speak about. On the contest day, Anil gets topic X. Determine whether Anil has any chances of winning the competition. Print "Yes" if it is possible for Anil to win the contest, else print "No".

## Program:

topic = str(input("Enter topic: "))

if ((topic == 'A') or (topic == 'B') or (topic == 'C') or (topic == 'a') or (topic == 'b') or (topic == 'c')):

print("Yes, there is a possibility for anil to win")

else:

print("No, Anil will fail the test")

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | a | Yes, there is a possibility for anil  to win | Yes, there is a possibility for anil  to win | Pass |
| 2 | S | Yes, there is a possibility for anil  to win | Yes, there is a possibility for anil  to win | Pass |
| 3 | SDJNL | Value Error | No, Anil will fail  the test | Fail |
| 4 | 42 | Value Error | No, Anil will fail  the test | Fail |

# EXP.NO-5: Programs to demonstrate iterative statements

## Task – 5.1:

**Problem Statement:**

Write a program in python to display the factorial of a number. (Hint: Input : 5; Expected output: 120)

## Program:

n = int(input()) factorial = 1

for i in range(1, (n+1)):

factorial \*= i print(factorial)

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test**  **Case#** | **Input** | **Expected**  **Output** | **Output After**  **Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | 5 | 120 | 120 | Pass |
| 2 | 42 | 1405006117752879 | 1405006117752879 | Pass |
|  |  | 8985431426062445 | 8985431426062445 |  |
|  |  | 1156993638400000 | 1156993638400000 |  |
|  |  | 0000 | 0000 |  |
| 3 | 5.8 | Value Error | Value Error | Fail |
| 4 | Six | Error | Invalid Input | Fail |

**Task – 5.2:**

**Problem Statement:**

A Prime Number is a number that cannot be made by multiplying other whole numbers. A prime number is a natural number greater than 1 that is not a product of two smaller

natural numbers.

Examples: 6 is not a prime mumber because it can be made by 2×3 = 6. 37 is a prime number because no other whole numbers multiply together to make it. Write a program to display all prime numbers within a range. (Hint: Prime numbers between 25 and 50 are: 29, 31, 37, 41, 43, 47).

## Program:

min = int(input("Enter the min : ")) max = int(input("Enter the max : ")) for n in range(min,max + 1):

if n > 1:

for i in range(2,n):

if (n % i) == 0:

break

else:

print(n)

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | 25  50 | 29 31 37 41 43 47 | 29 31 37 41 43 47 | Pass |
| 2 | 1  10 | 2 3 5 7 | 2 3 5 7 | Pass |
| 3 | 1  99999999 | 2 3 5 7 ……. | No Output | Fail |
| 4 | -12  -26 | No Output | No Output | Fail |

**Task – 5.3:**

**Problem Statement:**

The Fibonacci Sequence is a series of numbers. The next number is found by adding up the two numbers before it. The first two numbers are 0 and 1. For example, 0, 1, 1, 2, 3, 5, 8, 13, 21. The next number in this series above is 13+21 = 34. Write a python program to display Fibonacci series up to 10 terms.

(Hint: Set num1 = 0 and num2 =1 (first two numbers of the sequence). Run loop ten times. In each iteration, print num1 as the current number of the sequence. Add last two numbers to get the next number res = num1+ num2 update values of num1 and num2. Set num1=num2 and num2=res. Expected output: 0 1 1 2 3 5 8 13 21 34).

## Program:

n1 = 0

n2 = 1

i = 1

print("Fibonacci sequence:") while i<= 10:

print(n1)

next = n1 + n2 n1 = n2

n2 = next i += 1

**Output:**

Fibonacci sequence: 0

1

1

2

3

5

8

13

21

34

## Task – 5.4:

**Problem Statement:**

Print the following pattern using python 1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

## Program:

for i in range(1,6):

for j in range(1,i+1): print(j,end=' ')

print()

## Output:

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

## Task – 5.5:

**Problem Statement:**

Write a program to print the following start pattern using the for loop

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

## Program:

for i in range(1,6):

for j in range(1,i+1): print("\*",end=' ')

print()

for i in range(4,0,-1): for j in range(0,i):

print("\*",end=' ') print()

## Output:

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

# EXP.NO-6: Build modular programs using functions.

## Task – 6.1:

**Problem Statement:**

Let’s say we are going to a restaurant and we decide to leave a tip. We can create a function to easily calculate the amount to tip based on the total cost of the food and a percentage. This function will accept both of those values as inputs and return the amount of money to tip.

## Program:

def tip(a,p):

t = (a\*p) / 100 return t

amount = int(input("Enter total amount:")) percentage = int(input("Percentage to be tip:")) tip = tip(amount,percentage)

print("Tip : ",tip)

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test**  **Case#** | **Input** | **Expected**  **Output** | **Output**  **After Execution** | **Test Case**  **Result (Pass/ Fail)** |
| 1 | Enter total amount: 500  Percentage to be tip: 10 | 50 | 50 | Pass |
| 2 | Enter total amount: 1110  Percentage to be tip:20 | 222 | 222 | Pass |
| 3 | Enter total amount:195.3  Percentage to be tip:12 | 23.436 | Invalid literal | Fail |
| 4 | Enter total amount:0  Percentage to be tip:0 | 0 | 0 | Pass |

## Task – 6.2: Problem Statement:

Write a function, is\_vowel that returns the value true if a given character is a vowel, and otherwise returns false. Write another function main, in main() function accept a string from user and count number of vowels in that string.

## Program:

a=0

def vowel\_count(a): vowel=("aeiouAEIOU") a=input("Enter a string:") c=0

for alphabet in a:

if alphabet in vowel: c=c+1

print("No.of vowels :",c) vowel\_count(a)

## Output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | India is my  country | 6 | 6 | Pass |
| 2 | Aeiou | 5 | 5 | Pass |
| 3 | 134243 | Invalid input | 0 | Fail |
| 4 | As!23@#%! | Invalid Input | 1 | Fail |

**Task – 6.3: Problem Statement:**

Dog YearsSome say that every one year of a human’s life is equivalent to seven years of a dog’s life. Write a function named dog\_years() that has two parameters named name and age. The function should compute the age in dog years and return the following string: "{name}, you are {age} years old in dog years".

