Week 1: Introduction to Python, Python Basics

1.What is Python?

Python is a high-level, interpreted, general-purpose programming language. Being a general-purpose language, it can be used to build almost any type of application with the right tools/libraries. Additionally, python supports objects, modules, threads, exception-handling, and automatic memory management which help in modelling real-world problems and building applications to solve these problems.

2.What is an Interpreted language?

An Interpreted language executes its statements line by line. Languages such as Python, Javascript, R, PHP, and Ruby are prime examples of Interpreted languages. Programs written in an interpreted language runs directly from the source code, with no intermediary compilation step.

3.Which version of Python is currently up to date?

PYTHON 3.9

4.**In a Python context, the acronym IDLE stands for__?**

Integrated Development and Learning Environment

5.**When you see >>> inside IDLE, it means that:**

Python is waiting for you to give it some instructions

6.**In python how many keywords are reserved internally?**

33

7.**Find the output of the code fragment given below?**

Code starts

```
@value = 2
value = ‘skill’


print(value**@value)


# Code Ends
```

a.Invalid Syntax

b.hellohello

c.Hello2

d.None of these.

Answer: a. Invalid Syntax

8.How the python is memory efficient ?

Python uses its private heap space to manage the memory.Basically, all the objects and data structures are stored in the private heap space. Even the programmer can not access this private space as the interpreter takes care of this space. Python also has an inbuilt garbage collector, which recycles all the unused memory and frees the memory and makes it available to the heap space.

9.Who invented Python Programming Language ?

Guido van Rossum

10. Is Python case sensitive?

Yes, Python case sensitive language. This means that Function is function both are different in pythons like SQL and Pascal

11.What type of language is python? Programming or scripting ?

Generally, Python us an all purpose Programmimng Language, In addition to that pythhon is also Capable to perform scripting.

12.How to remove whitespaces from a string in Python?

Using strip([str2]) in-built function.

13.What do you mean by Python literals?

Literals can be defined as a data which is given in a variable or constant.

They are two types of literals

a)Numeric Literals

b)String Literals

14.What are the advantages of Python?

Advantages of Python are:

Python is Interpreted language

It is Free and open source

It is Extensible

Object-oriented

It has Built-in data structure

Readability

High-Level Language

Cross-platform

Portable

15.Which of the following doesn't require an identifier for its recognition?

Import Statements

16.In the given set choose the set with all valid identifiers?

a.True and val b.Val and val c.@wert and 4rt d._point and value

Answer: b.val and val

17.Which of the following is not a valid identifier?

a.C_score

b.CScore

c.Cscore

d.all correct

Answer: d. all correct

18.Which of the following is not a keyword in Python?

a.def

b.or

c.yield

d.None of these

Answer:d. None of these

19.Find the output of the code fragment given below?

Code starts

@value = 2

value = 'skill'

```
print(value**@value)


# Code Ends
```

a.Invalid Syntax

b.hellohello

c.Hello2

d.None of these.

Answer: a.Invalid Syntax

Week 2:Strings, Decision control statements

1.What is an operator in Python?

An operator is a particular symbol which is used on some values and produces an output as a result.

They are three types of operator:

a) Unary Operator

b) Binary Operator

c) Ternary Operator

2.**What are the different types of operators in Python?**

a) Arithmetic OperatorsRelational Operators

b) Assignment Operators

c) Logical Operators

d) Membership Operators

e) Identity Operators

f) Bitwise Operators

3.Find the output of the code given below?

```
# Code starts

val = ‘Skill’
Val = 'lync'

print(“We work @ “,val+Val)

# Code Ends
```

a.Syntax Error

b.We work @ SkillSkill

c.We work @LyncSkill

d.We work @SkillLync

d.We work @SkillLync

4.What will be the output of the following code snippet?

```
# Code starts


true = 34

false = 90


valTotal = 67-true+63*21-false*12

print(valTotal)

# Code Ends
```

276

5.**What is the data type of print(type(0xFF))**

Int

6.**What is the result of print(type([]) is list)**

True

7. **What are comments and how can you add comments in Python?**

Comments in Python refer to a piece of text intended for information.

It is especially relevant when more than one person works on a set of codes. It can be used to analyse code, leave feedback, and debug it. There are two types of comments which includes:

Single-line comment

Multiple-line comment

Codes needed for adding a comment

#Note –single line comment

“””Note

Note

Note”””—multiline comment

8. **What is the output of the following code:**

```
strOne = str("pynative")
strTwo = "pynative"
print(strOne == strTwo)
print(strOne is strTwo)
```

True

True

9. Which method should I use to convert String "welcome to the beautiful world of python" to "Welcome To The Beautiful World Of Python"

title()

10.How would you check if each word in a string begins with a capital letter?

The istitle() function checks if each word is capitalized.

for example: print('The Hilton'.istitle()) #=> True

print('The dog'.istitle()) #=> False

11.Write A program to count the number of a specific character in a string

```
str=input()
print(len(str))
```

12.Write a program to reverse the WORDS in a string

```
string =input()
s = string.split()[::-1]
l = []
for i in s:
    l.append(i)
print(" ".join(l))
```

13.Write a program to display last digit of a number

```
number=int(input("enter the number"))
print("last digit of the number is ",number%10)
```

14.How to check a string contains only numbers?

Python has a isdigit() method to check if any string contains any number or not.,”Log”]

```
string1 = '123456789'
string2 = 'quescol123'
```

```
if string1.isdigit():
    print ("String1 contains only numbers")
else:
    print ("String1 doesn't contains only numbers")
if string2.isdigit():
    print ("String2 contains only numbers ")
else:
    print ("String2 doesn't contains only numbers ")
```

15. Python program to split and join a string

```
s = 'Welcome to Skill-lync'
# print the string after split method
print(s.split(" "))
# print the string after join method
print("-".join(s.split()))
```

16.What are the common built-in data types in Python ?

Immutable data types:

- number
- string
- tuple

mutable data type

- list
- dictionary
- sets

17.Python Program to Convert Kilometers to Miles

```
kilometers = float(input("Enter value in kilometers: "))
conv_fac = 0.621371
miles = kilometers * conv_fac
print('%0.2f kilometers is equal to %0.2f miles' %(kilometers,miles))
```

18.Identity operators in Python

```
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'
x3 = [1,2,3]
y3 = [1,2,3]

print(x1 is not y1)
print(x2 is y2)
print(x3 is y3)
```

19.Python (!=) Not Equal Comparison Operator

This Not equal operator is used to compare two operands and returns True statement if they are Not equal, and False if they are Equal.

```
x = 20
y = 25
# Output: x != y is True
print('x != y is',x!=y)
```

20.Replace to K into "-" at ith Index in String ?

```
test_str = 'Skill5Lync'
```

```
print("The original string is : " + str(test_str))
K = '-'
i = 5
res = test_str[: i] + K + test_str[i + 1:]
print("The constructed string : " + str(res))
```

Week 3: Repetition Statements, Console Input-Output.

1.What will be the output of the following code snippet?

```
# Code starts
true = 34
false = 90
valTotal = 67-true+63*21-false*12
print(valTotal)
# Code Ends
```

276

2.Write a program to reverse the WORDS in a string

```
string =input()
s = string.split()[::-1]
l = []
for i in s:
    l.append(i)
print(" ".join(l))
```

3. Given the nested if-else below, what will be the value x when the code executed successfully

```
x = 0
a = 5
b = 5
if a > 0:
  if b < 0:
    x = x + 5
  elif a > 5:
    x = x + 4
```

```
    else:
        x = x + 3
else:
    x = x + 2
print(x)
```

4.Write a program to reverse a number in Python?

```
n=int(input("Enter number :"))
rev=0
while(n>0):
    dig=n%10
    rev=rev*10+dig
    n=n//10
print("The reverse of the number:",rev)
```

5. Write a program in Python to produce Star triangle?

```
n=int(input("enter a number :"))
for x in range(n):
    print(' '*(n-x-1)+'*'*(2*x+1))
```

6.Simple interest formula is given by: Simple Interest = (P x T x R)/100 Where, P is the principle amount T is the time and R is the rate

```
principle, rate, time = 10000, 10.25, 5

    Amount = principle * (pow((1 + rate / 100), time))
    CI = Amount - principle
    print("Compound interest is", CI)
```

7.Python program for removing i-th character from a string?

```
string1 = "Skill-Lync"
for j in range(len(string1)):
        if j == i:
            string = string1.replace(string1[i], "", 1)
            i = 5
print(string1, i)
```

8.write a program to check Break condition in while loop.

```
for letter in 'Python':     # First Example
   if letter == 'h':
      break
   print('Current Letter :', letter)

var = 10                    # Second Example
while var > 0:
   print('Current variable value :', var)
   var = var -1
   if var == 5:
      Break
```

9.Write a program to check a person is eligible for voting or not

```
            age=int(input("enter your age"))
            if age>=18:
                        print("Eligible for voting")
            else:
                        print("Not eligible foe voting")
```

10.Write a program to check whether a number is divisible by 7 or Not

```
N=int(input("enter the number")
if N%7==0:
        print("number is divisible by 7")
else:
        print("number is not divisible by 7")
```

11.Write a program to print first 10 natural numbers except 7

```
for i in range(1,11):
        if i == 7:
        continue
print("The Number is :" , i)
```

12.How to check a string contains only numbers?

Python has a isdigit() method to check if any string contains any number or not.,”Log”]

```
string1 = '123456789'
string2 = 'quescol123'
if string1.isdigit():
   print ("String1 contains only numbers")
else:
   print ("String1 doesn't contains only numbers")
if string2.isdigit():
   print ("String2 contains only numbers ")
else:
   print ("String2 doesn't contains only numbers ")
```

13.Why do we need a break?

Break helps control the Python loop by breaking the current loop from execution and transferring the control to the next block.

14.Why do we need a continue?

A continue helps control the Python loop by making jumps to the next iteration of the loop without exhausting it.

