Chapter Three

Data Structure and Files

Outline

Sequence

1. Lists, List Operations,

- Plotting List Data with the matplotlib Package

2. Tuples in python

3 . Python Sets

4. Python Dictionaries,

Introduction to File Input and Output(file handling)

- Using Loops to Process Files,
- Processing Records,
- Exception Handling

Data Structure

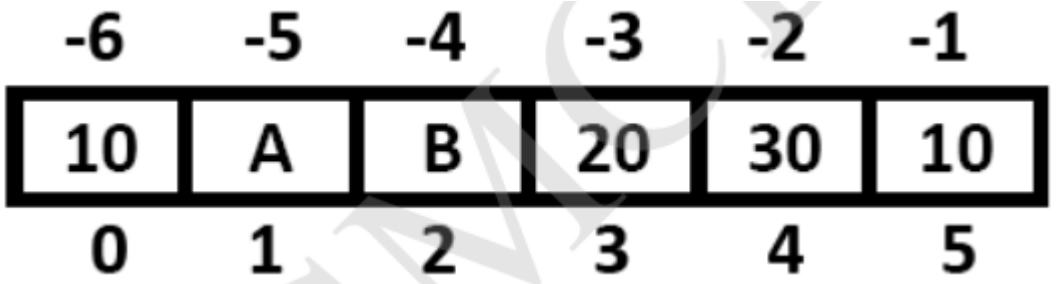
- A data structure is a way of storing data in a computer so that it can be used efficiently.
- Organizing or handling of data in computer memory and the operation that may be performed upon them.
- Consider as an example books in a library. It can be stored in a shelf arbitrarily or using some defined orders such as sorted by Title, Author and so on.
- It can be said that the study of data structures involves the storage, retrieval and manipulation of information. In other words, the possible logical arrangement of data items to provide an efficient solution for a problem is called Data Structure.

Python list data type

- A Python list contains items separated by commas and enclosed within square brackets ([]).
- Python lists are similar to arrays in C++. One difference between them is that all the items belonging to a Python list can be of different data type.
- If we want to represent a group of individual objects as a single entity where insertion order is preserved and duplicates are allowed,
- Insertion order preserved. (The elements stored and accessed in the same order)
- Duplicate objects are allowed
- Heterogeneous objects are allowed.
- List is dynamic because we can increase the size and decrease the size.

Python List data type

- In List the elements will be placed within square brackets and with comma separator.
- We can differentiate duplicate elements by using index and we can preserve insertion order by using index.
- Python supports both positive and negative indexes. +ve index means from left to right where as negative index means right to left
- Example :- list=[10,"A","B",20, 30, 10]



print(list[0]?

print(list[-1]?

print(list[3]?

print(list[-2]?

Python List operations

len( )	➢ **It returns the number of elements present in the list**
Index ()	➢ It returns the index of first occurrence of the specified item.
count( )	➢ It returns the number of occurrences of specified item in the list
append()	➢ We can use append () function to add item at the end of the list. ➢ By using this append function, we always add an element at last position
insert ()	➢ It is used to insert item at specified index position
extend()	➢ If we want to add all items of one list to another list, we use extend () method
remove ()	➢ We can use this function to remove specified item from the list. If the item present multiple times, then only first occurrence will be removed
pop ()	➢ It removes and returns the last element of the list. This is only function which manipulates list and returns some element follows LIFO (Last In First Out) order
reverse()	➢ It is used to reverse the order of elements in the list
sort( )	➢ In list by default insertion order is preserved
clear( )	➢ We can use clear() function to remove all elements of List.

Append () operation

- We can use append() function to add item at the end of the list.
- append function always add an element at last position output

Example Write a Python Program to add the following elements in to the list

- Lists=[2, IT,2024, DBU]

```
lists =[] #Declaring the empty
list
for i in range(0,4):
    lists.append(input("Enter the item:"))
print("printing the list items..")
for i in lists:
        print(i,end=" ")
```

- If you want add additional element use lists.append(2016), the element add at last position

Q1.Write a Python Program to add all elements to list up to 100 which are divisible by 10.