## Program:

def dog\_yrs(n,a):

age = a\*7

str1 = name + " you are "+ str(age)+" years old in dog years" return str1

name = input("Enter name:") age = int(input("Enterage:")) print(dog\_yrs(name,age))

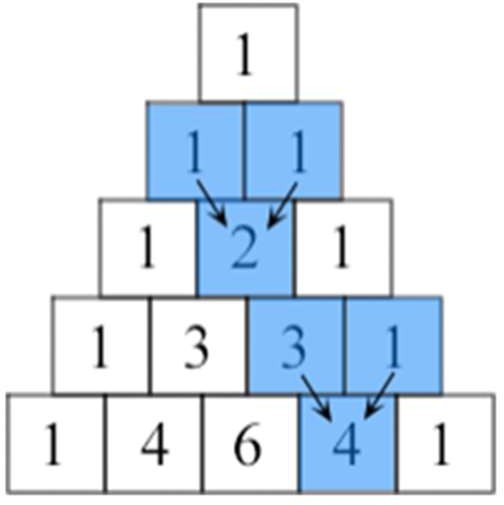
**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | Raju  20 | 140 | 140 | Pass |
| 2 | Ram  0 | 0 | 0 | Pass |
| 3 | Jayanth  -42 | Invalid age | -140 | Fail |
| 4 | 123  20 | Invalid name | 140 | Fail |

**Task – 6.4:**

**Problem Statement:**

Write a Python function that prints out the first n rows of Pascal's triangle. Note: Pascal's triangle is an arithmetic and geometric figure first imagined by Blaise Pascal. Sample Pascal's triangle:



## Program:

def pascal(n):

for i in range (0,n): str1=str(11\*\*i) for j in str1 :

print(j,end=" ") print()

n=int(input("n?")) pascal(n)

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test**  **Case#** | **Input** | **Expected**  **Output** | **Output After**  **Execution** | **Test Case**  **Result (Pass/ Fail)** |
| 1 | N? 4 | 1 | 1 | Pass |
|  |  | 1 1 | 1 1 |  |
|  |  | 1 2 1 | 1 2 1 |  |
|  |  | 1 3 3 1 | 1 3 3 1 |  |
| 2 | N? 6 | 1 | 1 | Pass |
|  |  | 1 1 | 1 1 |  |
|  |  | 1 2 1 | 1 2 1 |  |
|  |  | 1 3 3 1 | 1 3 3 1 |  |
|  |  | 1 4 6 4 1 | 1 4 6 4 1 |  |
|  |  | 1 6 1 0 5 1 | 1 6 1 0 5 1 |  |
| 3 | N? a | No Output | Value Error | Fail |
| 4 | N? 0.5 | No Output | Value Error | Fail |

## Task – 6.5: Problem Statement:

Print multiplication table of 12 using recursion

## Program:

def multiply(n, i): print(n," X ",i," = ",n \* i) if (i < 10):

multiply(n, i + 1) print("Multiplication Table of 12 is:") multiply(12, 1)

## Output:

Multiplication Table of 12 is:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 12 | X | 1 | = | 12 |
| 12 | X | 2 | = | 24 |
| 12 | X | 3 | = | 36 |
| 12 | X | 4 | = | 48 |
| 12 | X | 5 | = | 60 |
| 12 | X | 6 | = | 72 |
| 12 | X | 7 | = | 84 |
| 12 | X | 8 | = | 96 |
| 12 | X | 9 | = | 108 |
| 12 | X | 10 = 120 | | |

# EXP.NO-7: Programs to perform operations on strings, regular expressions with built – in functions.

## Task – 7.1: Problem Statement:

Write a python program to Reverse a given string

## Program:

str1 = "PYnative" print("Original String is:", str1) str1 = "".join(reversed(str1)) print("Reversed String is:", str1)

## Output:

Original String is: PYnative Reversed String is: evitanYP

## Task – 7.2: Problem Statement:

Write a program to arrange the characters of a string so that all lowercase letters should come first. Given string contains a combination of the lower-case and upper-case letters.

## Program:

str1 = input("str1=") lower = []

upper = []

for char in str1:

if char.islower(): lower.append(char)

else:

upper.append(char) final= ''.join(lower + upper) print(final)

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | PyNaTive | yaivePNT | yaivePNT | Pass |
| 2 | KjSdjNL | jdjKSNL | jdjKSNL | Pass |
| 3 | 12345 | No Output | 12345 | Fail |
| 4 | !@#$% | No Output | !@#$% | Fail |

**Task – 7.3:**

**Problem Statement:**

Write a python program to count all letters, digits, and special symbols from a given string.

## Program:

str1=input("str1=") alpha=digits=special=0 for i in range(len(str1)):

if(str1[i].isalpha()): alpha = alpha + 1 elif(str1[i].isdigit()): digits = digits + 1

else:

special = special + 1 print("Char:", alpha) print("Digits:", digits) print("Special:", special)

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result**  **(Pass/ Fail)** |
| 1 | P@#yn26at^&i5ve | Char: 8  Digits: 3  Special: 4 | Char: 8  Digits: 3  Special: 4 | Pass |
| 2 | K!^j38S$@ | Char: 3  Digits: 2  Special: 4 | Char: 3  Digits: 2  Special: 4 | Pass |
| 3 | S$@d46j49N%\*L^@ | Char: 5  Digits: 4  Special: 6 | Char: 5  Digits: 4  Special: 6 | Pass |
| 4 | A%&R^$ | Char: 2  Digits: 0  Special: 4 | Char: 2  Digits: 0  Special: 4 | Pass |