15.Python Program to demonstrate String slicing

```
String1 = "Welcome to Skill-Lync"
print("Initial String: ")
print(String1)
print(String1[0:3])
print(String1[10:])
```

16.Write a program to check Break condition in while loop.

```
for letter in 'Python':     # First Example
  if letter == 'h':
     break
  print('Current Letter :', letter)
 var = 10                  # Second Example
while var > 0:
  print('Current variable value :', var)
  var = var -1
  if var == 5:
     Break
```

17.Python Program to Swap Two Variables

```
x = input('Enter value of x: ')
```

```
y = input('Enter value of y: ')
temp = x
x = y
y = temp
print('The value of x after swapping: {}'.format(x))
print('The value of y after swapping: {}'.format(y))
```

18. Python program to split and join a string

```
s = 'Welcome to Skill-lync'
# print the string after split method
print(s.split(" "))
# print the string after join method
print("-".join(s.split()))
```

19.Write code to Calculate frequency of characters in a string?

```
Str1 = input('Enter the string :')
Character = input('Enter character :')
freq = 0
freq = Str1.count(Character)
print(str(freq) + ' is the frequency of given character')
```

20.Python Program to demonstrate String slicing

```
String1 = "Welcome to Skill-Lync"
print("Initial String: ")
print(String1)
Sprint(String1[0:3])
print(String1[10:])
```

WEEK 4 :List, Tuples, Set, Dictionary

1. How to count unique values within a list?

(a)

```
input_list = [10, 11, 10, 13, 12, 14, 12]
new_list = []
for number in input_list:
        if number not in new_list:
                count += 1
                new_list.append(number)
print('No of unique items are: ', count)
```

(b)

```
input_list = [10, 11, 10, 13, 12, 14, 12]
new_set = set(input_list)
print('No of unique items are: ', len(new_set))
```

2. Given 3 digits a, b and c. Find all possible combinations from these digits.

```
digits = ['a', 'b', 'c']
for i in range(3):
   for j in range(3):
        for k in range(3):
           if i != j and j != k and i != k:
                print(digits[i], digits[j], digits[k])
```

3.Iterate through the list and print the longest string in the list?

```
string_list = ['Hello', 'everyone', 'let"s', 'learn', 'from', 'skill-lync']
longest_string = []
```

```
for string in string_list:
    if len(string) > len(longest_string):
        longest_string = string
    print(longest_string)
```

4..Convert list of characters to a string?

```
character_list = ['S', 'k', 'i', 'l', 'l', '-', 'L', 'y', 'n', 'c']
result = "".join(character_list)
print('The string is :', result)
```

5. Remove duplicates from a list?

(a)

```
number_list = [1, 3, 5, 7, 1, 3, 5, 9]
no_duplicate = []
for i in number_list:
    if i not in no_duplicate:
        no_duplicate.append(i)
 print('Unique numbers in list are: ',no_duplicate)
```

(b)

```
number_list = [1, 3, 5, 7, 1, 3, 5, 9]
no_duplicate = list(set(number_list)))
print('Unique numbers in list are: ',no_duplicate)
```

6. Check if a list is empty?

```
list1 = [1, 2, 4]
if len(list1) == 0 :
```

```
    print('The list is empty.')
else:
    print('The list is not empty.')
```

7. Reverse the list?

(a) Using reverse() function

```
list1 = [1, 2, 3, 4, 5, 6]
list1.reverse()
print('The reversed list is:', list1)
```

(b) Using list slicing

```
list1 = [1, 2, 3, 4, 5, 6]
reversed_list = list1[::-1]
print('The reversed list is:', reversed_list)
```

8. Reverse the key : value of dictionary to value : key and sort the result in ascending order?

```
my_dict = {'a': 91, 'b': 82, 'c': 3, 'd': 64, 'e' : 42}
new_dict ={}
for k, v in my_dict.items():
    new_dict[v] = k
print(sorted(new_dict))
```

9. Convert 2 list into a dictionary?

```
list1 = ['Apple', 'Banana', 'Watermelon']
list2 = [10, 20, 5]

my_dict = dict(zip(list1, list2))
print('The dictionary is: ', my_dict)
```

10. What are the different mathematical operations on sets?

```
set1 = {10, 20, 30}
set2 = {30, 40, 50}
print('The union of sets', set1|set2)
print('The intersection of sets', set1&set2)
print('The difference of sets', set1-set2)
print('The symmetric difference of sets', set1^set2)
```

11- Sort the tuple of tuples by first item?

```
tup1 = ((20, 'a'), (32, 'b'), (45, 'c'), (67, 'd'))
tup1 = tuple(sorted(list(tup1)))
print(tup1)
```

12- What is difference between remove and pop in lists?

REMOVE - Removes the first item from the list whose value is mentioned

It will modify the original list itself

Raises VALUEERROR if there is no such item

POP-

Removes the item at the given position and returns it.

If index is not mentioned it removes and returns the last element in the list

It will return the element removed from the list.

13. What is difference between indexing and slicing?

INDEXING - used to obtain individual elements.

- Attempting to use index that is too large will result in IndexError.

- returns 1 item.

- starts from 0 . Negative indexing starts from -1

SLICING - used to obtain sequence of elements.

- returns new list.

- Out of range indexed are handelled well, doesn't result in an error.

- can specify range of indexes.

Q14 - How to find number of occurances of an element in a python list?

```
list1 = [1, 3, 4, 1, 5, 6, 3, 7, 8]
print('No of occurances of 1 is', list1.count(1))
```

15. What is Python Dictionary?

Python Dictionary is an in-built collection data type that does not contain duplicates generally known as an associative array.

It is an ordered collection.A dictionary consists of a collection of key-value pairs. Each key-value pair maps the key to its associated value.

Dictionary is listed in curly brackets, inside these curly brackets, keys and values are declared.

Each key is separated from its value by a colon (:), while commas separating each element.

Dictionary contains unique keys, but values can be duplicated.

We can’t use the index to access the dictionary element else use the keys because Dictionary is indexed by keys.

16. How to remove item from Python Dictionary?

pop(<key>) - Removes the entry with the specified key

- Returns the value of key being deleted.

popitem() - removes last inserted item in the dictionary.

del()- removes the item with specified key from the dictionary

- just deletes the item, doesn't returns the associated value.

clear() - removes all the entries.

17. What is the difference between list, tuple, sets and dictionary?

List - mutable ordered heterogenous collection of objects. Mutable means we can add & remove items to a list.

- Ordered - order of the list is maintained. Depicted by []

Tuple - immutable ordered collection of objects. Each element in a tuple has a specific position(index) and position doesn't change over time.

Not possible to add / remove elements from tuple once it has been created.

Sets - Unordered collection of immutable objects. Sets doesnt allow duplicates. Sets dont have a defined order.

Dictionary -set of associations or mapping between key and value that is unordered, mutable and indexed by key values.

keys must be unique but values dont need to be unique.

18. Write a program to perform sum of tuple elements?

a) Using for loop

```
tup1 = (12, 15, 17, 19, 32, 4)
tup_sum = 0
for i in tup1:
    tup_sum += i
print('The sum of tuple elements: ',tup_sum)
```

b) Using list() + sum()

```
tup1 = (12, 15, 17, 19, 32, 4)
tup_sum = sum(list(tup1))
print('The sum of tuple elements: ',tup_sum)
```

19, Write a code to check if the input string is binary or not?

```
my_str = set(input('Enter a string'))
val = {'0', '1'}
if val == my_str or my_str == {'0'} or my_str == {'1'}:
    print('The string is binary')
else:
    print('The string is not binary')
```

20. Write a program to check if string contains all vowels or not?

```
str1 = input('Enter a string: ')
vowels = ['A','E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u']
for char in str1:
    if char not in vowels:
        print(f"{str1} : Doesn't contains all vowels")
        break
else:
    print(f"{str1} : Contains all vowels")
```

21. Check if string is panagram or not ?

a) Using IF- ELSE

```
string = input('Enter a string: ')
alphabets = "abcdefghijklmnopqrstuvwxyz"

for char in alphabets:
    if char not in string.lower():
        print('No', string, 'is not a panagram')
        break
```

```
else:
    print(string, 'is a panagram')
```

b) Using sets

```
string = input('Enter a string: ')
alphabets = "abcdefghijklmnopqrstuvwxyz"
if set(string.lower()) >= set(alphabets):
    print('Panagram')
else:
    print('Not a Panagram')
```

WEEK 5 : Functions and Recursion, Functional Programming and Lambda functions

1. Define function in python.

A function is a block of code which only runs when it is called.

2. Difference between arguments and parameters.

A **parameter** is the variable listed inside the parentheses in the function definition.

An **argument** is the value that are sent to the function when it is called.

3. How will you define a function in python ?

A function can be defined as **def** keyword

Syntax for defining function is : **def funcname():**

4. What is global and local variables?

Local variables are those which can be accessed only inside the function in which they are declared,

Global variables are those which can be accessed throughout the program body by all functions.

5. Create a function that returns sum of two numbers

```
def sum(a,b):
        a+b
sum(2,3)
```

6.. Create a function that calculates area of circle

```
def area (r):
    Pie = 3.14
    Circle_area = Pie*(r**2)
```

```
        print( Circle_area)
area(2)
```

7.. Create a function which checks string is palindrome or not

```
def ispalindrome(s):
    if s == s[::-1]:
        print(" it is palindrome)
    else:
        print(" it is not palindrome)
ispalindrome("Malayalam")
```

8. Write a Python function to calculate the factorial of a number

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
print(factorial(5))
```

9.Write a Python function that takes a number as a parameter and check the number is prime or not.

```
def test_prime(n):
    if (n==1):
        return False
    elif (n==2):
        return True;
    else:
        for x in range(2,n):
            if(n % x==0):
                return False
        return True
print(test_prime(9))
```

10.Write a Python function to check whether a number falls in a given range.