Insert () operation

- It is used to insert item at specified index position
- Lists=[3,6,abebe,7] , insert 88 at the 2 index?

```
Lists=[3,6,"abebe",7]
print(Lists)
Lists.insert(2,88)
print(Lists)
```

- If the specified index is greater than max index then element will be inserted at last
- If the specified _ve index is smaller than or=min index then element will be inserted at first
- If the index is negative then element will be inserted in at index-1 postion

```
Lists=[3,6,"abebe",7]
print(Lists)
Lists.insert(-1,88)
print(Lists)
```

Insert at index -2

index(),len() & count () operation

- Index():- returns the index of first occurrence of the specified item
- Count():- It returns the number of occurrences of specified item in the list
- Len():- It returns the number of elements present in the list

Example:-

```
n=[1,2,2,2,2,3,3]
print(n.index(3))
print(n.count(3))
print(len(n))
```

- If the specified element not present in the list then we will get ValueError.

extend () operation

➢If we want to add all items of one list to another list, we use extend() method.

Example 1:-

```
order1=["Chicken",“bread","Fish"]
order2=["RC","KF","FO"]
order1.extend(order2)# order1=order1+order2
print(order1)
print(order2)
```

Example 2:-

```
order=["Chicken","Mutton","Fish"]
order.extend("Mushroom")
print(order) # It adds every character as a
single element to the list
```

➢In example 2, 'Mushroom' is a string type, in this string 8 elements are there. These elements are added separately

remove() & pop () operation

- Remove():-We can use this function to remove specified item from the list. If the item present multiple times then only first occurrence will be removed

- Pop():-It removes and returns the last element of the list. This is only function which manipulates list and returns some element

```
n=[10,20,30,40]
print(n.pop())
print(n)
```

```
n=[10,20,10,30]
n.remove(10)
print(n)
```

- n.pop(index) ==> To remove and return element present at specified index.

- n.pop() ==> To remove and return last element of the list

remove()	pop()
1) We can use to remove special element from the List.	1) We can use to remove last element from the List.
2) It can't return any value.	2) It returned removed element.
3) If special element not available then we get VALUE ERROR.	3) If List is empty then we get Index Error.

reverse(),sort() and clear()operation

- Reverse():- It is used to reverse the order of elements in the list
- Sort():-In list by default insertion order is preserved(ascending order).

Example :-

```
n=[25,20,30,15]
n.reverse()
print(n)
n.sort()
print(n)
```

- To use sort() function, compulsory list should contain only homogeneous elements, otherwise we will get TypeError.

Clear():-We can use clear() function to remove all elements of List.

```
n=[25,20,30,15]
n.clear()
print(n)
```

Python tuple data type

- Tuple is exactly same as List except that it is immutable. i.e., once we creates Tuple object, we cannot perform any changes in that object. Hence Tuple is Read Only Version of List.
- If our data is fixed and never changes then we should go for Tuple.
- Insertion Order is preserved.
- Duplicates are allowed.
- Heterogeneous objects are allowed.
- We can preserve insertion order and we can differentiate duplicate objects by using index. Hence index will play very important role in Tuple also

Python tuple data type

- Tuple support both +ve and -ve index. +ve index means forward direction(from left to right) and -ve index means backward direction(from right to left).