**Task – 7.4:**

**Problem Statement:**

Write a python program for verification of phone number and Email ID

## Program:

import re

def phone(phn):

if re.match('w{3}-w{3}-w{4}', phn): print("Valid")

else:

print("Invalid") def email(mail):

pat = r'\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b' if re.search(pat,mail):

print("Valid") else:

print("Invalid")

phn = input("Phone number?") phone(phn)

mail = input("mail?") email(mail)

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result (Pass/**  **Fail)** |
| 1 | 123-456-9876  [21501a0500@pvpsit.ac.in](mailto:21501a0500@pvpsit.ac.in) | Valid  Valid | Valid  Valid | Pass |
| 2 | 111-11-123  [Tarak78@gmail.com](mailto:Tarak78@gmail.com) | Invalid  Valid | Valid  Valid | Pass |
| 3 | 12344-324  Raj%78gmail.com | Invalid  Invalid | Valid  Valid | Pass |
| 4 | 123-456-5432  [1234@345mail.cm](mailto:1234@345mail.cm) | Valid  Invalid | Valid  Valid | Pass |

**Task – 7.5:**

**Problem Statement:**

Write a function that employs regular expressions to ensure the password given to the function is strong.

## Program:

import re

def uppercase\_check(password): if re.search('[A-Z]', password):

return True return False

def lowercase\_check(password): if re.search('[a-z]', password):

return True return False

def digit\_check(password):

if re.search('[0-9]', password): return True

return False

def user\_input\_password\_check(): password = input("Enter password : ")

if len(password) >= 8 and uppercase\_check(password) and lowercase\_check(password) and digit\_check(password):

print("Strong Password") else:

print("Weak Password") user\_input\_password\_check()

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result (Pass/**  **Fail)** |
| 1 | gurr@004 | Strong  Password | Strong  Password | Pass |
| 2 | Kjhg`1234 | Weak  Password | Weak  Password | Pass |
| 3 | Kjs@7116 | Strong  Password | Strong  Password | Pass |
| 4 | Aerty!@#$^ | Weak  Password | Weak  Password | Pass |

# EXP.NO-8: Implement programs using various data structures.

## Task – 8.1:

**Problem Statement:**

* Write a program to remove the item present at index 4 and add it to the 2nd position and at the end of the list.

## Program:

l = [54, 44, 27, 79, 91, 41]

a = l.pop(4)

print("List After removing element at index 4 ", l) l.insert(2, a)

print("List after Adding element at index 2 ", l) l.append(a)

print("List after Adding element at last ", l)

## Output:

List After removing element at index 4 [54, 44, 27, 79, 41]

List after Adding element at index 2 [54, 44, 91, 27, 79, 41]

List after Adding element at last [54, 44, 91, 27, 79, 41, 91]

## Task – 8.2:

**Problem Statement:**

Sort a tuple of tuples by 2nd item

## Program:

tuple=(('a', 23),('b', 37),('c', 11), ('d',29))

sorted\_list = sorted(tuple,key=lambda t: t[1]) print(sorted\_list)

## Output:

[('c', 11), ('a', 23), ('d', 29), ('b', 37)]

## Task – 8.3:

**Problem Statement:**

Create a Python set such that it shows the element from both lists in a pair.

## Program:

first\_list = [2, 3, 4, 5, 6, 7, 8]

second\_list = [4, 9, 16, 25, 36, 49, 64] result = zip(first\_list, second\_list) result\_set = set(result)

print("Result is ", result\_set)

## Output:

Result is {(7, 49), (2, 4), (4, 16), (8, 64), (6, 36), (3, 9), (5, 25)}

## Task – 8.4:

**Problem Statement:**

Iterate a given list and check if a given element exists as a key’s value in a dictionary. If not, delete it from the list.

## Program:

rollNumber = [47, 64, 69, 37, 76, 83, 95, 96, 97]

sampleDict = {'John':47, 'Emma':69, 'Kelly':76, 'Jason':97}

for current\_item in rollNumber:

if not current\_item in sampleDict.values(): rollNumber.remove(current\_item)

print("After removing unwanted elements from list ", rollNumber)

## Output:

After removing unwanted elements from list [47, 69, 76, 95, 97]

## Task – 8.5:

**Problem Statement:**

Write a Python program to get all values from the dictionary and add them to a list but don’t add duplicates

## Program:

speed = {'jan': 47, 'feb': 52, 'march': 47, 'April': 44, 'May': 52, 'June': 53,

'july': 54, 'Aug': 44, 'Sept': 54} speed\_list=list()

for val in speed.values(): if val not in speed\_list:

speed\_list.append(val) print(speed\_list)

## Output:

[47, 52, 44, 53, 54]

# EXP.NO-9: Implement programs using various data structures.

## Task – 9.1:

**Problem Statement:**

Write a python program to find the longest words.

## Program:

def longest\_word(filename):

with open(filename, 'r') as infile: words = infile.read().split()

max\_len = len(max(words, key=len))

return [word for word in words if len(word) == max\_len] print(longest\_word("text.txt"))

## Output:

['general-purpose,', 'object-oriented,']

## Task – 9.2:

**Problem Statement:**

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

## Program:

def file\_read(fname): content\_array = [] with open(fname) as f:

for line in f: content\_array.append(line)

print(content\_array) file\_read('text.txt')

## Output:

['What is Python language?\n', 'Python is a widely used high-level, general-purpose, interpreted, dynamic\n', 'programming language. Its design philosophy emphasizes code readability, and\n', 'its syntax allows programmers to express concepts in fewer lines of code than\n', 'possible in languages such as C++ or Java.\n', 'Python supports multiple programming paradigms, including object-oriented,\n', 'imperative and functional programming or procedural styles.It features a dynamic\n', 'type system andautomatic memory management and has a large and\n', 'comprehensive standard library.The best way we learn anything is by practice and\n', 'exercise questions. We have started this section for those (beginner to\n', 'intermediate) who are familiar with Python.']

