APPLICATION DEVELOPMENT WITH PYTHON

LECTURE NOTES

II-B.Tech & I-SEM

Department of Electronics and Communication Engineering

VEMU INSTITUTE OF TECHNOLOGY::P.KOTHAKOTA

Chittoor – Tirupathi National Highway, Near Pakala, Chittoor(Dt.), AP – 517112

(Accredited by NAAC & NBA (CSE, ECE, EEE))

(Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Anantapuramu)

MODULE-1

Abstraction:

It refers to the construction of a simpler version of a problem by ignoring the details. The principle of constructing an abstraction is popularly known as **modelling**.

It is the simplification of a problem by focusing on only one aspect of the problem while omitting all other aspects. When using the principle of abstraction to understand a complex problem, we focus our attention on only one or two specific aspects of the problem and ignore the rest.

Whenever we omit some details of a problem to construct an abstraction, we construct a model of the problem. In everyday life, we use the principle of abstraction frequently to understand a problem or to assess a situation.

Decomposition:

Decomposition is a process of breaking down. It will be breaking down functions into smaller parts. It is another important principle of software engineering to handle problem complexity. This principle is profusely made use by several software engineering techniques to contain the exponential growth of the perceived problem complexity. The decomposition principle is popularly is says the **divide and conquer principle**.

Functional Decomposition:

It is a term that engineers use to describe a set of steps in which they break down the overall function of a device, system, or process into its smaller parts.

Steps for the Functional Decomposition:

1. Find the most general function

2. Find the closest sub-functions

3. Find the next levels of sub-functions

SDLC:

Software Development Life Cycle, SDLC for short, is a well-defined, structured sequence of stages in software engineering to develop the intended software product.

SDLC Activities

SDLC provides a series of steps to be followed to design and develop a software product efficiently. SDLC framework includes the following steps:
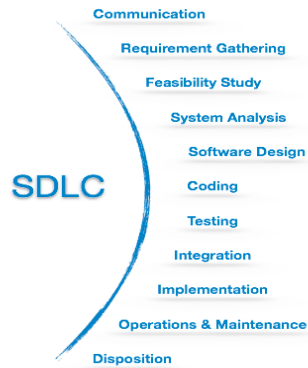
Communication

This is the first step where the user initiates the request for a desired software product. He contacts the service provider and tries to negotiate the terms. He submits his request to the service providing organization in writing.

Requirement Gathering

This step onwards the software development team works to carry on the project. The team holds discussions with various stakeholders from problem domain and tries to bring out as much information as possible on their requirements. The requirements are contemplated and segregated into user requirements, system requirements and functional requirements. The requirements are collected using a number of practices as given -

- studying the existing or obsolete system and software,
- conducting interviews of users and developers,
- referring to the database or
- collecting answers from the questionnaires.

Feasibility Study

After requirement gathering, the team comes up with a rough plan of software process. At this step the team analyzes if a software can be made to fulfill all requirements of the user and if there is any possibility of software being no more useful. It is found out, if the project is financially, practically and technologically feasible for the organization to take up. There are many algorithms available, which help the developers to conclude the feasibility of a software project.

System Analysis

At this step the developers decide a roadmap of their plan and try to bring up the best software model suitable for the project. System analysis includes Understanding of software product limitations, learning system related problems or changes to be done in existing systems beforehand, identifying and addressing the impact of project on organization and personnel etc. The project team analyzes the scope of the project and plans the schedule and resources accordingly.

Software Design

Next step is to bring down whole knowledge of requirements and analysis on the desk and design the software product. The inputs from users and information gathered in requirement gathering phase are the inputs of this step. The output of this step comes in the form of two designs; logical design and physical design. Engineers produce meta-data and data dictionaries, logical diagrams, data-flow diagrams and in some cases pseudo codes.

Coding

This step is also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently.

Testing

An estimate says that 50% of whole software development process should be tested. Errors may ruin the software from critical level to its own removal. Software testing is done while coding by the developers and thorough testing is conducted by testing experts at various levels of code such as module testing, program testing, product testing, in-house testing and testing the product at user's end. Early discovery of errors and their remedy is the key to reliable software.

Integration

Software may need to be integrated with the libraries, databases and other program(s). This stage of SDLC is involved in the integration of software with outer world entities.

Implementation

This means installing the software on user machines. At times, software needs post-installation configurations at user end. Software is tested for portability and adaptability and integration related issues are solved during implementation.

Operation and Maintenance

This phase confirms the software operation in terms of more efficiency and less errors. If required, the users are trained on, or aided with the documentation on how to operate the software and how to keep the software operational. The software is maintained timely by updating the code according to the changes taking place in user end environment or technology. This phase may face challenges from hidden bugs and real-world unidentified problems.

Disposition

As time elapses, the software may decline on the performance front. It may go completely obsolete or may need intense up gradation. Hence a pressing need to eliminate a major portion of the system arises. This phase includes archiving data and required software components, closing down the system, planning disposition activity and terminating system at appropriate end-of-system time.

Software Project
A Software Project is the complete procedure of software development from requirement gathering to testing and maintenance, carried out according to the execution methodologies, in a specified period of time to achieve intended software product.

Software Management Activities

Software project management comprises of a number of activities, which contains planning of project, deciding scope of software product, estimation of cost in various terms, scheduling of tasks and events, and resource management. Project management activities may include:

- **Project Planning**
- **Scope Management**
- **Project Estimation**

Project Planning

Software project planning is task, which is performed before the production of software actually starts. It is there for the software production but involves no concrete activity that has any direction connection with software production; rather it is a set of multiple processes, which facilitates software production. Project planning may include the following:

Scope Management

It defines the scope of project; this includes all the activities, process need to be done in order to make a deliverable software product. Scope management is essential because it creates boundaries of the project by clearly defining what would be done in the project and what would not be done. This makes project to contain limited and quantifiable tasks, which can easily be documented and in turn avoids cost and time overrun.

During Project Scope management, it is necessary to -

- Define the scope
- Decide its verification and control
- Divide the project into various smaller parts for ease of management.
- Verify the scope
- Control the scope by incorporating changes to the scope

Project Estimation

For an effective management accurate estimation of various measures is a must. With correct estimation managers can manage and control the project more efficiently and effectively.

Project estimation may involve the following:

- **Software size estimation**

  Software size may be estimated either in terms of KLOC (Kilo Line of Code) or by calculating number of function points in the software. Lines of code depend upon coding practices and Function points vary according to the user or software requirement.