- We can represent Tuple elements within Parenthesis and with comma separator

```
x=(2023, "Python", 3.11, 8, 22)
print(type(x))
<class 'tuple'>
```

- We can create a tuple object using tuple() function:

- If you have any sequence (i.e., string, list, range etc.,) which can be easily converted into a tuple by using tuple() function.

```
list=[10,20,30]
t=tuple(list)
print(t)
print(type(t))
```

Python tuple data type

- Which of the following are valid/Invalid tuples?

```
t=()                    # valid
t=10,20,30,40           # valid
t=10                    # not valid
t=10,                   # valid
t=(10)                  # notvalid
t=(10,)                 # valid
t=(10,20,30,40)         # valid
t= (10,20,30,)          # valid
```

- If you have any sequence (i.e., string, list, range etc.,) which can be easily converted into a tuple by using tuple() function

```
list=[10,20,30]
t=tuple(list)
print(t)
print(type(t))
```

Tuple operations

len( )	➢ **It returns the number of elements present in the list**
Index ()	➢ It returns the index of first occurrence of the specified item.
count( )	➢ It returns the number of occurrences of specified item in the tuple
max()	➢ max() function return the maximum value
min()	➢ min() function return the minimum value according to default natural sorting
sorted( )	➢ In tuple by default insertion order is preserved

```
x=(2023, 56, 3.11, 8, 22,8)
print(x.count(8))
print(x[3])
print(x.index(8))
print(sorted(x))
print(min(x))
print(max(x))
```

Set data type in python

- if we want to represent a group of unique values as a single entity then we should go for set.

Key features of Set Data Type:

- Duplicates are not allowed.
- Insertion order is not preserved. But we can sort the elements.
- Indexing and slicing not allowed for the set.
- Heterogeneous elements are allowed.
- Set objects are mutable i.e. once we creates set object we can perform any changes in that object based on our requirement.

Set data type in python

- We can represent set elements within curly braces and with comma separation.
- We can apply mathematical operations like union, intersection ,difference etc. on set objects

Declaration

Example :-

```
s = {30,40,10,5,20} # In the output order not preserved
print(type(s))
print(s)
```

Output

```
<class 'set'>
{20, 5, 40, 10, 30}
```

- While creating empty set we have to take special care. Compulsory we should use set() function. s={} ==>It is treated as dictionary but not empty set

Python set operations

add()	➢ It Adds an item 'x' to the set, single element only
update()	➢ This method is used to add multiple items to the set. ➢ Arguments are not individual elements, multiple argument.
copy()	➢ It returns copy of the set. It is cloned object (Backup copy)
pop()	➢ It removes and returns some random element from the set
remove()	➢ It removes specified element from the set. ➢ If the specified element not present in the Set then we will get Key Error
discard()	➢ It removes the specified element from the set. ➢ If the specified element not present in the set then we won't get any error
clear()	➢ It is used to remove all elements from the Set
union()	➢ We can use this function to return all elements present in both x and y sets(x | y)
intersection()	➢ This operation returns common elements present in both sets x and y.(x & y)
difference()	➢ This operation returns the elements present in x but not in y (x-y) ,(y-x)

Set data type in python

- Add, update and copy function in python set

```
s={10,20,30}
print(s)
s.add(25)
print(s)
s.add("IT")
print(s)
s.update("IT")
print(s)
s1=s.copy()
print(s1)
```

Output

Note

- s.add(10) ==> Valid
- s.add(10,20,30) ==> TypeError:
- s.update(10) ==>TypeError:
- s.update(range(1,10,2),range(0,10,2)) ==>Valid.

Set data type in python

➢Pop and remove function in python set

Pop()

```
s={70,23,67,10,20,30}
s.pop()
print(s)
s.pop()
print(s)
```

Remove()

```
s={70,23,67,10,20,30}
s.remove(20)
s.remove(10)
print(s)
```

Clear()

```
s={70,23,67,10,20,30}
s.clear()
Print(s)
```

Union operation

We can perform union operation in two ways

1. x.union(y) ==> by calling through union() method.
2. x|y ==> by using '|' operator

```
x={10,20,30}
y={5,7,15,33}
print(x|y) #or x.uniuon(y)
```

Set data type in python

➢Intersection() operation

➢We can perform intersection operation in two ways:

1. x.intersection(y) ==> by calling through intersection() method.