## Task – 9.3:

**Problem Statement:**

Write a Python program to count the frequency of words in a file

## Program:

from collections import Counter def word\_count(fname):

with open(fname) as f:

return Counter(f.read().split())

print("Number of words in the file :",word\_count("text.txt"))

## Output:

Number of words in the file : Counter({'and': 5, 'is': 3, 'Python': 3, 'a': 3, 'programming': 3, 'dynamic': 2, 'code': 2, 'to': 2, 'in': 2, 'or': 2, 'What': 1,

'language?': 1, 'widely': 1, 'used': 1, 'high-level,': 1, 'general-purpose,': 1,

'interpreted,': 1, 'language.': 1, 'Its': 1, 'design': 1, 'philosophy': 1, 'emphasizes': 1,

'readability,': 1, 'its': 1, 'syntax': 1, 'allows': 1, 'programmers': 1, 'express': 1,

'concepts': 1, 'fewer': 1, 'lines': 1, 'of': 1, 'than': 1, 'possible': 1, 'languages': 1,

'such': 1, 'as': 1, 'C++': 1, 'Java.': 1, 'supports': 1, 'multiple': 1, 'paradigms,': 1,

'including': 1, 'object-oriented,': 1, 'imperative': 1, 'functional': 1, 'procedural': 1,

'styles.It': 1, 'features': 1, 'type': 1, 'system': 1, 'andautomatic': 1, 'memory': 1,

'management': 1, 'has': 1, 'large': 1, 'comprehensive': 1, 'standard': 1,

'library.The': 1, 'best': 1, 'way': 1, 'we': 1, 'learn': 1, 'anything': 1, 'by': 1,

'practice': 1, 'exercise': 1, 'questions.': 1, 'We': 1, 'have': 1, 'started': 1, 'this': 1,

'section': 1, 'for': 1, 'those': 1, '(beginner': 1, 'intermediate)': 1, 'who': 1, 'are': 1,

'familiar': 1, 'with': 1, 'Python.': 1})

## Task – 9.4:

**Problem Statement:**

Write a Python program to read a random line from a file

## Program:

import random

def random\_line(fname):

lines = open(fname).read().splitlines() return random.choice(lines)

print(random\_line('text.txt'))

## Output:

type system andautomatic memory management and has a large and

## Task – 9.5:

**Problem Statement:**

Write a Python program that takes a text file as input and returns the number of words of a given text file.

## Program:

def count\_words(filepath): with open(filepath) as f:

data = f.read() data.replace(",", " ")

return len(data.split(" ")) print(count\_words("text.txt"))

## Output:

89

# EXP.NO-10: Python programming to demonstrate Exception handling.

## Task – 10.1:

**Problem Statement:**

You’re going to write an interactive calculator! User input is assumed to be a formula that consist of a number, an operator (at least + and -), and another number, separated by white space (e.g. 1 + 1). Split user input using str.split(), and check whether the resulting list is valid:

If the input does not consist of 3 elements, raise a FormulaError, which is a custom Exception. Try to convert the first and third input to a float

(like so: float\_value = float(str\_value)).

Catch any ValueError that occurs, and instead raise a FormulaError If the second input is not ‘+’ or ‘-’, again raise a FormulaError

If the input is valid, perform the calculation and print out the result. The user is then prompted to provide new input, and so on, until the user enters quit.

## Program:

class FormulaError(Exception): pass def parse\_input(user\_input):

input\_list = user\_input.split() if len(input\_list) != 3:

raise FormulaError('Input does not consist of three elements') n1, op, n2 = input\_list

try:

n1 = float(n1) n2 = float(n2)

except ValueError:

raise FormulaError('The first and third input value must be numbers') return n1, op, n2

def calculate(n1, op, n2): if op == '+':

return n1 + n2 if op == '-': return n1 - n2 if op == '\*': return n1 \* n2

if op == '/':

return n1 / n2

raise FormulaError('{0} is not a valid operator'.format(op)) while True:

user\_input = input('>>> ') if user\_input == 'quit':

break

n1, op, n2 = parse\_input(user\_input) result = calculate(n1, op, n2) print(result)

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result (Pass/**  **Fail)** |
| 1 | 42 + 58 | 100.0 | 100.0 | Pass |
| 2 | 57 \* 64 | 3648.0 | 3648.0 | Pass |
| 3 | 46-62 | -16 | Formula Error | Fail |
| 4 | 49 / 0 | Error | Zero Division  Error | Fail |

**Task – 10.2:**

**Problem Statement:**

Create a short program that prompts the user for a list of grades separated by commas. Split the string into individual grades and use a list comprehension to convert each string to an integer. You should use a try statement to inform the user when the values they entered cannot be converted.

Let’s start by writing our input call to grab the list of grades. We can also use the split method to split the string we get back from input all on the same line.

## Program:

grades = input("Please enter your grades, separated by commas: ").split(",") try:

grades = [int(grade) for grade in grades] print('Valid Grades')

except ValueError: print("Invalid Grades")

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case#** | **Input** | **Expected Output** | **Output After Execution** | **Test Case Result (Pass/**  **Fail)** |
| 1 | 10, 10, 10 | Valid Grades | Valid Grades | Pass |
| 2 | A, B, C | Invalid Grades | Invalid Grades | Pass |
| 3 | 9.5, 8, 9 | Valid Grades | Invalid Grades | Fail |
| 4 | A, 10, C | Invalid Grades | Invalid Grades | Pass |

**Task – 10.3:**

**Problem Statement:**

Imagine you have a file named data.txt. Open it for reading using python, but make sure to use a try block to catch an exception that arises if the file doesn’t exist. Once you have verified your solution works with actual file, delete the file and see if your try block is able to handle it.