```
def test_range(n):
```

```
    if n in range(3,9):
        print( " %s is in the range"%str(n))
    else :
        print("The number is outside the given range.")
test_range(5)
```

11. What do you understand by lambda function in python ?

A lambda function is **a small anonymous function**. A lambda function can take any number of arguments, but can only have one expression.

Syntax : lambda args: expression

12. Create a lambda function that multiplies argument x with argument y and print the result.

```
a=lambda x,y =  x*y
print(a(10,4))
```

13. Create a list and sort list using lambda

```
l1 = [ ]
l1.sort(key = lambda x: x[1])
```

14.Filter the even numbers from given list

```
l1 =[ 1,2,3,4,5,6,7,8]
list(filter(lambda x: x%2 == 0, l1 ))
```

15.Write a Python program to find whether a given string starts with a given character using Lambda.

```
starts_with = lambda x: True if x.startswith('P') else False
print(starts_with('Python'))
```

16. What is the main difference between map and filter function in python?

map creates a new array by transforming every element in an array individually.

filter creates a new array by removing elements that don't belong.

17. Write a Python program to add three given lists using Python map and lambda.

```
a= [ 3 , 4 , 5 ]
b= [ 4 , 5 , 6 ]
c= [ 10 , 1 , 5 ]
result = map(lambda x, y, z: x + y + z, a,b,c)
print ( result )
```

18. Write a Python program to find intersection of two given arrays using Lambda.

```
 array_nums1 = [1, 2, 3, 5, 7, 8, 9, 10]
array_nums2 = [1, 2, 4, 8, 9]
result = list(filter(lambda x: x in array_nums1, array_nums2))
print (" Intersection of the said arrays: ",result)
```

19.Program to access function inside function

```
    def test(a):
      def add(b):
        nonlocal a
        a += 1
        return a+b
      return add
func= test(4)
print(func(4))
```

Q..20 Write a Python program to converting an integer to a string in any base.

```
def to_string(n,base):
  conver_tString = "0123456789ABCDEF"
  if n < base:
```

```
        return conver_tString[n]
    else:
        return to_string(n//base,base) + conver_tString[n % base]

    print(to_string(2835,16))
```

21.Define Reduce function

The **reduce** function is used to **apply a particular function passed in its argument to all of the list elements** mentioned in the sequence passed along.

22.In which module , reduce function is present ? Write a code for it.

It is present in **functool** module

```
import functool

functool.reduce()
```

23. How to find sum of all number in the list using reduce.

```
lis = [1, 3, 5, 6, 2]

print(functools.reduce(lambda a, b: a+b, lis))
```

24. Use recursion to solve the Fibonacci sequence in python.

```
def fibonacci(n):

    if n == 1 or n == 2:
        return 1
    else:
        return (fibonacci(n - 1) + (fibonacci(n - 2)))

print(fibonacci(7))
```

WEEK 6: File Input-Output, Modules

1. What is file handling in Python?

File handling is the process of reading, writing, and manipulating files in Python. This can be done with the built-in open() function, which takes two arguments: the path to the file and the mode in which to open it. The mode argument specifies how the file should be opened, and can be 'r' for reading, 'w' for writing, or 'a' for appending.

2. Can you explain the different modes of opening a file in Python?

There are three different modes of opening a file in Python: read mode, write mode, and append mode. Read mode is the default mode and allows you to read the contents of a file. Write mode will allow you to write to a file, and append mode will allow you to add new data to the end of an existing file.

3. How do you create a text file using Python?

You can create a text file using Python by opening a new file in write mode. This will allow you to write data to the file.

4. How do you read and write to an existing file in Python?

To read an existing file in Python, you can use the built-in open() function. This function takes two arguments: the name of the file to read, and the mode in which to open the file. The mode argument is optional, but it allows you to specify how you want to interact with the file. The most common mode is 'r', which stands for read-only. To write to an existing file, you can use the same open() function, but you must specify the 'w' mode argument. This will allow you to write to the file.

5. Is it possible to open multiple files using Python? If yes, then how?

Yes, it is possible to open multiple files using Python. You can do this by using the built-in open() function. When you call open(), you can specify the mode in which you want to open the file, as well as the name of the file. If you want to open multiple files, you can do so by passing a list of filenames to open().

6. How should you handle exceptions when dealing with files in Python?

When working with files in Python, it is important to be aware of potential exceptions that could be raised. Some common exceptions include IOError, which is raised when there is an error reading or writing to a file, and ValueError, which is raised when there is an issue with the format of the data in the file. It is important to handle these exceptions appropriately in order to avoid potential data loss or corruption.

7. What are some important methods used for reading from a file in Python?

Some important methods used for reading from a file in Python are the read() and readline() methods. The read() method reads the entire contents of a file into a string, while the readline() method reads a single line from a file.

8. What are some common errors that can occur while working with files in Python?

Some common errors that can occur while working with files in Python include trying to open a file that does not exist, or trying to read from a file that has not been opened. Other errors can occur if you try to write to a file that is read-only, or if you try to close a file that is already closed.

9. What's the difference between binary and text files in Python?

Binary files are files that are not meant to be read by humans – they are meant to be read by computers. Binary files are typically faster to read and write than text files because the computer can read the entire file in one go, without having to stop to process the data. Text files, on the other hand, are meant to be read by humans. They are typically slower to read and write than binary files because the computer has to stop to process the data.

10. Why is it not recommended to use pickle or cPickle modules for serializing objects into a file?

The pickle and cPickle modules are not recommended for use because they are insecure. Because these modules can execute arbitrary code, they could be used by an attacker to compromise the security of your system.

11. Which functions allow us to check if we have reached the end of a file in Python?

The functions that allow us to check if we have reached the end of a file in Python are the eof() and tell() functions. The eof() function returns true if we have reached the end of the file, and the tell() function returns the current position of the file pointer.

12. When trying to open a file in Python, I get the error "FileNotFoundError: [Errno 2] No such file or directory". What could be the cause of this issue?

There are a few potential causes for this error. The first is that the file you are trying to open does not actually exist in the directory you are looking in. The second is that you do not have permission to access the file. The third is that the file is already open by another process.

13. How can you convert a CSV file to a dataframe in Pandas?

You can use the read_csv() function in Pandas to convert a CSV file to a dataframe.

14. How can you sort a list of strings alphabetically without using sorted()?

You can use the "sort" method.

15. List down the steps involved in processing a large file in Python?

There are a few steps involved in processing a large file in Python:

- Open the file using the open() function.

- Read in the file using either the read() or readline() function.
- Process the data in the file as needed.
- Close the file using the close() function.

16. What does the following code snippet do?

The following code snippet opens the file "myfile.txt" for reading, and then reads in the contents of the file line by line, printing each line out as it goes.

17. Explain the parameters [a = open("file", 'w')]?

The open() function opens the file in write mode. The 'w' character stands for write. The file will be created if it doesn't exist. If it exists, the data in the file will be overwritten.

18. .Explain the parameters [a.write('Hello World')]?

This code will write the string 'Hello World' to the file 'a'.

19. What is the best way to extract all lines that start with a particular letter?

One way to do this would be to use the Python built-in open() function to open the file, then use the for loop to iterate through each line in the file. For each line, you can use the Python startswith() method to check if the line starts with the desired letter. If it does, you can add it to a list or print it out.

20. What are some examples of sequence types in Python?

Some examples of sequence types in Python include lists, tuples, and strings.

21. What is the usage of yield keyword in Python?

The yield keyword is used in Python to define generators. Generators are a special type of function that allow the programmer to return values one at a time, rather than returning all values at once. This can be useful when working with large data sets, as it allows the programmer to process the data one piece at a time, rather than having to load the entire data set into memory at once.

22. What is the purpose of enumerate() function in Python?

The enumerate() function is used to iterate over a list of items, while keeping track of the index of each item in the list. This can be useful when you need to know not only the contents of the list, but also where each item is located in the list.

23.What is Serialization?

What is serialization in Python?

Object serialization is the process of converting state of an object into byte stream. This byte stream can further be stored in any file-like object such as a disk file or memory stream.

24.What is Deserialization?

Deserialization is the process of reconstructing the object from the byte stream.

25. What is JSON data?

JSON, or JavaScript Object Notation, is a minimal, readable format for structuring data. In JSON data is nothing but a information. It is used primarily to transmit data between a server and web application.

26. What is Python pickle used for?

Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.

27. What are Dump and Dumps method?

json.dump() method used to write Python serialized object as JSON formatted data into a file.

json.dumps() method is used to encodes any Python object into JSON formatted String.

28. What are Loads and load?

json.load() takes a file object and returns the json object. It is used to read JSON encoded data from a file and convert it into a Python dictionary and deserialize a file itself i.e. it accepts a file object.

json.loads() method can be used to parse a valid JSON string and convert it into a Python Dictionary. It is mainly used for deserializing native string, byte, or byte array which consists of JSON data into Python Dictionary.

WEEK 7:Classes and Objects

1. **What is the purpose of using dir() function?**

   Ans. The dir() function returns all properties and methods of the specified object, without the values. This function will return all the properties and methods, even built-in properties which are default for all object.

2. **What is the difference between dir() and Var() function?**

   Ans. dir() Function:
   This function displays more attributes than vars() function, as it is not limited to an instance. It displays the class attributes as well. It also displays the attributes of its ancestor classes.
   vars() Function:
   This function displays the attribute of an instance in the form of a dictionary.
   The below table provides with some significant difference between var() and dir():

3. **What is operator Overloading ?**

   Ans. Operator Overloading means giving extended meaning beyond their predefined operational meaning. For example operator + is used to add two integers as well as join two strings and merge two lists. It is achievable because '+' operator is overloaded by int class and str class. You might have noticed that the same built-in operator or function shows different behavior for objects of different classes, this is called Operator Overloading.

4. **How is a constructor defined in a Python class?**

   Ans. A constructor is defined by using __init__()

5. **Does Python support constructor overloading?**

   Ans. No

6. **Which function overloads the + operator?**

   Ans. __add__()

7. **What are Benefits of inheritance ?**

   It represents real-world relationships well.
   It provides the reusability of a code. We don't have to write the same code again and again. Also, it allows us to add more features to a class without modifying it.
   It is transitive in nature, which means that if class B inherits from another class A, then all the subclasses of B would automatically inherit from class A.

8. **What is a child class?**

   Ans. It is another class which is going to inherit the properties of the base class.

9. What are the Different types of Inheritance?

Ans. Single inheritance: When a child class inherits from only one parent class, it is called single inheritance. We saw an example above.

Multiple inheritances: When a child class inherits from multiple parent classes, it is called multiple inheritances.

10. How to check if a class is subclass of another?

Ans. Python provides a function is subclass() that directly tells us if a class is subclass of another class.

11. What do you know about Code reuse?

Ans. A Python function can be defined once and used many times. So it aids in code reuse: you don't want to write the same code more than once.

12. What is the purpose of Constructors?

Ans. Constructors are used to initialize the object's state. Like methods, a constructor also contains a collection of statements that are executed at the time of Object creation.

13. What are the main features of OOPs?

Ans. Inheritance

Encapsulation

Polymorphism

Data Abstraction

14. What is the difference between a class and a structure?

Class: User-defined blueprint from which objects are created. It consists of methods or set of instructions that are to be performed on the objects.

Structure: A structure is basically a user-defined collection of variables which are of different data types.

15. Can you call the base class method without creating an instance?

Ans. Yes, you can call the base class without instantiating it if:

It is a static method

The base class is inherited by some other subclass

16. What is the difference between a class and an object?

Ans. Object is an instance of a class. Class is a blueprint or template from which objects are created.

17. What is hybrid inheritance?

Ans. Hybrid inheritance is a combination of multiple and multi-level inheritance.

18. What is hierarchical inheritance?

Ans. Hierarchical inheritance refers to inheritance where one base class has more than one subclasses. For example, the vehicle class can have ‘car’, ‘bike’, etc as its subclasses.

19. What are the limitations of inheritance?

Ans. Increases the time and effort required to execute a program as it requires jumping back and forth between different classes

The parent class and the child class get tightly coupled

20. What is a superclass?

Ans. A superclass or base class is a class that acts as a parent to some other class or classes. For example, the Vehicle class is a superclass of class Car.

21. What is method overloading?

Ans. Method overloading is a feature of OOPs which makes it possible to give the same name to more than one method within a class if the arguments passed differ**.**

22. What is method overriding?

Ans. Method overriding is a feature of OOPs by which the child class or the subclass can redefine methods present in the base class or parent class. Here, the method that is overridden has the same name as well as the signature meaning the arguments passed and the return type.

23. What are virtual functions?

Ans. Virtual functions are functions that are present in the parent class and are overridden by the subclass. These functions are used to achieve runtime polymorphism.

24. What is an exception?

Ans. An exception is a kind of notification that interrupts the normal execution of a program. Exceptions provide a pattern to the error and transfer the error to the exception handler to resolve it. The state of the program is saved as soon as an exception is raised.

25. What is the difference between an error and an exception?

Ans. The error indicates trouble that primarily occurs due to the scarcity of system resources. The exceptions are the issues that can appear at runtime and compile time.

WEEK 8: Exception Handling, Iterators And Generators

1. How many except statements can a try-except block have?
a) zero
b) one
c) more than one
d) more than zero

Answer: d
Explanation: There has to be at least one except statement.

2. When will the else part of try-except-else be executed?
a) always
b) when an exception occurs
c) when no exception occurs
d) when an exception occurs in to except block

Answer: c
Explanation: The else part is executed when no exception occurs.

3. Is the following Python code valid?