- **Effort estimation**

  The managers estimate efforts in terms of personnel requirement and man-hour required to produce the software. For effort estimation software size should be known. This can either be derived by managers' experience, organization's historical data or software size can be converted into efforts by using some standard formulae.

- **Time estimation**

  Once size and efforts are estimated, the time required to produce the software can be estimated. Efforts required is segregated into sub categories as per the requirement specifications and interdependency of various components of software. Software tasks are divided into smaller tasks, activities or events by Work Breakthrough Structure (WBS). The tasks are scheduled on day-to-day basis or in calendar months.

  The sum of time required to complete all tasks in hours or days is the total time invested to complete the project.

- **Cost estimation**

This might be considered as the most difficult of all because it depends on more elements than any of the previous ones. For estimating project cost, it is required to consider -

- o Size of software
- o Software quality
- o Hardware
- o Additional software or tools, licenses etc.
- o Skilled personnel with task-specific skills
- o Travel involved
- o Communication
- o Training and support

Project Estimation Techniques

We discussed various parameters involving project estimation such as size, effort, time and cost.

Project manager can estimate the listed factors using two broadly recognized techniques –

Decomposition Technique

This technique assumes the software as a product of various compositions.

There are two main models -

- **Line of Code** Estimation is done on behalf of number of line of codes in the software product.
- **Function Points** Estimation is done on behalf of number of function points in the software product.

Empirical Estimation Technique

This technique uses empirically derived formulae to make estimation.These formulae are based on LOC or FPs.

- **Putnam Model**

  This model is made by Lawrence H. Putnam, which is based on Norden's frequency distribution (Rayleigh curve). Putnam model maps time and efforts required with software size.

- **COCOMO**

  COCOMO stands for COnstructive COst MOdel, developed by Barry W. Boehm. It divides the software product into three categories of software: organic, semi-detached and embedded.

Project Scheduling

Project Scheduling in a project refers to roadmap of all activities to be done with specified order and within time slot allotted to each activity. Project managers tend to define various tasks, and project milestones and arrange them keeping various factors in mind. They look for tasks lie in critical path in the schedule, which are necessary to complete in specific manner (because of task interdependency) and strictly within the time allocated. Arrangement of tasks which lies out of critical path are less likely to impact over all schedule of the project.

For scheduling a project, it is necessary to -

- Break down the project tasks into smaller, manageable form
- Find out various tasks and correlate them
- Estimate time frame required for each task
- Divide time into work-units
- Assign adequate number of work-units for each task

- Calculate total time required for the project from start to finish

THE EVOLUTION OF SOFTWARE ENGINEERING TECHNIQUES:

Software is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called **software product.**

Engineering on the other hand, is all about developing products, using well-defined, scientific principles and methods.

Software engineering **is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.**

Definitions

IEEE defines software engineering as:

(1) The application of a systematic,disciplined,quantifiable approach to the development,operation and maintenance of software; that is, the application of engineering to software.

(2) The study of approaches as in the above statement.

Fritz Bauer, a German computer scientist, defines software engineering as:

Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and work efficiently on real machines.

TASK: 1 IDENTIFYING THE REQUIREMENTS FROM PROBLEM STATEMENTS

1. INTRODUCTION

Requirements identification is the first step of any software development project. Until the requirements of a client have been clearly identified, and verified, no other task (design, coding, testing) could begin. Usually business analysts having domain knowledge on the subject matter discuss with clients and decide what features are to be implemented.

In this experiment we will learn how to identify functional and non-functional requirements from a given problem statement. Functional and non-functional requirements are the primary components of a Software Requirements Specification.

2. REQUIREMENTS

Sommerville defines "requirement" [1] as a specification of what should be implemented. Requirements specify how the target system should behave. It specifies what to do, but not how to do. Requirements engineering refers to the process of understanding what a customer expects from the system to be developed, and to document them in a standard and easily readable and understandable format. This documentation will serve as reference for the subsequent design, implementation and verification of the system.

It is necessary and important that before we start planning, design and implementation of the software system for our client, we are clear about it's requirements. If we don't have a clear vision of what is to be developed and what all features are expected, there would be serious problems, and customer dissatisfaction as well.

Characteristics of Requirements

Requirements gathered for any new system to be developed should exhibit the following three properties:

- **Unambiguity:** There should not be any ambiguity what a system to be developed should do. For example, consider you are developing a web application for your client. The client requires that enough number of people should be able to access the application simultaneously. What's the "enough number of people"? That could mean 10 to you, but, perhaps, 100 to the client. There's an ambiguity.
- **Consistency:** To illustrate this, consider the automation of a nuclear plant. Suppose one of the clients say that it the radiation level inside the plant exceeds R1, all reactors should be shut down. However, another person from the client side suggests that the threshold radiation level should be R2. Thus, there is an inconsistency between the two end users regarding what they consider as threshold level of radiation.
- **Completeness:** A particular requirement for a system should specify what the system should do and also what it should not. For example, consider a software to be developed for ATM. If a customer enters an amount greater than the maximum permissible withdrawal amount, the ATM should display an error message, and it should not dispense any cash.

Categorization of Requirements

Based on the target audience or subject matter, requirements can be classified into different types, as stated below:

- **User requirements:** They are written in natural language so that both customers can verify their requirements have been correctly identified
- **System requirements:** They are written involving technical terms and/or specifications, and are meant for the development or testing teams

Requirements can be classified into two groups based on what they describe:

- **Functional requirements (FRs):** These describe the functionality of a system -- how a system should react to a particular set of inputs and what should be the corresponding output.
- **Non-functional requirements (NFRs):** They are not directly related what functionalities are expected from the system. However, NFRs could typically define how the system should behave under certain situations. For example, a NFR could say that the system should work with 128MB RAM. Under such condition, a NFR could be more critical than a FR.

Non-functional requirements could be further classified into different types like:

- **Product requirements:** For example, a specification that the web application should use only plain HTML, and no frames
- **Performance requirements:** For example, the system should remain available 24x7
- **Organizational requirements:** The development process should comply to SEI CMM level 4

Functional Requirements

Identifying Functional Requirements

Given a problem statement, the functional requirements could be identified by focusing on the following points:

- Identify the high level functional requirements simply from the conceptual understanding of the problem. For example, a Library Management System, apart from anything else, should be able to issue and return books.