## Program:

try:

with open("text.txt", "r") as text\_file: print(text\_file.read())

except FileNotFoundError: print("Error: Couldn't find data.txt")

## Output:

#### Displays data int the file

(What is Python language? Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles.It features a dynamic type system andautomatic memory management and has a large and comprehensive standard library.The best way we learn anything is by practice and exercise questions. We have started this section for those (beginner to intermediate) who are familiar with Python.)

1. **Error: Couldn't find data.txt**

**EXP.NO-11: Programs to demonstrate access specifiers.**

## Task – 11.1:

**Problem Statement:**

Write a Program to create a class by name Students, and initialize attributes like name, age, and grade while creating an object.

## Program:

class Students:

def init (self, name, age, grade): self.name = name

self.age = age self.grade = grade

def display(self): print("Name: ", self.name) print("Age: ", self.age) print("Grade: ", self.grade)

kjs = Students('kjs', 57, 'A') kjs.display()

## Output:

(The details of the student are printed in the output) Name: kjs

Age: 57 Grade: A

## Task – 11.2:

**Problem Statement:**

Create a class Teacher with name, age, and salary attributes, where salary must be a private attribute that cannot be accessed outside the class.

## Program:

class Teacher():

def init (self, name, age, salary): self.name = name

self.age = age

self. salary = salary def show\_details(self):

print("Name:", self.name) print("Age:", self.age) print("Salary: ", self. salary)

teacher = Teacher("KJS", 57, 100000) teacher.show\_details()

## Output:

(The details of the teacher are printed in the output) Name: KJS

Age: 57

Salary: 100000

## Task – 11.3:

**Problem Statement:**

Create a Python class called BankAccount which represents a bank account, having as attributes: accountNumber (numeric type), name (name of the account owner as string type), balance. Create a constructor with parameters: accountNumber, name, balance. Create a Deposit() method which manages the deposit actions. Create a Withdrawal() method which manages withdrawals actions. Create an bankFees() method to apply the bank fees with a percentage of 5% of the balance account. Create a display() method to display account details.

## Program:

class BankAccount:

def init (self,accountNumber, name, balance): self.accountNumber = accountNumber self.name = name

self.balance = balance def Deposit(self , d ):

self.balance = self.balance + d def Withdrawal(self , w):

if(self.balance < w):

print("impossible operation! Insufficient balance !") else:

self.balance = self.balance - w def bankFees(self):

self.balance = (95/100)\*self.balance def display(self):

print("Account Number : " , self.accountNumber) print("Account Name : " , self.name) print("Account Balance : " , self.balance, "/-")

newAccount = BankAccount(163842, "KJS" , 2700) newAccount.Withdrawal(300) newAccount.Deposit(200)

newAccount.bankFees() newAccount.display()

## Output:

Account Number : 163842 Account Name : KJS Account Balance : 2470.0 /-

# EXP.NO-12: Programs to demonstrate types of Inheritance and polymorphism.

## Task – 12.1:

**Problem Statement:**

Create a Bus child class that inherits from the Vehicle class. The default fare charge of any vehicle is seating capacity \* 100. If Vehicle is Bus instance, we need to add an extra 10% on full fare as a maintenance charge. So total fare for bus instance will become the final amount = total fare + 10% of the total fare.

## Program:

class Vehicle:

def init (self, name, mileage, capacity): self.name = name

self.mileage = mileage self.capacity = capacity

def fare(self):

return self.capacity \* 100

class Bus(Vehicle):

def fare(self):

fare\_car = self.capacity \* 100 total\_fare = fare\_car + (0.1 \*fare\_car) return total\_fare

School\_bus = Bus("KJS Garuda", 16, 38) print("Total Bus fare is:", School\_bus.fare())

## Output:

Total Bus fare is: 4180.0

## Task – 12.2:

**Problem Statement:**

Class Person and child class Student: Create a Python class Person with attributes: name and age of type string. Create a display() method that displays the name and age of an object created via the Person class. Create a child class Student which inherits from the Person class and which also has a section attribute. Create a method displayStudent() that displays the name, age and section of an object created via the Student class. Create a student object via an instantiation on the Student class and then test the displayStudent method.

## Program:

class Person:

def init (self, name, age): self.name = name

self.age = age def display(self):

print("Person name : ", self.name) print("Person age = ", self.age)

class Student(Person):

def init (self, name , age , section): Person. init (self,name, age) self.section = section

def displayStudent(self): print("Student name : ", self.name) print("Student age = ", self.age)

print("Student section = ", self.section) P = Person("Keerthi", 21)

P.display()

print(" ")

S = Student("Jayanth", 19 , "Mathematics") S.displayStudent()

## Output:

Person name : Tomas Wild Person age = 37

Student name : Albert Student age = 23

Student section = Mathematics

## Task – 12.3:

**Problem Statement:**

Write a Python program that overloads the operator + and > for a custom class.

## Program:

class Operators:

def init (self, items): self.items = items

def add (self, other):

return self.items + other.items def gt (self, other):

return len(self.items) > len(other.items) k = 16

j = 38

print("k + j =", k + j) print("k > j =", k > j)

## Output:

k + j = 54

k > j = False

# EXP.NO-13: Installing, importing accessing and computations on a dataset using Numpy library.

## Task – 13.1:

**Problem Statement:**

Create a 4X2 integer array and Prints its attributes. The element must be a type of unsigned int16 and print the following Attributes: –

1. The shape of an array.
2. Array dimensions.
3. The Length of each element of the array in bytes.

## Program:

import numpy

firstArray = numpy.empty([4,2], dtype = numpy.uint16) print("Printing Array")

print(firstArray)

print("Printing numpy array Attributes") print("1> Array Shape is: ", firstArray.shape)

print("2>. Array dimensions are ", firstArray.ndim)

print("3>. Length of each element of array in bytes is ", firstArray.itemsize)

## Output:

Printing Array [[ 1344 55024]

[53692 35010]

[63863 62139]

[58628 12504]]

Printing numpy array Attributes 1> Array Shape is: (4, 2)

2>. Array dimensions are 2

3>. Length of each element of array in bytes is 2

## Task – 13.2:

**Problem Statement:**

Create a 5X2 integer array from a range between 100 to 200 such that the difference between each element is 10.