2. x&y ==> by using '&' operator.

```
x={10,20,30}
y={5,20,15,33}
print(x&y)
#x.intersection(y)
```

difference() operation

➢We can perform difference operation in two ways:

1. x.difference(y) ==> by calling through difference() method.

2. x-y ==> by using '-' operator.

```
x={10,20,30}
y={5,20,15,33}
print(x-y)
#x.difference(y)
```

This operation returns the elements present in x but not in y

Dictionary data type in python

- If we want to represent a group of objects as key-value pairs then we should go for Dictionary. But list set, tuple used for single entity object

Key features of Dictionary Data type:

- Duplicate keys are not allowed but values can be duplicated.
- Heterogeneous objects are allowed for both key and values.
- insertion order is not preserved.
- Dictionaries are mutable.
- Dictionaries are dynamic in nature.
- indexing and slicing concepts are not applicable

Dictionary data type in python

- Curly brackets are the simplest way to generate a Python dictionary, With many key-value pairs surrounded in curly brackets and a colon separating each key from its value, the dictionary can be built. (:). The following provides the syntax for defining the dictionary.

Declaration

Example :-

```
d = {"name":"Abebe","age":20,"Dept":"IT"}
print(type(d))
print(d)
```

Name , age, dept are keys and abebe,20,IT values

- While creating empty dictionary we should use s={} ==>It is treated as dictionary

Dictionary data type in python

- We can create an empty dictionary by using following approach also:

- Example :- d={}, We can add entries into a dictionary as follows: d[key] = value

```
d = {}
d[100]="karthi"
d[200]="sahasra"
d[300]="sri"
d['rgm'] = 'Nandyal'
print(d)
```

output :- {100: 'karthi', 200: 'sahasra', 300: 'sri', 'rgm': 'Nandyal'}

Q1 Write a Python program to enter name and percentage marks in a dictionary and display information on the screen for five students .

Dictionary data type in python

```
d=dict({100:"karthi",200:"saha"})
print(d)
d=dict([(100,"karthi"),(200,"saha"),(300,"sri")])
print(d)
d=dict(((100,"karthi"),(200,"saha"),(300,"sri")))
print(d)
d=dict({(100,"karthi"),(200,"saha"),(300,"sri")})
print(d)
d=dict(({100,"karthi"},{200,"saha"},{300,"sri"}))
print(d)
d=dict({[100,"karthi"],[200,"saha"],[300,"sri"]})
print(d)
```

Make error

Compulsory internally we need to take tuple, set only . If you take list it gives error

Dictionary operations

dict()	➢ This function is used to create a dictionary
len()	➢ It returns the number of items in the dictionary
clear()	➢ This function is used to remove all entries from the dictionary
get()	➢ It is used To get the value associated with the specified key
pop()	➢ It removes the entry associated with the specified key and returns the corresponding value
popitem()	➢ It removes an arbitrary item(key-value) from the dictionary and returns it
keys()	➢ It is used to remove all elements from the Set
values()	➢ It returns all values associated with the dictionary.
update	➢ d.update(x) All items present in the dictionary 'x' will be added to dictionary 'd'.
copy()	➢ This method is used to create exactly duplicate dictionary(cloned copy).

Dictionary get(),update() & clear() operation

get()

```
Employee = {"Name": "Dev", "Age":
20,"salary":45000,"Company":"WIPRO"}
print(Employee.get("salary"))
```

Clear()

```
d={100:"karthi",200:"sahasra",300:"sri"}
print(d)
d.clear()
print(d)
```

Update()

```
Employee = {"Name": "Dev", "Age": 20, "salary":45000,"Company":"WIPRO"}
Employee.update({"salary":8000})
print(Employee)
```

Dictionary keys(),pop() & popitem() operation

pop()

```
Employee = {"Name": "Dev", "Age":
20,"salary":45000,"Company":"WIPRO"}
Employee.pop("Age")#
print(Employee)
```

```
possible also use del Employee
("Age")
```

popitem()

```
d={100:"hana",200:"abebe",300:
"elsa"}
print(d)
print(d.popitem())
print(d)
```