- Identify the cases where an end user gets some meaningful work done by using the system. For example, in a digital library a user might use the "Search Book" functionality to obtain information about the books of his interest.
- If we consider the system as a black box, there would be some inputs to it, and some output in return. This black box defines the functionalities of the system. For example, to search for a book, user gives title of the book as input and get the book details and location as the output.
- Any high level requirement identified could have different sub-requirements. For example, "Issue Book" module could behave differently for different class of users, or for a particular user who has issued the book thrice consecutively.

Preparing Software Requirements Specifications

Once all possible FRs and non-FRs have been identified, which are complete, consistent, and non-ambiguous, the Software Requirements Specification (SRS) is to be prepared. IEEE provides a template [iv], also available here, which could be used for this purpose. The SRS is prepared by the service provider, and verified by its client. This document serves as a legal agreement between the client and the service provider. Once the concerned system has been developed and deployed, and a proposed feature was not found to be present in the system, the client can point this out from the SRS. Also, if after delivery, the client says a new feature is required, which was not mentioned in the SRS, the service provider can again point to the SRS. The scope of the current experiment, however, doesn't cover writing a SRS.

3. SIMULATION:

We show here how to extract functional requirements when a problem statement is given. The case under study is a online voting system.

Internet has led to discussion of e-democracy and online voting. Many peoples think that the internet could replace representative democracy, enabling everyone to vote on everything and anything by online voting .Online voting could reduce cost and make voting more convenient. This type of voting can be done for e-democracy, or it may be used for finalizing a solution, if many alternatives are present. Online voting make's use of authentication, hence it needs security, and the system must be able to address obtaining, marking, delivering and counting ballots via computer. Advantage of online voting is it could increase voter turnout because of convenience, and it helps to reduce fraud voting.

5. CASE STUDY

#1 : A Library Information System for SE VLabs Institute

The SE VLabs Institute has been recently setup to provide state-of-the-art research facilities in the field of Software Engineering. Apart from research scholars (students) and professors, it also includes quite a large number of employees who work on different projects undertaken by the institution.

As the size and capacity of the institute is increasing with the time, it has been proposed to develop a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member

is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (eg., passwords) is stored in plain text.

Identification of functional requirements

The above problem statement gives a brief description of the proposed system. From the above, even without doing any deep analysis, we might easily identify some of the basic functionality of the system:

- **New user registration:** Any member of the institute who wishes to avail the facilities of the library has to register himself with the Library Information System. On successful registration, a user ID and password would be provided to the member. He has to use this credentials for any future transaction in LIS.
- **Search book:** Any member of LIS can avail this facility to check whether any particular book is present in the institute's library. A book could be searched by its:
    - Title
    - Authors name
    - Publisher's name

User login: A registered user of LIS can login to the system by providing his employee ID and password as set by him while registering. After successful login, "Home" page for the user is shown from where he can access the different functionalities of LIS: search book, issue book, return book, reissue book. Any employee ID not registered with LIS cannot access the "Home" page -- a login failure message would be shown to him, and the login dialog would appear again. This same thing happens when any registered user types in his password wrong. However, if incorrect password has been provided for three time consecutively, the security question for the user (specified while registering) with an input box to answer it are also shown. If the user can answer the security question correctly, a new password would be sent to his email address. In case the user fails to answer the security question correctly, his LIS account would be blocked. He needs to contact with the administrator to make it active again.

Issue book: Any member of LIS can issue a book against his account provided that:
- The book is available in the library i.e. could be found by searching for it in LIS
- No other member has currently issued the book
- Current user has not issued the maximum number of books that can

If the above conditions are met, the book is issued to the member. Note that this FR would remain **incomplete** if the "maximum number of books that can be issued to a member" is not defined. We assume that this number has been set to four for students and research scholars, and to ten for professors. Once a book has been successfully issued, the user account is updated to reflect the same.

Return book: A book is issued for a finite time, which we assume to be a period of 20 days. That is, a book once issued should be returned within the next 20 days by the corresponding member of LIS. After successful return of a book, the user account is updated to reflect the same.

Reissue book: Any member who has issued a book might find that his requirement is not over by 20 days. In that case, he might choose to reissue the book, and get the permission to keep it for another 20 days. However, a member can reissue any book at most twice, after which he has to return it. Once a book has been successfully reissued, the user account is updated to reflect the information.

In a similar way we can list other functionality offered by the system as well. However, certain features might not be evident directly from the problem system, but which, nevertheless, are required. One such functionality is "User Verification". The LIS should be able to judge between a registered and non-registered member. Most of the functionality would be available to a registered member. The "New User Registration" would, however, be available to non-members. Moreover, an already registered user shouldn't be allowed to register himself once again.

Having identified the (major) functional requirements, we assign an identifier to each of them [v] for future reference and verification. Following table shows the list:

Table 01: Identifier and priority for software requirements		
#	Requirement	Priority
R1	New user registration	High
R2	User Login	High
R3	Search book	High
R4	Issue book	High
R5	Return book	High
R6	Reissue book	Low

Identification of non-functional requirements

Having talked about functional requirements, let's try to identify a few non-functional requirements.

- **Performance Requirements:**
  - This system should remain accessible 24x7
  - At least 50 users should be able to access the system altogether at any given time

Security Requirements:
  - This system should be accessible only within the institute LAN
  - The database of LIS should not store any password in plain text -- a hashed value has to be stored

Software Quality Attributes

Database Requirements

Design Constraints:
  - The LIS has to be developed as a web application, which should work with Firefox 5, Internet Explorer 8, Google Chrome 12, Opera 10
  - The system should be developed using HTML 5

Once all the functional and non-functional requirements have been identified, they are documented formally in SRS, which then serves as a legal agreement.

1. OPERATORS

a. Read a list of numbers and write a program to check whether a particular element is present or not using membership operators.

In:
x = ["apple", "banana"]
print("banana" in x)
output:True

notin:
x = ["apple", "banana"]
print("pineapple" not in x)
output:True

b.Read your name and age and write a program to display the year in which you will turn 100 years old.

name=str(input("enter your name:"))

a=int(input("enter your age:"))

print(name)

print(a)

year=int(input("enter the present year:"))

t=100-a

print(t)

if t>=100:

   print("you have already crossed the age of 100")

else:

   ans=year+t

   print("in the ",ans,"year you will turn 100")

output:

```
enter your name:bujji
enter your age:20
bujji
20
enter the present year:37
80
in the  117 year you will turn 100
```

c.Read radius and height of a cone and write a program to find the volume of a cone.