```
try:
    # Do something
except:
    # Do something
finally:
    # Do something
```

a) no, there is no such thing as finally
b) no, finally cannot be used with except
c) no, finally must come before except
d) yes

Answer: d

4. Is the following Python code valid?

```
try:
    # Do something
except:
    # Do something
else:
    # Do something
```

a) no, there is no such thing as else
b) no, else cannot be used with except
c) no, else must come before except
d) yes

Answer: d

5. Can one block of except statements handle multiple exception?
a) yes, like except TypeError, SyntaxError [,…]
b) yes, like except [TypeError, SyntaxError]
c) no
d) none of the mentioned

Answer: a
Explanation: Each type of exception can be specified directly. There is no need to put it in a list.

6. When is the finally block executed?
a) when there is no exception
b) when there is an exception
c) only if some condition that has been specified is satisfied
d) always

Answer: d
Explanation: The finally block is always executed.

7. What will be the output of the following Python code?

```
def foo():
    try:
        return 1
    finally:
        return 2
k = foo()
print(k)
```

a) 1
b) 2
c) 3
d) error, there is more than one return statement in a single try-finally block

Answer: b
Explanation: The finally block is executed even there is a return statement in the try block.

8. What will be the output of the following Python code?

```
def foo():
```

```
    try:
        print(1)
    finally:
        print(2)
foo()
```

a) 1 2
b) 1
c) 2
d) none of the mentioned

Answer: a
Explanation: No error occurs in the try block so 1 is printed. Then the finally block is executed and 2 is printed.

9. What will be the output of the following Python code?

```
try:
    if '1' != 1:
        raise "someError"
    else:
        print("someError has not occurred")
except "someError":
    print ("someError has occurred")
```

a) someError has occurred
b) someError has not occurred
c) invalid code
d) none of the mentioned

Answer: c
Explanation: A new exception class must inherit from a BaseException. There is no such inheritance here.

10. What happens when '1' == 1 is executed?
a) we get a True
b) we get a False
c) an TypeError occurs
d) a ValueError occurs
View Answer
Answer: b
Explanation: It simply evaluates to False and does not raise any exception.

11.The following Python code will result in an error if the input value is entered as -5.

```
assert False, 'Spanish'
```

a) True

b) False

Answer: a
Explanation: The code shown above results in an assertion error. The output of the code is:
Traceback (most recent call last):
File "<pyshell#0>", line 1, in <module>
assert False, 'Spanish'
AssertionError: Spanish
Hence, this statement is true.

12. What will be the output of the following Python code?

```
x=10
y=8
assert x>y, 'X too small'
```

a) Assertion Error
b) 10 8
c) No output
d) 108

Answer: c
Explanation: The code shown above results in an error if and only if x<y. However, in the above case, since x>y, there is no error. Since there is no print statement, hence there is no output.

13. What will be the output of the following Python code?

```
#generator
def f(x):
    yield x+1
g=f(8)
print(next(g))
```

a) 8
b) 9
c) 7
d) Error

Answer: b
Explanation: The code shown above returns the value of the expression x+1, since we have used to keyword yield. The value of x is 8. Hence the output of the code is 9.

14. What will be the output of the following Python code?

```
def f(x):
```

```
    yield x+1
    print("test")
    yield x+2
g=f(9)
```

a) Error
b) test
c) test
10
12
d) No output

Answer: d
Explanation: The code shown above will not yield any output. This is because when we try to yield 9, and there is no next(g), the iteration stops. Hence there is no output.

15. What will be the output of the following Python code?

```
def f(x):
    yield x+1
    print("test")
    yield x+2
g=f(10)
print(next(g))
print(next(g))
```

a) No output

b)
11
test
12

c)
11
test

d) 11

Answer: b
Explanation: The code shown above results in the output:
11
test
12
This is because we have used next(g) twice. Had we not used next, there would be no output.

16. What will be the output of the following Python code?

```
def a():
    try:
        f(x, 4)
    finally:
        print('after f')
    print('after f?')
a()
```

a) No output
b) after f?
c) error
d) after f

Answer: c
Explanation: This code shown above will result in an error simply because ‘f’ is not defined. ‘try’ and ‘finally’ are keywords used in exception handling.

17. What will be the output of the following Python code?

```
def f(x):
    for i in range(5):
        yield i
g=f(8)
print(list(g))
```

a) [0, 1, 2, 3, 4]
b) [1, 2, 3, 4, 5, 6, 7, 8]
c) [1, 2, 3, 4, 5]
d) [0, 1, 2, 3, 4, 5, 6, 7]

Answer: a
Explanation: The output of the code shown above is a list containing whole numbers in the range (5). Hence the output of this code is: [0, 1, 2, 3, 4].

18. The error displayed in the following Python code is?

```
import itertools
l1=(1, 2, 3)
l2=[4, 5, 6]
l=itertools.chain(l1, l2)
print(next(l1))
```

a) ‘list’ object is not iterator
b) ‘tuple’ object is not iterator
c) ‘list’ object is iterator
d) ‘tuple’ object is iterator

Answer: b
Explanation: The error raised in the code shown above is that: ‘tuple’ object is not iterator. Had we given l2 as argument to next, the error would have been: ‘list’ object is not iterator.

19. Which of the following is not an exception handling keyword in Python?
a) try
b) except
c) accept
d) finally

Answer: c
Explanation: The keywords ‘try’, ‘except’ and ‘finally’ are exception handling keywords in python whereas the word ‘accept’ is not a keyword at all.

20. What will be the output of the following Python code?

```
g = (i for i in range(5))
type(g)
```

a) class <’loop’>
b) class <‘iteration’>
c) class <’range’>
d) class <’generator’>

Answer: d
Explanation: Another way of creating a generator is to use parenthesis. Hence the output of the code shown above is: class<’generator’>.

21. What happens if the file is not found in the following Python code?

```
a=False
while not a:
    try:
        f_n = input("Enter file name")
        i_f = open(f_n, 'r')
    except:
        print("Input file not found")
```

a) No error
b) Assertion error
c) Input output error
d) Name error

Answer: a
Explanation: In the code shown above, if the input file in not found, then the statement: “Input file not found” is printed on the screen. The user is then prompted to reenter the file name. Error is not thrown.

22. What will be the output of the following Python code?

```
lst = [1, 2, 3]
lst[3]
```

a) NameError
b) ValueError
c) IndexError
d) TypeError

Answer: c
Explanation: The snippet of code shown above throws an index error. This is because the index of the list given in the code, that is, 3 is out of range. The maximum index of this list is 2.

23. What will be the output of the following Python code?