Keys()

```
d={100:“hana",200:“sosi",3
00:“kebede"}
print(d.keys())
```

values()

```
d={100:"hana",200:"sosi",300:"
kebede"}
print(d.values())
```

Dictionary items() operation

items() :-It returns list of tuples representing key-value pairs like as shown below

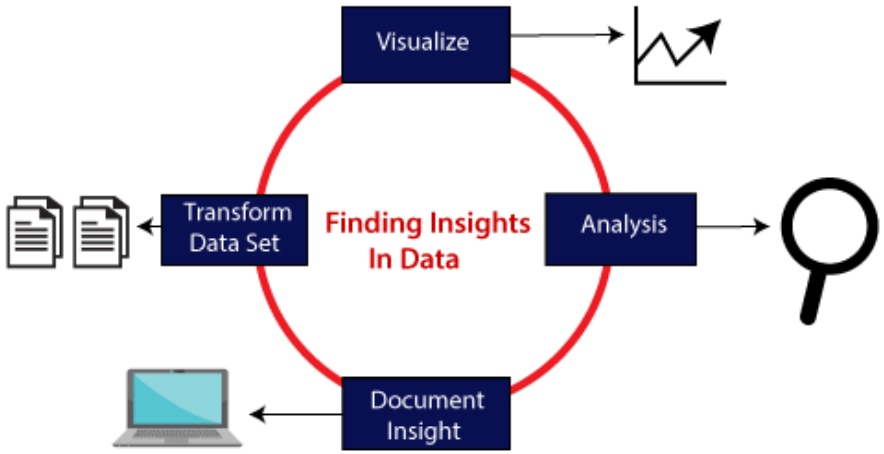
```
d={100:“hana",200:“sosi",300:“kebede"}
list = d.items()
print(list)
```

Q1. Write a Python program to take dictionary from the keyboard and print the sum of values

Q4. Write a program to accept student name and marks from the keyboard and creates a dictionary. Also display student marks by taking student name as input.

Plotting List Data with the matplotlib Package

- Matplotlib is a powerful and versatile Python library for creating static, animated, and interactive visualizations.
- It's a cornerstone for data visualization in Python and widely used in fields like science, engineering, and finance.



Data visualization can perform below tasks:

- It identifies areas that need improvement and attention.
- It clarifies the factors.
- It helps to understand which product to place where.
- Predict sales volumes.

Plotting List Data with the matplotlib Package

- Matplotlib is a Python library which is defined as a multi-platform data visualization library
- We can import from matplotlib import pyplot as plt
- The pyplot functions are used to make some changes to figure such as create a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot including labels, etc.
- The pyplot module provide the plot() function which is frequently use to plot a graph.

Plotting List Data with the matplotlib Package

1. Line graph:- is one of charts which shows information as a series of the line.

plot the following list data using line graph

x = ["USA","canada","Ethiopia","sudan","japan",“England"] , y = [40,48,57,62,74,87]

```
from matplotlib import pyplot as plt
x = ["USA","canada","Ethiopia","sudan","japan","England"]
y = [40,48,57,62,74,87]
plt.style.use("ggplot")
plt.plot(x,y)
plt.title("Line graph")
plt.ylabel('Y axis')
plt.xlabel('X axis')
plt.show()
```

Plotting List Data with the matplotlib Package

2. Bar graphs:- one of the most common types of graphs and are used to show data associated with the categorical variables. Matplotlib provides a bar() to make bar graphs

Example :- draw a bar graph using the following data

x = ["USA","canada","Ethiopia","sudan","japan",“England"] , y = [40,48,57,62,74,87]

```
from matplotlib import pyplot as plt
x = ["USA","canada","Ethiopia","sudan","japan","England"]
y = [40,48,57,62,74,87]
plt.bar(x, y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Bar Plot')
plt.show()
```