Python Program to find Volume and Surface Area of a Cone

```
import math

radius = float(input('Please Enter the Radius of a Cone: '))
height = float(input('Please Enter the Height of a Cone: '))

# Calculate Length of a Slide (Slant)
l = math.sqrt(radius * radius + height * height)

# Calculate the Surface Area
SA = math.pi * radius * (radius + l)

# Calculate the Volume
Volume = (1.0/3) * math.pi * radius * radius * height
print("\n Length of a Side (Slant)of a Cone = %.2f" %l)
print(" The Surface Area of a Cone = %.2f " %SA)
print(" The Volume of a Cone = %.2f" %Volume);
```

output:

```
Please Enter the Radius of a Cone: 5
Please Enter the Height of a Cone: 12

 Length of a Side (Slant)of a Cone = 13.00
 The Surface Area of a Cone = 282.74
 The Volume of a Cone = 314.16
```

d.Write a program to compute distance between two points taking input from the user (Hint: use Pythagorean theorem)

x1=int(input("enter x1 : "))

x2=int(input("enter x2 : "))

y1=int(input("enter y1 : "))

y2=int(input("enter y2 : "))

result= ((((x2 - x1)**2) + ((y2-y1)**2))**0.5)

print("distance between",(x1,x2),"and",(y1,y2),"is : ",result)

output:
```
enter x1 : 4
enter x2 : 6
enter y1 : 0
enter y2 : 6
distance between (4, 6) and (0, 6) is :
6.324555320336759
```

e.Arithmetic operators

```python
x = 15
y = 4

# Output: x + y = 19
print('x + y =',x+y)
```

```python
# Output: x - y = 11
print('x - y =',x-y)

# Output: x * y = 60
print('x * y =',x*y)

# Output: x / y = 3.75
print('x / y =',x/y)

# Output: x // y = 3
print('x // y =',x//y)

# Output: x ** y = 50625
print('x ** y =',x**y)
```

output:
```
x + y = 19
x - y = 11
x * y = 60
x / y = 3.75
x // y = 3
x ** y = 50625
```

f.comparision operators

```python
x = 10
y = 12

# Output: x > y is False
print('x > y is',x>y)

# Output: x < y is True
print('x < y is',x<y)

# Output: x == y is False
print('x == y is',x==y)

# Output: x != y is True
```

```python
print('x != y is',x!=y)

# Output: x >= y is False
print('x >= y is',x>=y)

# Output: x <= y is True
print('x <= y is',x<=y)
```

output:
```
x > y is False
x < y is True
x == y is False
x != y is True
x >= y is False
x <= y is True
```

2. CONTROL STRUCTURES

1.Read *your* email id and write a program to display the no of vowels,consonants,digits and white spaces in it using if elif else *statement* in python

```python
def count(str):

    # Declare the variable vowels,
    # consonant, digit and special
    # characters
    vowels = 0
    consonant = 0
    specialChar = 0
    digit = 0

    # str.length() function to count
    # number of character in given string.
    for i in range(0, len(str)):

        ch = str[i]

        if ( (ch >= 'a' and ch <= 'z') or
            (ch >= 'A' and ch <= 'Z') ):

            # To handle upper case letters
            ch = ch.lower()

            if (ch == 'a' or ch == 'e' or ch == 'i'
                    or ch == 'o' or ch == 'u'):
                vowels += 1
            else:
```

```
                consonant += 1

        elif (ch >= '0' and ch <= '9'):
            digit += 1
        else:
            specialChar += 1

    print("Vowels:", vowels)
    print("Consonant:", consonant)
    print("Digit:", digit)
    print("Special Character:", specialChar)


# Driver function.
str = "asfsgfgfhgfdghh123@gmail.com"
count(str)
output:
Vowels: 4
Consonant: 19
Digit: 3
Special Character: 2
```

2. Write a program to create and display a dictionary by storing the antonyms of words. Find the antonym of a particular word given by the user from the dictionary using for loop.

```
print('enter word from following words ')

no ={ 'right': 'left', 'correct':'wrong', 'yes':'no' }

for i in no.keys():
```

```python
    print(i)

y = input()

if y in no :

    print('antonym is ', no[y])

else:

    print('not in the list given : ')
```

output:

enter word from following words
right
correct
yes
right
antonym is left

3. Write a Program to find the sum of a Series 1/1! + 2/2! + 3/3! + 4/4! +.......+ n/n!. (Input :n = 5, Output : 2.70833)

```python
def sumOfSeries(num):

    # Computing MAX
    res = 0
    fact = 1

    for i in range(1, num+1):
        fact *= i
        res = res + (i/ fact)

    return res
```

```
n = 5
print("Sum: ", sumOfSeries(n))
```

output:
Sum: 2.708333333333333

4. In number theory, an abundant number or excessive number is a number for which the sum of its proper divisors is greater than the number itself. Write a program to find out, if the given number is abundant. (Input: 12, Sum of divisors of 12 = 1 + 2 + 3 + 4 + 6 = 16, sum of divisors 16 > original number 12)

```
print('Enter the number:')

n=int(input( ))

sum=1 # 1 can divide any number

for i in range(2,n):
  if(n%i==0):   #if number is divisible by i add the number
   sum=sum+i

if(sum>n):
  print(n, 'is Abundant Number')

else:
  print(n, 'is not Abundant Number')
```

output:
Enter the number:
15
15 is not Abundant Number

1. Read a list of numbers and print the numbers divisible by x but not by y (Assume x = 4 and y = 5).

```
def findNoIsDivisibleOrNot(n, l =[]):

    # Checking if a number is divided
    # by every element or not
    for i in range(0, len(li)):
        if li[i]% n != 0:
            return 0
    return 1

# Driver code
li = [14, 12, 4, 18]
n = 2
if findNoIsDivisibleOrNot(n, li) == 1:
    print ("Yes")
else:
    print ("No")
```

output:
Yes

2. Read a list of numbers and print the sum of odd integers and even integers from the list.(Ex: [23, 10, 15, 14, 63], odd numbers sum = 101, even numbers sum = 24)

```
# Python Program to find Sum of Even and Odd Numbers in a List

NumList = []
Even_Sum = 0
Odd_Sum = 0

Number = int(input("Please enter the Total Number of List Elements: "))
for i in range(1, Number + 1):
    value = int(input("Please enter the Value of %d Element : " %i))
    NumList.append(value)

for j in range(Number):
    if(NumList[j] % 2 == 0):
        Even_Sum = Even_Sum + NumList[j]
    else:
        Odd_Sum = Odd_Sum + NumList[j]

print("\nThe Sum of Even Numbers in this List =  ", Even_Sum)
print("The Sum of Odd Numbers in this List =  ", Odd_Sum)
```

output:
Please enter the Total Number of List Elements: 5
Please enter the Value of 1 Element : 23
Please enter the Value of 2 Element : 15
Please enter the Value of 3 Element : 10
Please enter the Value of 4 Element : 14