```
t[5]
```

a) IndexError
b) NameError
c) TypeError
d) ValeError

Answer: b
Explanation: The expression shown above results in a name error. This is because the name ‘t’ is not defined.

24. What will be the output of the following Python code, if the time module has already been imported?

```
4 + '3'
```

a) NameError
b) IndexError
c) ValueError
d) TypeError

Answer: d
Explanation: The line of code shown above will result in a type error. This is because the operand ‘+’ is not supported when we combine the data types ‘int’ and ‘str’. Sine this is exactly what we have done in the code shown above, a type error is thrown.

25. What will be the output of the following Python code?

```
int('65.43')
```

a) ImportError
b) ValueError

c) TypeError
d) NameError

Answer: b
Explanation: The snippet of code shown above results in a value error. This is because there is an invalid literal for int() with base 10: ’65.43’.

26. Compare the following two Python codes shown below and state the output if the input entered in each case is -6?

```
CODE 1
import math
num=int(input("Enter a number of whose factorial you want to find"))
print(math.factorial(num))

CODE 2
num=int(input("Enter a number of whose factorial you want to find"))
print(math.factorial(num))
```

a) ValueError, NameError
b) AttributeError, ValueError
c) NameError, TypeError
d) TypeError, ValueError

Answer: a
Explanation: The first code results in a ValueError. This is because when we enter the input as -6, we are trying to find the factorial of a negative number, which is not possible. The second code results in a NameError. This is because we have not imported the math module. Hence the name ‘math’ is undefined.

27. What will be the output of the following Python code?

```
def getMonth(m):
    if m<1 or m>12:
        raise ValueError("Invalid")
    print(m)
getMonth(6)
```

a) ValueError
b) Invalid
c) 6
d) ValueError(“Invalid”)

Answer: c
Explanation: In the code shown above, since the value passed as an argument to the function is

between 1 and 12 (both included), hence the output is the value itself, that is 6. If the value had been above 12 and less than 1, a ValueError would have been thrown.

28. What will be the output of the following Python code if the input entered is 6?

```
valid = False
while not valid:
    try:
        n=int(input("Enter a number"))
        while n%2==0:
            print("Bye")
        valid = True
    except ValueError:
        print("Invalid")
```

a) Bye (printed once)
b) No output
c) Invalid (printed once)
d) Bye (printed infinite number of times)

Answer: d
Explanation: The code shown above results in the word “Bye” being printed infinite number of times. This is because an even number has been given as input. If an odd number had been given as input, then there would have been no output.

29. Identify the type of error in the following Python codes?

```
Print(“Good Morning”)
print(“Good night)
```

a) Syntax, Syntax
b) Semantic, Syntax
c) Semantic, Semantic
d) Syntax, Semantic

Answer: b
Explanation: The first code shows an error detected during execution. This might occur occasionally. The second line of code represents a syntax error. When there is deviation from the rules of a language, a syntax error is thrown.

30. Which of the following statements is true?
a) The standard exceptions are automatically imported into Python programs
b) All raised standard exceptions must be handled in Python
c) When there is a deviation from the rules of a programming language, a semantic error is thrown
d) If any exception is thrown in try block, else block is executed

Answer: a
Explanation: When any exception is thrown in try block, except block is executed. If exception in not thrown in try block, else block is executed. When there is a deviation from the rules of a programming language, a syntax error is thrown. The only true statement above is: The standard exceptions are automatically imported into Python programs.

31. Which of the following is not a standard exception in Python?
a) NameError
b) IOError
c) AssignmentError
d) ValueError

Answer: c
Explanation: NameError, IOError and ValueError are standard exceptions in Python whereas Assignment error is not a standard exception in Python.

32. Syntax errors are also known as parsing errors.
a) True
b) False

Answer: a
Explanation: Syntax errors are known as parsing errors. Syntax errors are raised when there is a deviation from the rules of a language. Hence the statement is true.

33. An exception is _____
a) an object
b) a special function
c) a standard module
d) a module

Answer: a
Explanation: An exception is an object that is raised by a function signaling that an unexpected situation has occurred, that the function itself cannot handle.

34. _____ exceptions are raised as a result of an error in opening a particular file.
a) ValueError
b) TypeError
c) ImportError
d) IOError

Answer: d
Explanation: IOError exceptions are raised as a result of an error in opening or closing **a particular file.**

35. Which of the following blocks will be executed whether an exception is thrown or not?
a) except
b) else
c) finally

d) assert

Answer: c
Explanation: The statements in the finally block will always be executed, whether an exception is thrown or not. This clause is used to close the resources used in a code.

36. What are the Iterators and generators in python?
Iterators are primarily used to iterate through other objects or convert them to iterators using the iter() method. Generators are commonly used in loops to create an iterator by returning all the data without impacting the loop's iteration. Iterator makes use of the iter() and next() methods. The generator uses the yield keyword.

37.What is the difference between iterable and iterator in Python?
Iterable is a type of object that can be iterated over. Iterators are used to iterate through iterable objects using the __next__() function. Iterators have a __next__() function that returns the object's next item. It is important to note that every iterator is also iterable, but not every iterable is an iterator.

38.What is the purpose of the yield statement?
A yield statement is similar to a return statement in its simplest form, except that instead of halting execution of the function and returning, yield instead delivers a value to the code looping over the generator and stops execution of the generator function.

39.**Describe generators in Python and give a brief example of how to use them**
A generator in Python **is an easy way of creating an iterable object** without having to implement the whole iteration protocol. It is just a simple function that contains a yield statement instead of the return one.

When the interpreter finds a yield statement, it returns a value like it would with a return one but instead of terminating the functions it puts it in pause, keeping the values and the state of all the variables and objects contained.

For example, let's say that we want to create a function to get the Fibonacci sequence like in the former example.

With a generator, we just need to create a function with a loop that calculates these numbers and after each number is found returns it by using the yield keyword:

```
def fibonacci():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a+b
```

```
if __name__ == '__main__':
    # Create a generator of fibonacci numbers smaller than 1 million
    fibonacci_generator = fibonacci()

    # print out all the sequence until CTRL-C is pressed
    try:
        for fibonacci_number in fibonacci_generator:
            print(fibonacci_number)
    except KeyboardInterrupt:
        print("terminated")
```

As you can see, creating the sequence with this method is really trivial since you just need to create a function that calculates all the sequence and returns the numbers to the caller with the yield keyword.

40. What are iterators in Python and how can you make an object iterable?

An iterator is **an object that allows you to traverse a container object**, iterating all the values it contains. If a container object can be iterated, it is said to be **iterable**.

In Python, there are lots of iterable objects, for example, **lists** and **strings**.

Iterating a list lets you iterate over every single value contained in a list while iterating a string lets you iterate over every char of the string, like in the following example:

```
input_string = "Hello Pythonista!"

for char in input_string:
    print(char)
```

However, you can easily build your custom iterable object by yourself just by implementing the **iterator protocol**. This protocol consists of two methods:

```
def __iter__(self):

def __next__(self):
```

In the .__iter__() method you have to return the object that can be iterated and in the .__next__() method you have to return each time a different value of your collection.

For a practical use case, look at this example that implements an iterable object that can return all the Fibonacci sequence. That is a sequence that starts with the 0 and 1 numbers and where each other number of the sequence is found by adding up the two numbers before it.

```
class fibonacci:

    def __init__(self, max=1000000):
        self.a, self.b = 0, 1
        self.max = max

    def __iter__(self):
        # Return the iterable object (self)
        return self

    def __next__(self):
        # When we need to stop the iteration we just need to raise
        # a StopIteration exception
        if self.a > self.max:
            raise StopIteration

        # save the value that has to be returned
        value_to_be_returned = self.a

        # calculate the next values of the sequence
        self.a, self.b = self.b, self.a + self.b

        return value_to_be_returned

my_fibonacci_iterable = fibonacci(1000)

for i in my_fibonacci_iterable:
```

```
    print(i)
```

As you can see, to create this iterable Fibonacci object in the .__iter__() method, we have just returned the object itself, and in the .__next__() method we have calculated the number to be returned at each iteration.

Week 9: Data Analysis with Pandas

1) Define the Pandas/Python pandas?

Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. It can be used for data analysis in Python and developed by Wes McKinney in 2008. It can perform five significant steps that are required for processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyze.

2) Mention the different types of Data Structures in Pandas?

Pandas provide two data structures, which are supported by the pandas library, Series, and DataFrames. Both of these data structures are built on top of the NumPy.

A Series is a one-dimensional data structure in pandas, whereas the DataFrame is the two-dimensional data structure in pandas.

65.1M

1.1K

Hello Java Program for Beginners

3) Define Series in Pandas?

A Series is defined as a one-dimensional array that is capable of storing various data types. The row labels of series are called the index. By using a 'series' method, we can easily convert the list, tuple, and dictionary into series. A Series cannot contain multiple columns.

4) How can we calculate the standard deviation from the Series?

The Pandas std() is defined as a function for calculating the standard deviation of the given set of numbers, DataFrame, column, and rows.

Series.std(axis=None, skipna=None, level=None, ddof=1, numeric_only=None, **kwargs)

5) Define DataFrame in Pandas?

A DataFrame is a widely used data structure of pandas and works with a two-dimensional array with labeled axes (rows and columns) DataFrame is defined as a standard way to store data and has two different indexes, i.e., row index and column index. It consists of the following properties:

The columns can be heterogeneous types like int and bool.

It can be seen as a dictionary of Series structure where both the rows and columns are indexed. It is denoted as "columns" in the case of columns and "index" in case of rows.

6) What are the significant features of the pandas Library?

The key features of the panda's library are as follows:

Memory Efficient

Data Alignment

Reshaping

Merge and join

Time Series

7) Explain Reindexing in pandas?

Reindexing is used to conform DataFrame to a new index with optional filling logic. It places NA/NaN in that location where the values are not present in the previous index. It returns a new object unless the new index is produced as equivalent to the current one, and the value of copy becomes False. It is used to change the index of the rows and columns of the DataFrame.

8) What is the name of Pandas library tools used to create a scatter plot matrix?

Scatter_matrix

9) Define the different ways a DataFrame can be created in pandas?

We can create a DataFrame using following ways:

Lists

Dict of ndarrays

Example-1: Create a DataFrame using List:

```
import pandas as pd

# a list of strings

a = ['Python', 'Pandas']

# Calling DataFrame constructor on list

info = pd.DataFrame(a)

print(info)
```

Output:

```
          0

0   Python

1   Pandas
```

Example-2: Create a DataFrame from dict of ndarrays:

```
import pandas as pd

info = {'ID' :[101, 102, 103],'Department' :['B.Sc','B.Tech','M.Tech',]}

info = pd.DataFrame(info)

print (info)
```

Output:

```
      ID      Department

0     101        B.Sc

1     102        B.Tech

2     103        M.Tech
```

10) Explain Categorical data in Pandas?

A Categorical data is defined as a Pandas data type that corresponds to a categorical variable in statistics. A categorical variable is generally used to take a limited and usually fixed number of possible values. Examples: gender, country affiliation, blood type, social class, observation time, or rating via Likert scales. All values of categorical data are either in categories or np.nan.

This data type is useful in the following cases:

It is useful for a string variable that consists of only a few different values. If we want to save some memory, we can convert a string variable to a categorical variable.

It is useful for the lexical order of a variable that is not the same as the logical order (?one?, ?two?, ?three?) By converting into a categorical and specify an order on the categories, sorting and min/max is responsible for using the logical order instead of the lexical order.

It is useful as a signal to other Python libraries because this column should be treated as a categorical variable.

11) How will you create a series from dict in Pandas?

A Series is defined as a one-dimensional array that is capable of storing various data types.

We can create a Pandas Series from Dictionary:

Create a Series from dict:

We can also create a Series from dict. If the dictionary object is being passed as an input and the index is not specified, then the dictionary keys are taken in a sorted order to construct the index.

If index is passed, then values correspond to a particular label in the index will be extracted from the dictionary.

```
import pandas as pd

import numpy as np

info = {'x' : 0., 'y' : 1., 'z' : 2.}

a = pd.Series(info)

print (a)
```

Output:

x 0.0

y 1.0

z 2.0

dtype: float64

12) How can we create a copy of the series in Pandas?

We can create the copy of series by using the following syntax:

pandas.Series.copy
Series.copy(deep=True)

The above statements make a deep copy that includes a copy of the data and the indices. If we set the value of deep to False, it will neither copy the indices nor the data.

13) How will you create an empty DataFrame in Pandas?

A DataFrame is a widely used data structure of pandas and works with a two-dimensional array with labeled axes (rows and columns) It is defined as a standard way to store data and has two different indexes, i.e., row index and column index.

Create an empty DataFrame:

The below code shows how to create an empty DataFrame in Pandas:

importing the pandas library

import pandas as pd

info = pd.DataFrame()

print (info)

Output:

Empty DataFrame

Columns: []

Index: []

14) How will you add a column to a pandas DataFrame?

We can add any new column to an existing DataFrame. The below code demonstrates how to add any new column to an existing DataFrame:

```
# importing the pandas library

import pandas as pd

info = {'one' : pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e']),

        'two' : pd.Series([1, 2, 3, 4, 5, 6], index=['a', 'b', 'c', 'd', 'e', 'f'])}


info = pd.DataFrame(info)


# Add a new column to an existing DataFrame object

print ("Add new column by passing series")

info['three']=pd.Series([20,40,60],index=['a','b','c'])

print (info)

print ("Add new column using existing DataFrame columns")

info['four']=info['one']+info['three']

print (info)
```

Output:

```
Add new column by passing series

   one  two  three

a  1.0   1   20.0

b  2.0   2   40.0
```

c	3.0	3	60.0
d	4.0	4	NaN
e	5.0	5	NaN
f	NaN	6	NaN

Add new column using existing DataFrame columns

	one	two	three	four
a	1.0	1	20.0	21.0
b	2.0	2	40.0	42.0
c	3.0	3	60.0	63.0
d	4.0	4	NaN	NaN
e	5.0	5	NaN	NaN
f	NaN	6	NaN	NaN

15) How to add an Index, row, or column to a Pandas DataFrame?

Adding an Index to a DataFrame

Pandas allow adding the inputs to the index argument if you create a DataFrame. It will make sure that you have the desired index. If you don?t specify inputs, the DataFrame contains, by default, a numerically valued index that starts with 0 and ends on the last row of the DataFrame.

Adding Rows to a DataFrame

We can use .loc, iloc, and ix to insert the rows in the DataFrame.

The loc basically works for the labels of our index. It can be understood as if we insert in loc[4], which means we are looking for that values of DataFrame that have an index labeled 4.

The iloc basically works for the positions in the index. It can be understood as if we insert in iloc[4], which means we are looking for the values of DataFrame that are present at index '4`.

The ix is a complex case because if the index is integer-based, we pass a label to ix. The ix[4] means that we are looking in the DataFrame for those values that have an index labeled 4. However, if the index is not only integer-based, ix will deal with the positions as iloc.

Adding Columns to a DataFrame

If we want to add the column to the DataFrame, we can easily follow the same procedure as adding an index to the DataFrame by using loc or iloc.

16) How to Delete Indices, Rows or Columns From a Pandas Data Frame?

Deleting an Index from Your DataFrame

If you want to remove the index from the DataFrame, you should have to do the following:

Reset the index of DataFrame.

Executing del df.index.name to remove the index name.

Remove duplicate index values by resetting the index and drop the duplicate values from the index column.

Remove an index with a row.

Deleting a Column from Your DataFrame

You can use the drop() method for deleting a column from the DataFrame.

The axis argument that is passed to the drop() method is either 0 if it indicates the rows and 1 if it drops the columns.

You can pass the argument inplace and set it to True to delete the column without reassign the DataFrame.

You can also delete the duplicate values from the column by using the drop_duplicates() method.

Removing a Row from Your DataFrame

By using df.drop_duplicates(), we can remove duplicate rows from the DataFrame.

You can use the drop() method to specify the index of the rows that we want to remove from the DataFrame.

17.How to Rename the Index or Columns of a Pandas DataFrame?

You can use the .rename method to give different values to the columns or the index values of DataFrame.

18. How to iterate over a Pandas DataFrame?

You can iterate over the rows of the DataFrame by using for loop in combination with an iterrows() call on the DataFrame.

19.How to get the items of series A not present in series B?

We can remove items present in p2 from p1 using isin() method.

```
import pandas as pd

p1 = pd.Series([2, 4, 6, 8, 10])

p2 = pd.Series([8, 10, 12, 14, 16])

p1[~p1.isin(p2)]
```

Solution

```
0  2

1  4

2  6

dtype: int64
```

20.How to get the items not common to both series A and series B?

We get all the items of p1 and p2 not common to both using below example:

```
import pandas as pd
```

```
import numpy as np

p1 = pd.Series([2, 4, 6, 8, 10])

p2 = pd.Series([8, 10, 12, 14, 16])

p1[~p1.isin(p2)]

p_u = pd.Series(np.union1d(p1, p2))  # union

p_i = pd.Series(np.intersect1d(p1, p2))  # intersect

p_u[~p_u.isin(p_i)]
```

Output:

```
0     2

1     4

2     6

5    12

6    14

7    16

dtype: int64
```

21.How to get the minimum, 25th percentile, median, 75th, and max of a numeric series?

We can compute the minimum, 25th percentile, median, 75th, and maximum of p as below example:

```
import pandas as pd

import numpy as np

p = pd.Series(np.random.normal(14, 6, 22))

state = np.random.RandomState(120)
```

```
p = pd.Series(state.normal(14, 6, 22))

np.percentile(p, q=[0, 25, 50, 75, 100])
```

Output:

```
array([ 4.61498692, 12.15572753, 14.67780756, 17.58054104, 33.24975515])
```

22. How to get frequency counts of unique items of a series?

We can calculate the frequency counts of each unique value p as below example:

```
import pandas as pd

import numpy as np

p= pd.Series(np.take(list('pqrstu'), np.random.randint(6, size=17)))

p = pd.Series(np.take(list('pqrstu'), np.random.randint(6, size=17)))

p.value_counts()
```

Output:

```
s    4

r    4

q    3

p    3

u    3
```

23.How to convert a numpy array to a dataframe of given shape?

We can reshape the series p into a dataframe with 6 rows and 2 columns as below example:

```
import pandas as pd

import numpy as np

p = pd.Series(np.random.randint(1, 7, 35))
```

```
# Input

p = pd.Series(np.random.randint(1, 7, 35))

info = pd.DataFrame(p.values.reshape(7,5))

print(info)
```

Output:

```
0  1  2  3  4

0  3  2  5  5  1

1  3  2  5  5  5

2  1  3  1  2  6

3  1  1  1  2  2

4  3  5  3  3  3

5  2  5  3  6  4

6  3  6  6  6  5
```

24.How can we convert a Series to DataFrame?

The Pandas Series.to_frame() function is used to convert the series object to the DataFrame.

Series.to_frame(name=None)

name: Refers to the object. Its Default value is None. If it has one value, the passed name will be substituted for the series name.