## Program:

import numpy

print("Creating 5X2 array using numpy.arange") sampleArray = numpy.arange(100, 200, 10) sampleArray = sampleArray.reshape(5,2)

print (sampleArray)

## Output:

Creating 5X2 array using numpy.arange [[100 110]

[120 130]

[140 150]

[160 170]

[180 190]]

## Task – 13.3:

**Problem Statement:**

Create a result array by adding the following two NumPy arrays. Next, modify the result array by calculating the square of each element

## Program:

import numpy

arrayOne = numpy.array([[5, 6, 9], [21 ,18, 27]])

arrayTwo = numpy.array([[15 ,33, 24], [4 ,7, 1]])

resultArray = arrayOne + arrayTwo print("addition of two arrays is \n") print(resultArray)

for num in numpy.nditer(resultArray, op\_flags = ['readwrite']): num[...] = num\*num

print("\nResult array after calculating the square root of all elements\n") print(resultArray)

## Output:

addition of two arrays is

[[20 39 33]

[25 25 28]]

Result array after calculating the square root of all elements

[[ 400 1521 1089]

[ 625 625 784]]

# EXP.NO-14: Installing, importing accessing and computations on a dataset using Pandas library.

## Task – 14.1:

**Problem Statement:**

Load the Automobile dataset for data analysis. This Dataset has different characteristics of an auto such as body-style, wheel-base, engine-type, price, mileage, horsepower, etc. From the given dataset print the first and last five rows.

## Program:

1. Program to print first five lines of the data import pandas as pd

df = pd.read\_csv("C:/Users/tarak/OneDrive/Desktop/Python files/Automobile\_data.csv")

df.head(5)

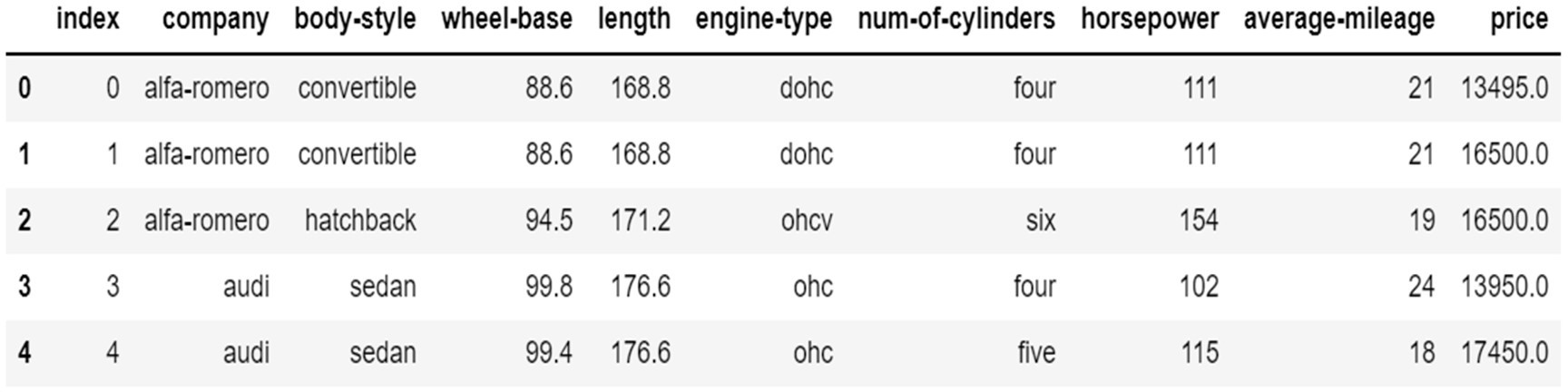
1. Program to print last five lines of the data import pandas as pd

df = pd.read\_csv("C:/Users/tarak/OneDrive/Desktop/Python files/Automobile\_data.csv")

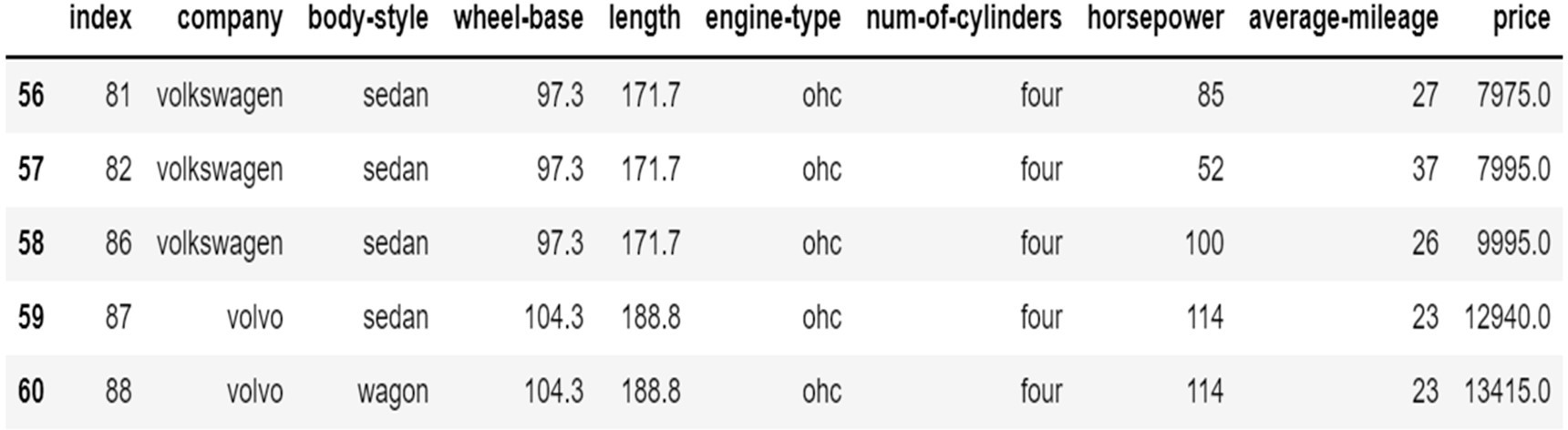
df.tail(5)

## Output:

First five lines of data



Last five lines of data



## Task – 14.2:

**Problem Statement:**

Load the Automobile dataset and print the total number of cars per company.