Please enter the Value of 5 Element : 63
The Sum of Even Numbers in this List =   24
The Sum of Odd Numbers in this List =   101

3. Read a list of numbers and print numbers present in odd index position. (Ex: [10, 25, 30, 47, 56, 84, 96], The numbers in odd index position: 25 47 84).

test_list =[10, 25, 30, 47, 56, 84, 96]

```
# printing original list
print("The original list : " + str(test_list))

# using naive method
# Separating odd and even index elements
odd_i = []
for i in range(0, len(test_list)):
    if i % 2:
        odd_i.append(test_list[i])

res = odd_i

# print result
print("odd  index list: " + str(res))
```

output:
The original list : [10, 25, 30, 47, 56, 84, 96]
odd index list: [25, 47, 84]

4. Read a list of numbers and remove the duplicate numbers from it. (Ex: Enter a list with duplicate elements: 10 20 40 10 50 30 20 10 80, The unique list is: [10, 20, 30, 40, 50, 80])
Program:
```
def Remove(duplicate):
    final_list = []
    for num in duplicate:
        if num not in final_list:
            final_list.append(num)
    return final_list

# Driver Code
duplicate = [10,20, 40 ,10, 50, 30, 20, 10, 80]
print(Remove(duplicate))
```

output:
[10, 20, 40, 50, 30, 80]

1.Given a list of tuples. Write a program to find tuples which have all elements divisible by K from a list of tuples.
test_list = [(6, 24, 12), (60, 12, 6), (12, 18, 21)], K = 6, Output : [(6, 24, 12), (60, 12, 6)]

Program:

```
test_list = [(6, 24, 12), (60, 12, 6), (12, 18, 21)]
# printing original list
print("The original list is : " + str(test_list))

# initializing K
K = 6

# all() used to filter elements
res = [sub for sub in test_list if all(ele % K == 0 for ele in sub)]

# printing result
print("K Multiple elements tuples : " + str(res))
```

output:
The original list is : [(6, 24, 12), (60, 12, 6), (12, 18, 21)]
K Multiple elements tuples : [(6, 24, 12), (60, 12, 6)]

2. Given a list of tuples. Write a program to filter all uppercase characters tuples from given list of tuples. (Input: test_list = [("GFG", "IS", "BEST"), ("GFg", "AVERAGE"), ("GfG",), ("Gfg", "CS")], Output : [(„GFG", „IS", „BEST")]).

Program:
```
test_list = [("GFG", "IS", "BEST"), ("GFg", "AVERAGE"), ("GFG", ), ("Gfg", "CS")]

# printing original list
print("The original list is : " + str(test_list))

res_list = []
for sub in test_list:
    res = True
    for ele in sub:

        # checking for uppercase
        if not ele.isupper():
            res = False
            break
    if res:
        res_list.append(sub)

# printing results
print("Filtered Tuples : " + str(res_list))
```

output:
The original list is : [('GFG', 'IS', 'BEST'), ('GFg', 'AVERAGE'), ('GFG',), ('Gfg', 'CS')]
Filtered Tuples : [('GFG', 'IS', 'BEST'), ('GFG',)]

3. Given a tuple and a list as input, write a program to count the occurrences of all items of the list in the tuple.
(Input : tuple = ('a', 'a', 'c', 'b', 'd'), list = ['a', 'b'], Output : 3)

Program:

```
from collections import Counter

def countOccurrence(tup, lst):
    counts = Counter(tup)
    return sum(counts[i] for i in lst)

# Driver Code
tup = ('a', 'a', 'c', 'b', 'd')
lst = ['a', 'b']
print(countOccurrence(tup, lst))
```

output:
3

1. Write a program to generate and print a dictionary that contains a number (between 1 and n) in the form (x, x*x).

 Program:

```
n=int(input("Input a number "))
d = dict()

for x in range(1,n+1):
    d[x]=x*x

print(d)
```

output:
Input a number 5
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

2. Write a program to perform union, intersection and difference using Set A and Set B.
Program:

```
A = {0, 2, 4, 6, 8};
B = {1, 2, 3, 4, 5};

# union
print("Union :", A | B)

# intersection
print("Intersection :", A & B)

# difference
print("Difference :", A - B)

# symmetric difference
print("Symmetric difference :", A ^ B)
```

output:

Union : {0, 1, 2, 3, 4, 5, 6, 8}
Intersection : {2, 4}
Difference : {0, 8, 6}
Symmetric difference : {0, 1, 3, 5, 6, 8}

3. Write a program to count number of vowels using sets in given string (Input : "Hello World", Output: No. of vowels : 3)

Program:

```
def vowel_count(str):

    # Initializing count variable to 0
    count = 0

    # Creating a set of vowels
    vowel = set("aeiouAEIOU")

    # Loop to traverse the alphabet
    # in the given string
    for alphabet in str:

        # If alphabet is present
        # in set vowel
        if alphabet in vowel:
            count = count + 1

    print("No. of vowels :", count)

# Driver code
str = "Hello World"

# Function Call
vowel_count(str)
```

output:
No. of vowels : 3

4.Write a program to form concatenated string by taking uncommon characters from two strings using set concept (Input : S1 = "aacdb", S2 = "gafd", Output : "cbgf").