Plotting List Data with the matplotlib Package

3. The scatter plots are mostly used for comparing variables when we need to define how much one variable is affected by another variable. The data is displayed as a collection of points. Each point has the value of one variable using scatter() function to draw it ,

x = ["USA","canada","Ethiopia","sudan","japan",“England"] , y = [98,48,57,102,74,87]

```
from matplotlib import pyplot as plt
x = ["USA","canada","Ethiopia","sudan","japan","England"]
y = [98,48,57,102,74,87]
plt.scatter(x, y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Bar Plot')
plt.show()
```

Plotting List Data with the matplotlib Package

4. A histogram is used for the distribution, whereas a bar chart is used to compare different entities. A histogram is a type of bar plot that shows the frequency of a number of values compared to a set of values ranges. Used hist() function to draw the graph

data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]

```
import matplotlib.pyplot as plt
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
plt.hist(data, bins=4, edgecolor='black')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram')
plt.show()
```

Introduction to File Input and Output

- As the part of programming requirement, we have to store our data permanently for future purpose. For this requirement we should go for files.
- Files are very common permanent storage areas to store our data.
- A file is defined as placed on the disk. where the group of related data can be stored.
- file handling is an important part of any web, desktop, android application
- A file is a named location on disk to store related information. We can access the stored information (non-volatile) after the program termination.

Introduction to File Input and Output

Two types of python file

- 1. Text Files
- Text files store data as plain text characters, like letters, numbers, punctuation, and whitespace. They are human-readable. .txt .csv,.html, .py
- Python uses the built-in open() function to open text files for reading, writing, or appending. data as strings
- 2. Binary Files
- Binary files store data in a raw, binary format (sequences of 0s and 1s) that is not directly human-readable. Images (.jpg, .png, .gif).Audio (.mp3, .wav),Video (.mp4, .avi), Compressed files (.zip, .rar), Executable programs (.exe)
- How Python Handles Them:
- Python uses open() with the 'rb' (read binary) or 'wb' (write binary) mode.Data is read as bytes, not strings.

Introduction to File Input and Output

a file operation can be done in the following order.

- Open a file
- Read or write operation
- Close the file

Example file=open(“filename", "access mode”)

File :-file object or file handler, file pointer

Filename:-is the name of the file to save

Access mode:- using various modes like read, write, or append

File Access mode

1. Reading Modes ('r', 'rb')

'r' (Read Text): Opens a file for reading text data. This is the default mode.

'rb' (Read Binary): Opens a file for reading binary data

2. Writing Modes ('w', 'wb')

'w' (Write Text): Opens a file for writing text data. If the file exists, it will be overwritten.

'wb' (Write Binary): Opens a file for writing binary data. If the file exists, it will be overwritten.

File Access mode

3. Appending Modes ('a', 'ab')

'a' (Append Text): Opens a file for appending text data. If the file doesn't exist, it will be created.

'ab' (Append Binary): Opens a file for appending binary data. If the file doesn't exist, it will be created.

4. Creating New Files ('x', 'xb')

'x' (Create Text): Opens a file for writing, but only if it doesn't already exist. If the file exists, an error will be raised.

'xb' (Create Binary): Opens a file for writing binary data, but only if it doesn't already exist.

File Access mode

- r+:-It opens the file to read and write both. The file pointer exists at the beginning of the file.
- rb+:- It opens the file to read and write both in binary format. The file pointer exists at the beginning of the file.
- w+:-It opens the file to write and read both. It is different from r+ in the sense that it overwrites the previous file if one exists whereas r+ doesn't overwrite the previously written file. It creates a new file if no file exists. The file pointer exists at the beginning of the file.
- Wb+:-It opens the file to write and read both in binary format. The file pointer exists at the beginning of the file.
- a+:-It opens a file to append and read both. The file pointer remains at the end of the file if a file exists. It creates a new file if no file exists with the same name.
- ab+:-It opens a file to append and read both in binary format. The file pointer remains at the end of the file.