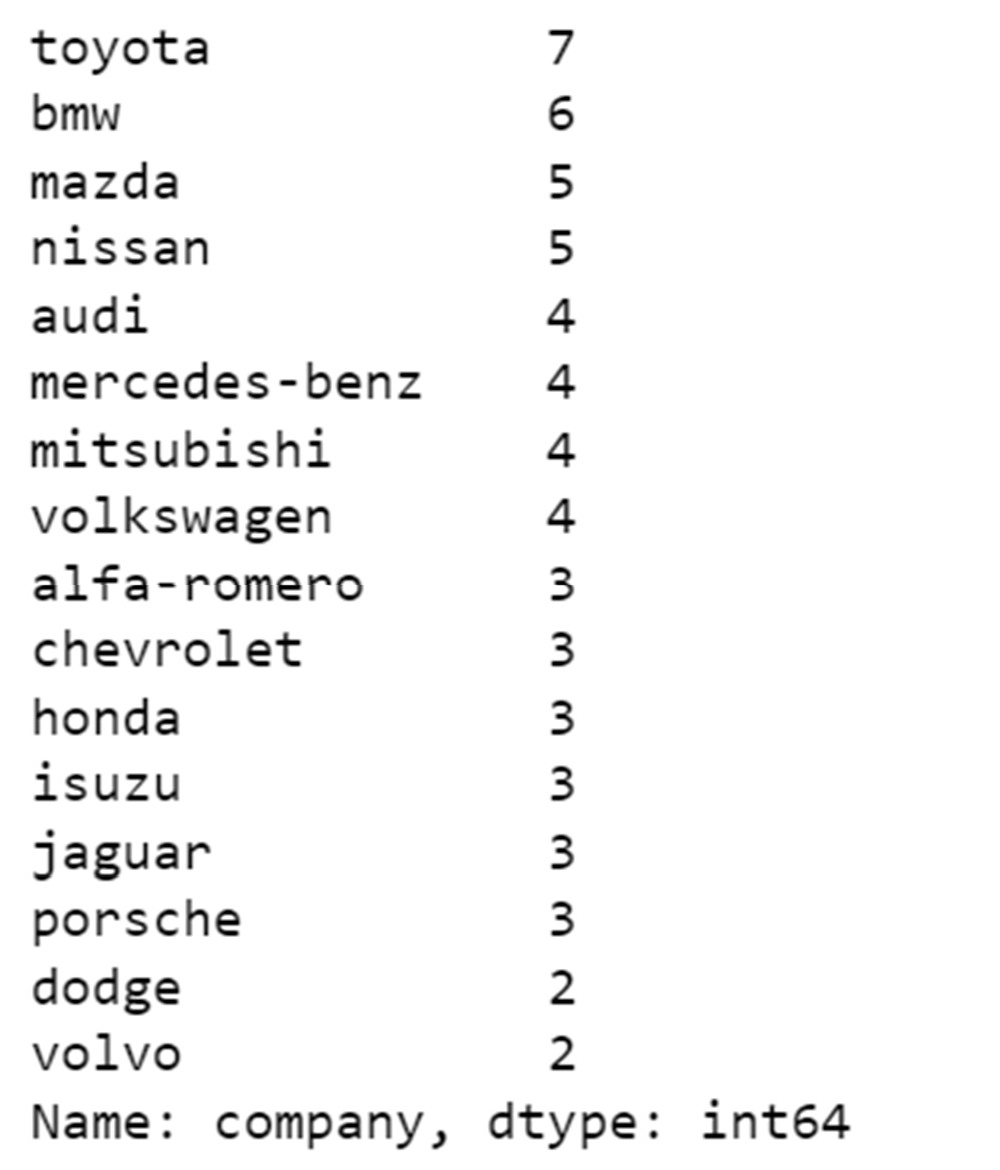
## Program:

import pandas as pd

df = pd.read\_csv("C:/Users/tarak/OneDrive/Desktop/Python files/Automobile\_data.csv")

df['company'].value\_counts()

**Output:**

****

**Task – 14.3:**

**Problem Statement:**

Load the Automobile dataset and find each company’s Highest price car.

## Program:

import pandas as pd

df = pd.read\_csv("C:/Users/tarak/OneDrive/Desktop/Python files/Automobile\_data.csv")

car\_Manufacturers = df.groupby('company')

priceDf = car\_Manufacturers['company','price'].max() priceDf

**Output:**

****

# EXP.NO-15: Installing, importing accessing and computations on a dataset using MatplotLib library.

## Task – 15.1:

**Problem Statement:**

Read Total profit of all months and show it using a line plot. Generated line plot must include the following properties:

X label name = Month Number Y label name = Total profit

## Program:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv("C:/Users/tarak/OneDrive/Desktop/Python files/company\_sales\_data.csv")

profitList = df ['total\_profit'].tolist() monthList = df ['month\_number'].tolist()