Program:
```
def uncommonConcat(str1, str2):

    # convert both strings into set
    set1 = set(str1)
    set2 = set(str2)

    # take intersection of two sets to get list of
    # common characters
    common = list(set1 & set2)

    # separate out characters in each string
    # which are not common in both strings
    result = [ch for ch in str1 if ch not in common] + [ch for ch in str2 if ch not in common]
```

```python
        # join each character without space to get
        # final string
        print( ''.join(result) )

# Driver program
if __name__ == "__main__":
    str1 = 'aacdb'
    str2 = 'gafd'
    uncommonConcat(str1,str2)
```

output:
cbgf

a.Create a empty dictionary with dict() method

Creating an empty Dictionary

Dict = {}

print("Empty Dictionary: ")

print(Dict)

output:

Empty Dictionary:

{}

b. Add elements one at a time

```
# Creating an empty Dictionary
Dict = {}
print("Empty Dictionary: ")
print(Dict)

# Adding elements one at a time
Dict[0] = 'Geeks'
Dict[2] = 'For'
Dict[3] = 1
print("\nDictionary after adding 3 elements: ")
print(Dict)
```

output:

Empty Dictionary:

{}

Dictionary after adding 3 elements: {0: 'Geeks', 2: 'For', 3: 1}

3. Update existing key''s value
# Creating an empty Dictionary
Dict = {}
print("Empty Dictionary: ")
print(Dict)

```
# Adding elements one at a time
Dict[0] = 'Geeks'
Dict[2] = 'For'
Dict[3] = 1
print("\nDictionary after adding 3 elements: ")
print(Dict)
# Updating existing Key's Value
Dict[2] = 'Welcome'
print("\nUpdated key value: ")
print(Dict)
```

output:

Empty Dictionary: {}

Dictionary after adding 3 elements:{0: 'Geeks', 2: 'For', 3: 1}

Updated key value: {0: 'Geeks', 2: 'Welcome', 3: 1}

4.Access an element using a key and also get() method
```
dic = {"A": 1, "B": 2}
print(dic.get("A"))
print(dic.get("C"))
print(dic.get("C", "Not Found ! "))
```

output:
1
None
Not Found !

5. Deleting a key value using del() method

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict)
```

output:
{'brand': 'Ford', 'year': 1964}

6.pop() method

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)
```

output:

{'brand': 'Ford', 'year': 1964}

7. popitem() method

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.popitem()
print(thisdict)
```

output:
{'brand': 'Ford', 'model': 'Mustang'}

8.clear() method
```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.clear()
print(thisdict)
```

output:
{}

9. Given a dictionary, write a program to find the sum of all items in the dictionary.

```
dic={ 'x':455, 'y':223, 'z':300, 'p':908 }

print("Dictionary: ", dic)

#using sum() and values()
print("sum: ",sum(dic.values()))
```

output:
Dictionary: {'x': 455, 'y': 223, 'z': 300, 'p': 908}
sum: 1886

10. Write a program to merge two dictionaries using update() method

```
def Merge(dict1, dict2):
    res = {**dict1, **dict2}
    return res

# Driver code
dict1 = {'a': 10, 'b': 8}
dict2 = {'d': 6, 'c': 4}
dict3 = Merge(dict1, dict2)
print(dict3)
```

output:
{'a': 10, 'b': 8, 'd': 6, 'c': 4}

1. Given a string, write a program to check if the string is symmetrical and palindrome or not. A string is said to be symmetrical if both the halves of the string are the same and a string is said to be a palindrome string if one half of the string is the reverse of the other half or if a string appears same when read forward or backward.

Program:

```
string = 'amaama'
half = int(len(string) / 2)

if len(string) % 2 == 0:  # even
    first_str = string[:half]
    second_str = string[half:]
else:  # odd
    first_str = string[:half]
    second_str = string[half+1:]

# symmetric
if first_str == second_str:
    print(string, 'string is symmertical')
else:
    print(string, 'string is not symmertical')

# palindrome
if first_str == second_str[::-1]:  # ''.join(reversed(second_str)) [slower]
    print(string, 'string is palindrome')
else:
    print(string, 'string is not palindrome')
```

output:

amaama string is symmertical

amaama string is palindrome

2. Write a program to read a string and count the number of vowel letters and print all letters except 'e' and 's'.
str=input("Please enter a string as you wish: ");

vowels=0

for i in str:

   if(i == 'a'or i == 'i'or i == 'o'or i == 'u' or

     i == 'A'or i == 'E'or i == 'I'or i == 'O'or i == 'U'):

        vowels=vowels+1;#vowel counter is incremented by 1

print("The number of vowels:",vowels);

output:

Please enter a string as you wish: fruites

The number of vowels: 2

3. Write a program to read a line of text and remove the initial word from given text. (Hint: Use split() method, Input : India is my country. Output : is my country)

test_str = "India is my country"

# printing original string
print("The original string is : " + test_str)

# Using split()
# Removing Initial word from string
res = test_str.split(' ', 1)[1]

# printing result
print("The string after omitting first word is : " + str(res))

output:
The original string is : India is my country
The string after omitting first word is : is my country

4. Write a program to read a string and count how many times each letter appears. (Histogram).

```
def letterFrequency(fileName, letter):
    # open file in read mode
    file = open(fileName, "r")