File Access mode

write only mode(w)

```
f1=open('stdu.txt','w')
f1.write("second year it students")
f1.close()
```

Append mode(a)

```
f1=open('stdu.txt','a')
f1.write("second year it students")
f1.close()
```

read mode(r)

```
f2=open('uuuu.txt','r')
data=f2.read()
print(data)
f2.close()
```

File Access mode

- with statement in python :-The with statement is useful in the case of manipulating the files. It is used in the scenario where a pair of statements is to be executed with a block of code in between. Example

```
with open("uuuu.txt",'r') as f:
    content = f.read();
    print(content)
```

- Here, the with statement handles opening the file open('uuuut', 'r') and closing it automatically, even if an error occurs during reading.
- The with statement is useful in the case of manipulating the files. It is used in the scenario where a pair of statements is to be executed with a block of code in between.
- we don't need to write the close() function

Read file through for loop

- We can use read() method when we open the file. Read method is also done through the for loop. We can read the file using for loop. Consider the following example.

```
f1=open('tu.txt','w')
f1.write('Computer science Engineering\n')
f1.write('Elecronics and Communication Engineering\n')
f1.write('Civil Engineering\n')
f1.close()
f2=open('tu.txt','r')
for i in f2:
    print(i)
```

Read file through for loop

- Python facilitates to read the file line by line by using a function readline() method.
- The readline() method reads the lines of the file from the beginning, i.e., if we use the readline() method two times, then we can get the first two lines of the file.

```
f1=open('tu.txt','w')
f1.write('Computer science Engineering\n')
f1.write('Elecronics and Communication Engineering\n')
f1.write('Civil Engineering\n')
f1.close()
f2=open('tu.txt','r')
data1=f2.readline()
data2=f2.readline()
print(data1)
print(data2)
f2.close()
```

Read file through for loop

- Python provides also the readlines() method which is used for the reading lines. It returns the list of the lines till the end of file(EOF) is reached..

```
f1=open('tu.txt','w')
f1.write('Computer science Engineering\n')
f1.write('Elecronics and Communication Engineering\n')
f1.write('Civil Engineering\n')
f1.close()
f2=open('tu.txt','r')
data1=f2.readlines()
print(data1)
f2.close()
```

Exception handling

- Exception handling is an essential part of writing reliable and robust Python programs.
- By using try-except blocks, you can gracefully manage errors and make your code more resilient to unexpected situations
- Exceptions: Exceptions are raised when the program is syntactically correct, but the code results in an error. This error does not stop the execution of the program, however, it changes the normal flow of the program.

```
x = 5
y = "hello“
z=x+y
Print(z)
```

```
x = 5
y = "hello"
try:
    z = x + y
except TypeError:
    print("Error: cannot
add an int and a str")
```

F strings, Join and split() in python

- F-strings (formatted string literals) are a powerful and convenient way to embed expressions and variables directly into strings in Python. Syntax:
- F-strings start with an f character before the opening quotation mark of the string.
- Inside the string, you can place expressions within curly braces {}. These expressions will be evaluated and their results inserted into the final string. Example

```
name = "Alice"
age = 30
message = f"Hello, my name is {name} and I am {age} years old."
print(message)
# Output: Hello, my name is Alice and I am 30 years old.
```

F strings, Join ,and split() in python

- The .join() method is a string method that's incredibly useful for combining elements of a list or other iterable into a single string. Here's the basic pattern:. Example

```
parts = ["hello", "world", "from", "Python"]
joined_string = " ".join(parts)  # Join with spaces
print(joined_string)  # Output: hello world from Python
```

- The split() method in Python is used to break down a string into smaller substrings.

```
# Python split() method example
# Variable declaration
str = "Java is a programming language"
str2 = str.split()
print(str)
print(str2)
```

Project

- Select your project title and submit until 19/09/2016 E.C