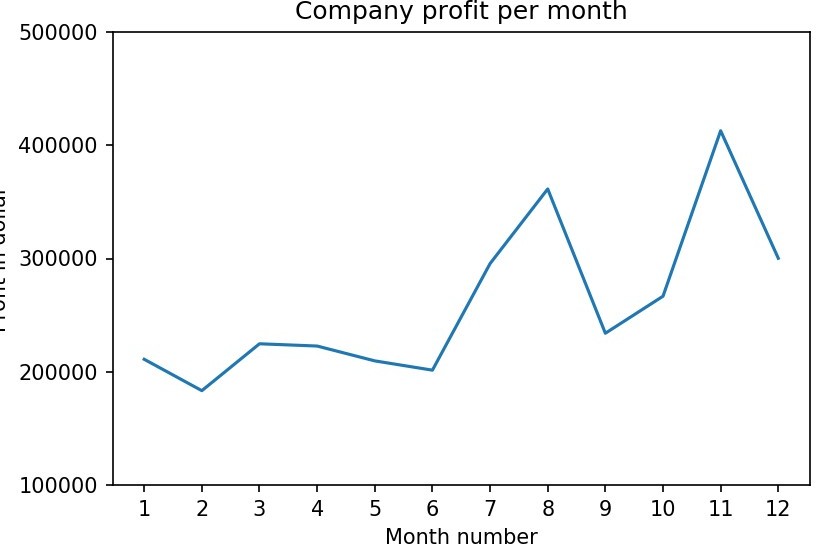
plt.plot(monthList, profitList, label = 'Month-wise Profit data of last year') plt.xlabel('Month number')

plt.ylabel('Profit in dollar') plt.xticks(monthList) plt.title('Company profit per month')

plt.yticks([100000, 200000, 300000, 400000, 500000])

plt.show()

**Output:**

****

**Task – 15.2:**

**Problem Statement:**

Read all product sales data and show it using a multiline plot. Display the number of units sold per month for each product using multiline plots. (i.e., Separate Plotline for each product).

## Program:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv("C:/Users/tarak/OneDrive/Desktop/Python files/company\_sales\_data.csv")

monthList = df ['month\_number'].tolist() faceCremSalesData = df ['facecream'].tolist() faceWashSalesData = df ['facewash'].tolist() toothPasteSalesData = df ['toothpaste'].tolist() bathingsoapSalesData = df ['bathingsoap'].tolist() shampooSalesData = df ['shampoo'].tolist() moisturizerSalesData = df ['moisturizer'].tolist()

plt.plot(monthList, faceCremSalesData, label = 'Face cream Sales Data', marker='o', linewidth=3)

plt.plot(monthList, faceWashSalesData, label = 'Face Wash Sales Data', marker='o', linewidth=3)

plt.plot(monthList, toothPasteSalesData, label = 'ToothPaste Sales Data', marker='o', linewidth=3)

plt.plot(monthList, bathingsoapSalesData, label = 'ToothPaste Sales Data', marker='o', linewidth=3)

plt.plot(monthList, shampooSalesData, label = 'ToothPaste Sales Data', marker='o', linewidth=3)

plt.plot(monthList, moisturizerSalesData, label = 'ToothPaste Sales Data', marker='o', linewidth=3)

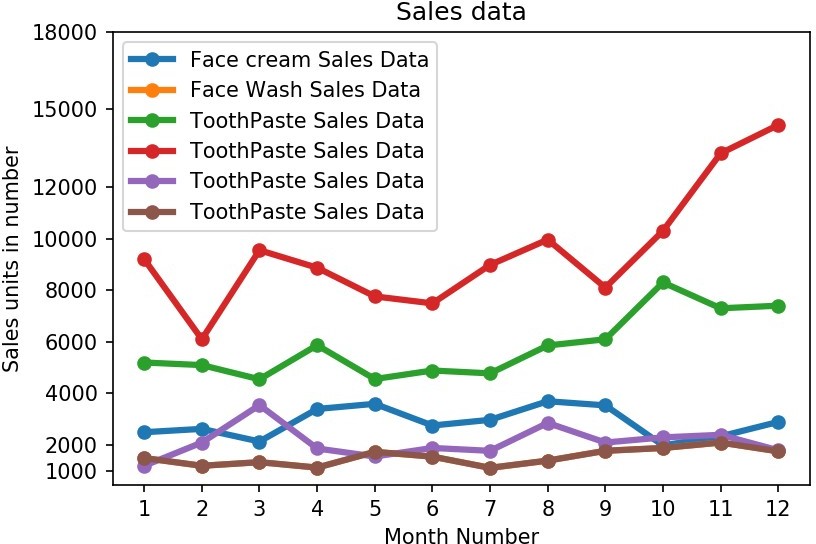
plt.xlabel('Month Number')

plt.ylabel('Sales units in number') plt.legend(loc='upper left') plt.xticks(monthList)

plt.yticks([1000, 2000, 4000, 6000, 8000, 10000, 12000, 15000, 18000])

plt.title('Sales data') plt.show()

**Output:**

****

**Task – 15.3:**

**Problem Statement:**

Read face cream and facewash product sales data and show it using the bar chart. The bar chart should display the number of units sold per month for each product. Add a separate bar for each product in the same chart.

## Program:

import pandas as pd

import matplotlib.pyplot as plt

df=pd.read\_csv("C:/Users/tarak/OneDrive/Desktop/Python files/company\_sales\_data.csv")

monthList = df ['month\_number'].tolist() faceCremSalesData = df ['facecream'].tolist() faceWashSalesData = df ['facewash'].tolist()

plt.bar([a-0.25 for a in monthList], faceCremSalesData, width= 0.25, label = 'Face Cream sales data', align='edge')

plt.bar([a+0.25 for a in monthList], faceWashSalesData, width= -0.25, label = 'Face Wash sales data', align='edge')

plt.xlabel('Month Number') plt.ylabel('Sales units in number') plt.legend(loc='upper left') plt.title(' Sales data')

plt.xticks(monthList)

plt.grid(True, linewidth= 1, linestyle="--") plt.title('Facewash and facecream sales data') plt.show()

**Output:**