```
s = pd.Series(["a", "b", "c"],

name="vals")

s.to_frame()
```

Output:

	vals
0	a
1	b
2	c

25. What is Pandas NumPy array?

Numerical Python (Numpy) is defined as a Python package used for performing the various numerical computations and processing of the multidimensional and single-dimensional array elements. The calculations using Numpy arrays are faster than the normal Python array.

26.How can we convert DataFrame into a NumPy array?

For performing some high-level mathematical functions, we can convert Pandas DataFrame to numpy arrays. It uses the DataFrame.to_numpy() function.

The DataFrame.to_numpy() function is applied to the DataFrame that returns the numpy ndarray.

DataFrame.to_numpy(dtype=None, copy=False)

27) How can we convert DataFrame into an excel file?
We can export the DataFrame to the excel file by using the to_excel() function.
To write a single object to the excel file, we have to specify the target file name. If we want to write to multiple sheets, we need to create an ExcelWriter object with target filename and also need to specify the sheet in the file in which we have to write.

28) How can we sort the DataFrame?

We can efficiently perform sorting in the DataFrame through different kinds:

By label

By Actual value

By label

The DataFrame can be sorted by using the sort_index() method. It can be done by passing the axis arguments and the order of sorting. The sorting is done on row labels in ascending order by default.

By Actual Value

It is another kind through which sorting can be performed in the DataFrame. Like index sorting, sort_values() is a method for sorting the values.

It also provides a feature in which we can specify the column name of the DataFrame with which values are to be sorted. It is done by passing the 'by' argument.

29) What is Time Series in Pandas?

The Time series data is defined as an essential source for information that provides a strategy that is used in various businesses. From a conventional finance industry to the education industry, it consists of a lot of details about the time.

Time series forecasting is the machine learning modeling that deals with the Time Series data for predicting future values through Time Series modeling.

30) What is Time Offset?

The offset specifies a set of dates that conform to the DateOffset. We can create the DateOffsets to move the dates forward to valid dates.

31) Define Time Periods?

The Time Periods represent the time span, e.g., days, years, quarter or month, etc. It is defined as a class that allows us to convert the frequency to the periods.

32) How to convert String to date?

The below code demonstrates how to convert the string to date:

```
fromdatetime import datetime



# Define dates as the strings

dmy_str1 = 'Wednesday, July 14, 2018'
```

```
dmy_str2 = '14/7/17'

dmy_str3 = '14-07-2017'


# Define dates as the datetime objects

dmy_dt1 = datetime.strptime(date_str1, '%A, %B %d, %Y')

dmy_dt2 = datetime.strptime(date_str2, '%m/%d/%y')

dmy_dt3 = datetime.strptime(date_str3, '%m-%d-%Y')


#Print the converted dates

print(dmy_dt1)

print(dmy_dt2)

print(dmy_dt3)
```

Output:

```
2017-07-14 00:00:00

2017-07-14 00:00:00

2018-07-14 00:00:00
```

33) What is Data Aggregation?

The main task of Data Aggregation is to apply some aggregation to one or more columns. It uses the following:

sum: It is used to return the sum of the values for the requested axis.

min: It is used to return a minimum of the values for the requested axis.

max: It is used to return a maximum values for the requested axis.

34) What is Pandas Index?

Pandas Index is defined as a vital tool that selects particular rows and columns of data from a DataFrame. Its task is to organize the data and to provide fast accessing of data. It can also be called a Subset Selection.

35) Define Multiple Indexing?

Multiple indexing is defined as essential indexing because it deals with data analysis and manipulation, especially for working with higher dimensional data. It also enables us to store and manipulate data with the arbitrary number of dimensions in lower-dimensional data structures like Series and DataFrame.

36) Define ReIndexing?

Reindexing is used to change the index of the rows and columns of the DataFrame. We can reindex the single or multiple rows by using the reindex() method. Default values in the new index are assigned NaN if it is not present in the DataFrame.

DataFrame.reindex(labels=None, index=None, columns=None, axis=None, method=None, copy=True, level=None, fill_value=nan, limit=None, tolerance=None)

37) How to Set the index?

We can set the index column while making a data frame. But sometimes, a data frame is made from two or more data frames, and then the index can be changed using this method.

38) How to Reset the index?

The Reset index of the DataFrame is used to reset the index by using the 'reset_index' command. If the DataFrame has a MultiIndex, this method can remove one or more levels.

39) Describe Data Operations in Pandas?

In Pandas, there are different useful data operations for DataFrame, which are as follows:

Row and column selection

We can select any row and column of the DataFrame by passing the name of the rows and columns. When you select it from the DataFrame, it becomes one-dimensional and considered as Series.

Filter Data

We can filter the data by providing some of the boolean expressions in DataFrame.

Null values

A Null value occurs when no data is provided to the items. The various columns may contain no values, which are usually represented as NaN.

40) Define GroupBy in Pandas?

In Pandas, groupby() function allows us to rearrange the data by utilizing them on real-world data sets. Its primary task is to split the data into various groups. These groups are categorized based on some criteria. The objects can be divided from any of their axes.

DataFrame.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, **kwargs)

\

WEEK 10 : Numeric and Scientific Computing using Numpy

1. What is Numpy used for?

Numpy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, and matrices.

2. How to import numpy in python?

import numpy as np

where np is giving short name to numpy , it is known as aliasing.

3. Write a code to print null vector and ones of size 10.

np.zeros(10)

np.ones (10)

4. Create 1D array using numpy and convert it to 2D array

a = np.array([1,2,3,4,5,6,7,8])

a.reshape(4,2)

5. How will you print the numbers from 1 – 10 with step 2 using numpy

It can be done by **arrange** function

np.arange(1,11,2)

6. When will we use ravel function in numpy? Write a code to use ravel.

It is used to change the shape from high dimension to lower dimension.

np.ravel (a)

7. Create the 2D array of x = [[1,2,3] , [4,5,6] , [7,8,9]] and out of that print the given below values

a. 2 **b.** [2 , 3] **c.** [[2 , 3, 5 , 6]]

a. x[0,1]
b. x[0,1:]
c. x[0:2,1:]

8. Create two array a = ([[1,2,3],[4,5,6],[7,8,9]]),

b = ([[10,11,12],[13,14,15],[16,17,18]])

Concatenate these two horizontally as well as vertically.

np.hstack((a,b))

np.hstack((a,b))

9. Write a code to print 3 numbers between 1 to 4 with equidistance .

np.linspace (1 , 4 , 3)

10. Print the random numbers between 1 to 10 which may also include negative number

np.random.randn (1 , 10)

11. Create an array a = [[1,2,3] , [4,5,6] , [7,8,9]] . Print the location of maximum number present in the given array using numpy

np.argmax (a)

12. Write a code to create 3X3 identity matrix using numpy and after creating identity shift the position of 1.

To create a identity matrix -→ **np.eye(3)**

To shift the positision of 1-→ **np.eye (3 , k = 1)**

13. Generate any 10 random numbers an sort them.

```
a = np.random.random( 10 )
a.sort()
```

14. Create any random numbers and prints even numbers from them using where condition

```
a = np.arange( 10 )
np.where( a % 2==0)
```

15. Use numpy to print 4 number between 1 to 10

```
np.random.randint ( 1 , 10 , 4 )
```

16. Create any number and print the log of that number using numpy

```
num = 10
np.log( num )
```

17.Suppose x = np.array [1 ,2 , 3 , 4 , 5] . How can we save this x using numpy

```
np.save ( " name of numpy . npy " , x)
```

18. Write a NumPy program to multiply the values of two given vectors.

```
x = np.array([1, 8, 3, 5])
y= np.random.randint(0, 11, 4)
result = x * y
print("Multiply the values of two said vectors:")
print(result)
```

19. How to check total count of the values stored in variable using numpy

```
np.size ( variable)
```

20. Create an 1D array, after creating change entire array with 100.

```
a = np.array([ 1,2,3,4])
a[:] = 100
```

21. Write a NumPy program to extract all numbers from a given array which are less and greater than a specified number.

```
nums = np.array([[5.54, 3.38, 7.99],
        [3.54, 4.38, 6.99],
        [1.54, 2.39, 9.29]])
n = 5
print("\nElements of the said array greater than",n)
print(nums[nums > n])
n = 6
print("\nElements of the said array less than",n)
print(nums[nums < n])
```

22. Split an array in sub-arrays of (nearly) 3 identical size

```
np.array_split(array, 3)
```

23. What is the procedure to find the indices of an array on NumPy where some condition is true?

```
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
a > 3
```

24. Create a 3X3 matrix of a constant value

```
np.full((3,3),55)
```

25. **How can you identify the datatype of a given NumPy array?**