    # store content of the file in a variable
    text = file.read()

    # declare count variable
    count = 0

    # iterate through each character
    for char in text:

        # compare each character with
        # the given letter
        if char == letter:
            count += 1

    # return letter count
    return count
```
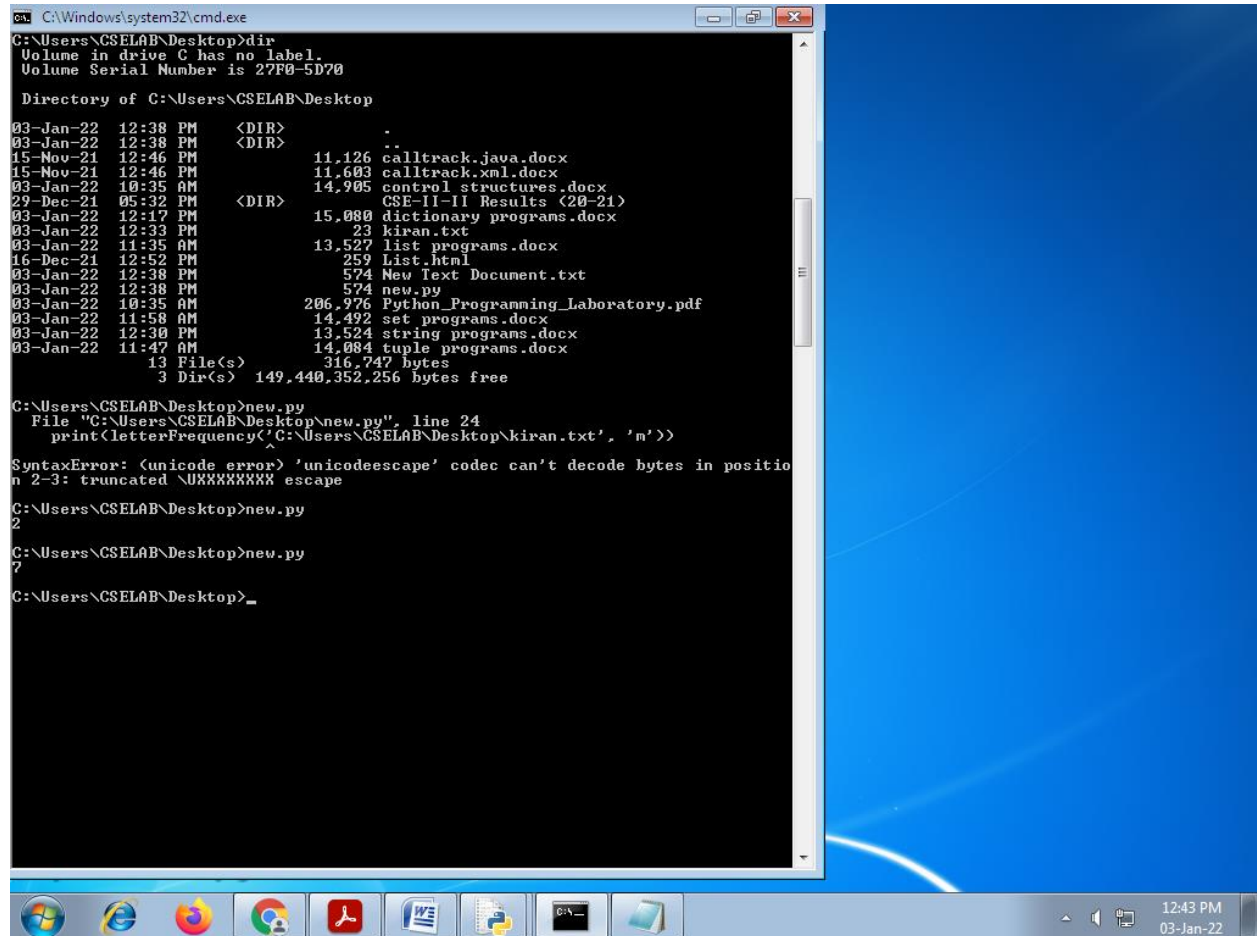
# call function and display the letter count
print(letterFrequency('kiran.txt', 'a'))

Text file: malayalam is a language
Output:

1.A generator is a function that produces a sequence of results instead of a single value. Write a generator function for Fibonacci numbers up to n.

Program:

```
def fib(num):
    a = 0
    b = 1
    for i in range(num):
        yield a
        a, b = b, a + b # Adds values together then swaps them
for x in fib(100):
    print(x)
```

output:
```
0
1
1
2
3
5
8
13
21
34
```

2. Write a function merge_dict(dict1, dict2) to merge two Python dictionaries.

Program:
```
def Merge(dict1, dict2):
    return(dict2.update(dict1))

# Driver code
dict1 = {'a': 10, 'b': 8}
dict2 = {'d': 6, 'c': 4}

# This return None
print(Merge(dict1, dict2))

# changes made in dict2
print(dict2)
```

output:

```
None
{'d': 6, 'c': 4, 'a': 10, 'b': 8}
```

3.Write a fact() function to compute the factorial of a given positive number.
Program:
```
num = int(input("Enter a number: "))
factorial = 1
if num < 0:
    print(" Factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1,num + 1):
        factorial = factorial*i
    print("The factorial of",num,"is",factorial)
```

output:

```
Enter a number: 5
The factorial of 5 is 120
```

4. Given a list of n elements, write a linear_search() function to search a given element x in a list.

Progam:

```
def linearsearch(arr, x):
    for i in range(len(arr)):
        if arr[i] == x:
            return i
    return -1
arr = ['t','u','t','o','r','i','a','l']
x = 'a'
print("element found at index "+str(linearsearch(arr,x)))
```

output:

```
element found at index 6
```

1. Write a program to demonstrate the working of built-in statistical functions mean(), mode(), median() by importing statistics library.

Mean:

```
n_num = [1, 2, 3, 4, 5]
n = len(n_num)

get_sum = sum(n_num)
mean = get_sum / n

print("Mean is: " + str(mean))
```

output:
```
Mean  is: 3.0
```

Median:
```
n_num = [1, 2, 3, 4, 5]
n = len(n_num)
n_num.sort()

if n % 2 == 0:
    median1 = n_num[n//2]
    median2 = n_num[n//2 - 1]
    median = (median1 + median2)/2
else:
    median = n_num[n//2]
print("Median is: " + str(median))
```

output:
```
Median is: 3
```

Mode:

```
import statistics
set1 =[1, 2, 3, 3, 4, 4, 4, 5, 5, 6]
print("Mode of given data set is % s" % (statistics.mode(set1)))
```

output:
```
Mode of given data set is 4
```

2.Write a program to demonstrate the working of built-in trignometric functions sin(), cos(), tan(), hypot(), degrees(), radians() by importing math module.

Program:

```
import math

print(math.sin(math.pi/3)) #pi/3 radians is converted to 60 degrees
print(math.tan(math.pi/3))
print(math.cos(math.pi/6))
```

```
print(math.hypot(2,2))
 math.radians(30)
math.degrees(math.pi/6)
```

output:

```
0.8660254037844386
1.7320508075688767
0.8660254037844387
2.8284271247461903
0.5235987755982988
29.999999999999996
```

3. Write a program to demonstrate the working of built-in Logarithmic and Power functions exp(), log(), log2(), log10(), pow() by importing math module.

```python
import math

# returning the exp of 4
print ("The e**4 value is : ", end="")
print (math.exp(4))

# returning the log of 2,3
print ("The value of log 2 with base 3 is : ", end="")
print (math.log(2,3))

# returning the log2 of 16
print ("The value of log2 of 16 is : ", end="")
print (math.log2(16))

# returning the log10 of 10000
print ("The value of log10 of 10000 is : ", end="")
print (math.log10(10000))

print ("The value of 3 to the power 2 is : ", end="")
print (math.pow(3,2))
```

output:

```
The e**4 value is : 54.598150033144236
The value of log 2 with base 3 is : 0.6309297535714574
The value of log2 of 16 is : 4.0
The value of log10 of 10000 is : 4.0
The value of 3 to the power 2 is : 9.0
```

4. Write a program to demonstrate the working of built-in numeric functions ceil(), floor(), fabs(), factorial(), gcd() by importing math module.

Program:

```python
import math
```

```python
my_int = 4.5467
print (math.ceil(my_int))



my_int = 4.5467
print (math.floor(my_int))

my_int = 4.5467
print (math.fabs(my_int))

my_val_1 = 8
my_val_2 = -6.9
print(math.copysign(my_val_1, my_val_2))

a = 5

# returning the factorial of 5
print("The factorial of 5 is : ", end="")
print(math.factorial(a))

a = 15
b = 5

# returning the gcd of 15 and 5
print ("The gcd of 5 and 15 is : ", end="")
print (math.gcd(b, a))
```

output:

```
5
4
4.5467
-8.0
The factorial of 5 is : 120
The gcd of 5 and 15 is : 5
```

1. Write a program to create a BankAccount class. Your class should support the following methods for
i) Deposit
ii) Withdraw
iii) GetBalanace

Program:

```
class Bank:
    def __init__(self):
        self.balance = 0
        print ("The account is created")

    def deposit(self):
        amount = float(input("Enter the amount to be deposit: "))
        self.balance = self.balance + amount
        print ("The deposit is successful and the balance in the account is %f" % self.balance)

    def withdraw(self):
        amount = float(input("Enter the amount to withdraw: "))
        if (self.balance >= amount):
            self.balance = self.balance - amount
            print ("The withdraw is successful and the balance is %f" % self.balance)
        else:
            print ('Insuficient Balance')

    def getbalance(self):
        print ("Balance in the account is %f" % self.balance)


acc = Bank()
acc.deposit()
acc.withdraw()
acc.getbalance()
```

output:

The account is created
Enter the amount to be deposit: 20000
The deposit is successful and the balance in the account is 20000.000000
Enter the amount to withdraw: 15000
The withdraw is successful and the balance is 5000.000000
Balance in the account is 5000.000000

2. Create a SavingsAccount class that behaves just like a BankAccount, but also has an interest rate and a method that increases the balance by the appropriate amount of interest (Hint:use Inheritance).

3. Write a program to create an employee class and store the employee name, id, age, and salary using the constructor. Display the employee details by invoking employee_info() method and also using dictionary (__dict__).

Program:
```
class Employee:
    __id=0
    __name=""
    __gender=""
    __city=""
    __salary=0

        # function to set data
    def setData(self,id,name,gender,city,salary):
        self.__id= id
        self.__name = name
        self.__gender = gender
        self.__city = city
        self.__salary = salary

        # function to get/print data
    def showData(self):
        print("Id\t\t:",self.__id)
        print("Name\t:", self.__name)
        print("Gender\t:", self.__gender)
        print("City\t:", self.__city)
        print("Salary\t:", self.__salary)

# main function definition
def main():
    #Employee Object
    emp=Employee()
    emp.setData(1,'pankaj','male','delhi',55000)
    emp.showData()

if __name__=="__main__":
    main()
```

output:

```
Id              : 1
Name    : pankaj
Gender  : male
City      : delhi
Salary    : 55000
```

4. Access modifiers in Python are used to modify the default scope of variables. Write a program to demonstrate the 3 types of access modifiers: public, private and protected.

```
class Super:

    # public data member
    var1 = None

    # protected data member
    _var2 = None

    # private data member
    __var3 = None
```

```python
    # constructor
    def __init__(self, var1, var2, var3):
        self.var1 = var1
        self._var2 = var2
        self.__var3 = var3


# public member function
def displayPublicMembers(self):

        # accessing public data members
        print("Public Data Member: ", self.var1)


# protected member function
def _displayProtectedMembers(self):

        # accessing protected data members
        print("Protected Data Member: ", self._var2)


# private member function
def __displayPrivateMembers(self):

        # accessing private data members
        print("Private Data Member: ", self.__var3)


# public member function
def accessPrivateMembers(self):

        # accessing private member function
        self.__displayPrivateMembers()

# derived class
class Sub(Super):

    # constructor
    def __init__(self, var1, var2, var3):
            Super.__init__(self, var1, var2, var3)

    # public member function
    def accessProtectedMembers(self):

                # accessing protected member functions of super class
                self._displayProtectedMembers()

# creating objects of the derived class
obj = Sub("Geeks", 4, "Geeks !")

# calling public member functions of the class
obj.displayPublicMembers()
obj.accessProtectedMembers()
obj.accessPrivateMembers()

# Object can access protected member
```

Output:
Public Data Member: Geeks
Protected Data Member: 4
Private Data Member: Geeks !

1. Write a program to find the maximum and minimum K elements in Tuple using slicing and sorted() method
   (Input: test_tup = (3, 7, 1, 18, 9), k = 2, Output: (1,3, 9, 18))

Program:

```
# initializing tuple
test_tup = (3, 7, 1, 18, 9)

# printing original tuple
print("The original tuple is : " + str(test_tup))

# initializing K
K = 2

# Maximum and Minimum K elements in Tuple
# Using slicing + sorted()
test_tup = list(test_tup)
temp = sorted(test_tup)
res = tuple(temp[:K] + temp[-K:])

# printing result
print("The extracted values : " + str(res))
```

output:

```
The original tuple is : (3, 7, 1, 18, 9)
The extracted values : (1, 3, 9, 18)
```

2.
Write a program to find the size of a tuple using getsizeof() method from sys module and built-in __sizeof__() method.

Program:

```
import sys

# sample Tuples
Tuple1 = ("A", 1, "B", 2, "C", 3)
Tuple2 = ("Geek1", "Raju", "Geek2", "Nikhil", "Geek3", "Deepanshu")
Tuple3 = ((1, "Lion"), ( 2, "Tiger"), (3, "Fox"), (4, "Wolf"))

# print the sizes of sample Tuples
print("Size of Tuple1: " + str(sys.getsizeof(Tuple1)) + "bytes")
```

```
print("Size of Tuple2: " + str(sys.getsizeof(Tuple2)) + "bytes")
print("Size of Tuple3: " + str(sys.getsizeof(Tuple3)) + "bytes")
print("Size of Tuple1: " + str(Tuple1.__sizeof__()) + "bytes")
print("Size of Tuple2: " + str(Tuple2.__sizeof__()) + "bytes")
print("Size of Tuple3: " + str(Tuple3.__sizeof__()) + "bytes")
```

output:

```
Size of Tuple1: 44bytes
Size of Tuple2: 44bytes
Size of Tuple3: 36bytes
Size of Tuple1: 36bytes
Size of Tuple2: 36bytes
Size of Tuple3: 28bytes
```