```
a.ndim
```

Week 11: Graphical User Interfaces with Tkinter

1 . What is python tkinter?

The tkinter package (“Tk interface”) is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, including macOS, as well as on Windows systems.

Tkinter supports a range of Tcl/Tk versions, built either with or without thread support. The official Python binary release bundles Tcl/Tk 8.6 threaded. See the source code for the _tkinter module for more information about supported versions.

2 .What is tkinter used for in Python?

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter app :

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the Python code. N

ote that the name of the module in Python 2.x is ‘Tkinter’ and in Python 3.x it is ‘tkinter’.

3 .What is root tkinter?

App and Root are two different objects. root is the root window into which all other widgets go. It is an instance of the class Tk, and every tkinter application must have exactly one instance of this class. app is an instance of the class App, which is a subclass of Frame. A frame is typically used

as a container for other widgets, in this case the widgets that make up the app (except for the root window). This frame is packed inside the root window. app and root are two completely different things. root is a container for app.

4 .What is Python Tkinter pack() method?

The pack() widget is used to organize widget in the block. The positions widgets added to the python application using the pack() method can be controlled by using the various options specified in the method call.

However, the controls are less and widgets are generally added in the less organized manner.

Syntax : `widget.pack(options)`

5 .What is Python Tkinter grid() method?

The grid() geometry manager organizes the widgets in the tabular form. We can specify the rows and columns as the options in the method call. We can also specify the column span (width) or rowspan(height) of a widget.

This is a more organized way to place the widgets to the python application. The syntax to use the grid() is given below.

Syntax : `widget.grid(options)`

6 .What is Python Tkinter place() method?

The place() geometry manager organizes the widgets to the specific x and y coordinates.

Syntax : `widget.place(options)`

7 .How many Python Tkinter pack() methods?

expand : When set to true, widget expands to fill any space not otherwise used in widget's parent.

fill : Determines whether widget fills any extra space allocated to it by the packer, or keeps its own minimal dimensions: NONE (default), X (fill only horizontally), Y (fill only vertically), or BOTH (fill both horizontally and vertically).

side : Determines which side of the parent widget packs against: TOP (default), BOTTOM, LEFT, or RIGHT.

8 .How many Python Tkinter grid() methods?

Column : The column number in which the widget is to be placed. The leftmost column is represented by 0.

Columnspan : The width of the widget. It represents the number of columns up to which, the column is expanded.

ipadx, ipady : It represents the number of pixels to pad the widget inside the widget's border.

padx, pady : It represents the number of pixels to pad the widget outside the widget's border.
row : The row number in which the widget is to be placed. The topmost row is represented by 0.

rowspan : The height of the widget, i.e. the number of the row up to which the widget is expanded.
Sticky : If the cell is larger than a widget, then sticky is used to specify the position of the widget inside the cell. It may be the concatenation of the sticky letters representing the position of the widget. It may be N, E, W, S, NE, NW, NS, EW, ES.

9 .How many Python Tkinter place() methods?

Anchor : It represents the exact position of the widget within the container. The default value (direction) is NW (the upper left corner)

bordermode : The default value of the border type is INSIDE that refers to ignore the parent's inside the border. The other option is OUTSIDE.

height, width : It refers to the height and width in pixels.

relheight, relwidth : It is represented as the float between 0.0 and 1.0 indicating the fraction of the parent's height and width.

relx, rely : It is represented as the float between 0.0 and 1.0 that is the offset in the horizontal and vertical direction.

x, y : It refers to the horizontal and vertical offset in the pixels.

10 .How to stop copy, paste, and backspace in text widget in tkinter?

The Text widget accepts multiline user input, where you can type text and perform operations like copying, pasting, and deleting. There are certain ways to disable the shortcuts for various operations on a Text widget.

In order to disable copy, paste and backspace in a Text widget, you've to bind the event with an event handler and return break using lambda keyword in python.

11.How to get a new API response in a Tkinter textbox?

APIs are extremely useful in implementing a service or feature in an application. APIs help to establish the connection between the server and a client, so whenever a client sends a request using one of the API methods to the server, the server responds with a status code (201 as a successful response) to the client.

You can make a request to any API you want using one of the methods (GET, POST, PUT or DELETE). However, if you want to create an application where you need a request to the server using one of the publicly available API (for example, Cat Facts API), then you can use the requests module in the Python library.

In the following application, we will create a textbox which will display the response (text) retrieved from the server using one of the Cat Facts API. You will also need to make sure that you have already installed the requests module in your environment. To install requests module, you can use the following command,

```
pip install requests
```

12.How to get the index of selected option in Tkinter Combobox?

If you want to create a dropdown list of items and enable the items of list to be selected by the user, then you can use the Combobox widget. The Combobox widget allows you to create a dropdown list in which the list of items can be selected instantly. However, if you want to get the index of selected items in the combobox widget, then you can use the get() method. The get() method returns an integer of the selected item known as the index of the item.

Example : Let's take an example to see how it works. In this example, we have created a list of days of weeks in a dropdown list and whenever the user selects a day from the dropdown list, it will print and display the index of selected item on a Label widget. To print the index, we can concatenate the string by typecasting the given index into string.

13. How to get an Entry box within a Messagebox in Tkinter?

There are various methods and built-in functions available with the messagebox library in tkinter. Let's assume you want to display a messagebox and take some input from the user in an Entry widget. In this case, you can use the askstring library from simpledialog. The askstring library creates a window that takes two arguments, the title of the window, and the input title before the Entry widget.

14.How to use a StringVar object in an Entry widget in Tkinter?

A StringVar object in Tkinter can help manage the value of a widget such as an Entry widget or a Label widget. You can assign a StringVar object to the textvariable of a widget.

For example :

```
data = ['Car', 'Bus', 'Truck', 'Bike', 'Airplane']
var = StringVar(win)
my_spinbox = Spinbox(win, values=data, textvariable=var)
```

Here, we created a list of strings followed by a StringVar object "var". Next, we assigned var to the textvariable of a Spinbox widget. To get the current value of the Spinbox, you can use var.get().

15. How to get the current date to display in a tkinter window?

To display the current date in tkinter window, we will use the datetime library.

```
date = dt.datetime.now()
```

Steps :

* Import the required libraries and create an instance of tkinter frame.
* Set the size of the frame using geometry method.
* Call datetime.now() and store the value in a variable "date".
* Next, create a label to display the date. In the text parameter of the label, pass the date value and format the data as text=f"{date:%A, %B %d, %Y}".
  * %A – Day of the week, full name
  * %B – Full month name
  * %d – Day of the month
  * %Y – Year with century as a decimal number

* Finally, run the mainloop of the application window.

16.How to Create an automatically maximized tkinter window

There are two different ways in which we can get an automatically maximized window in Tkinter.

* We can use the state() method of Tkinter and invoke it with the attribute "zoomed".
  root.state("zoomed")
* The second approach is to use the attributes method of Tkinter with the parameter "-fullscreen" and set it to True.

By default, Tkinter creates a window of a predefined size. The dimensions of the window can be customized using the geometry method. For example,

```
root.geometry("700 x 350")
```

17.How to use a StringVar object in an Entry widget in Tkinter?

A StringVar object in Tkinter can help manage the value of a widget such as an Entry widget or a Label widget. You can assign a StringVar object to the textvariable of a widget.

For example :

```
data = ['Car', 'Bus', 'Truck', 'Bike', 'Airplane']
var = StringVar(win)
my_spinbox = Spinbox(win, values=data, textvariable=var)
```

Here, we created a list of strings followed by a StringVar object "var". Next, we assigned var to the textvariable of a Spinbox widget. To get the current value of the Spinbox, you can use var.get().

18.How to reconfigure Tkinter canvas items?

Using the Canvas widget, we can create text, images, graphics, and visual content to add to the Canvas widget. If you need to configure the Canvas item dynamically, then tkinter provides itemconfig(**options) method. You can use this method to configure the properties and attributes of the Canvas items. For example, if we create a line inside the Canvas widget, we can configure its color or width using itemconfig() method.

19.How to create a Tkinter error message box?

The Tkinter library has many built-in functions and methods which can be used to implement the functional part of an application. We can use messagebox module in Tkinter to create various popup dialog boxes. The messagebox property has different types of built-in popup windows that the users can use in their applications.

If you need to display the error messagebox in your application, you can use showerror("Title", "Error Message") method. This method can be invoked with the messagebox itself.

20. How to make a new folder using askdirectory dialog in Tkinter?

To make a new folder using askdirectory dialog in Tkinter, we can take the following steps :

* Import the required modules. filedialog module is required for askdirectory method. os module is required for makedirs method.
* Create an instance of tkinter frame.
* Set the size of the frame using win.geometry method.
* Define a user-defined method "create_subfolder". Inside the method, call filedialog.askdirectory to select a folder and save the path in a variable, source_path.
* We can use askdirectory method of filedialog to open a directory. Save the path of the selected directory in a 'path' variable.
* Then, use os.path.join and makedirs to create a sub-folder inside the parent directory.
* Create a button to call the create_subfolder method.* Next, add the PIL image to the Canvas using canvas.create_image().

* Finally, run the mainloop of the application window

Week 12: Interacting with Databases

1. Describe Database?

A Database is essentially a way to store and retrieve data. Typically, there is some form of query language used with the database to help select the information to retrieve such as SQL or Structured Query Language.In a Relational Database the data is held in tables, where the columns define the properties or attributes of the data and each row defines the actual values being held. There are many different relational database systems which are used for a wide variety of purposes including:

Oracle, MySQL, Microsoft SQL Server, PostgreSQL and SQLite.

2. What is Sqlite3 Database?

SQLite is a python library that implements a self-contained, server-less, zero-configuration, transactional SQL database engine.

Zero-configured database means, it does not need any configuration like other databases. In SQLite data is directly stored in a cross-platform file.SQLite engine is not a standalone process like other databases.The SQLite engine is fully self-contained, thereby eliminating the need for dependencies

3.How to work with Sqlite3 Database?

In order to interact with SQLite3 database, the following 5 steps should be followed:

- Connect to Database
- Create a cursor object
- Write SQL Query
- Commit changes
- Close database connection

4.How to connect to the database?

To use SQLite3 in Python, we have to import the sqlite3 module and then create a connection object which will connect us to the database and will let us execute the SQL statements. • You can a connection object using the connect() function:

Code:

```
import sqlite3
con = sqlite3.connect('database_name.db')
```

To execute SQLite statements in Python, you need a cursor object. You can create it using the cursor() method.

5. How to create a cursor Object?

The SQLite3 cursor is a method of the connection object. To execute the SQLite3 statements, you should establish a connection at first and then create an object of the cursor using the connection object as follows:

Code:

```
con =
sqlite3.connect('mydatabase.db')
cursorObj = con.cursor()
```

- Now we can use the cursor object to call the execute() method to execute any SQL queries.

- When you create a connection with SQLite, that will create a database file automatically if it doesn't already exist.

6. How to create a table?

After establishing connection and creation of cursor object we can write a table creation query inside the execute method.

Code:

```
import sqlite3

def create_table():
    conn =
    sqlite3.connect("student.db")
```

```
    cur = conn.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS student(rollno
    INTEGER, name TEXT,gpa REAL)")
    conn.commit()
    conn.close()
create_table()      # function calling
```

7. **How to insert the data into the table?**

   To insert data in a table, we use the INSERT INTO statement. Consider the following line of code:

```
cur.execute("INSERT INTO student values(?,?,?)",(r,n,g))
```

   - The '?' symbol is a place-holder for values which have to be substituted in the query
   - A direct method of inserting values into a table is as follows: cur.execute("INSERT INTO student values(101,'Ramesh',9.0)")
   - The above statement is a static way of inserting values into tables

   Code: def insert(r,n,g):

```
        conn =
        sqlite3.connect("student.d")
        cur = conn.cursor()
        cur.execute("INSERT INTO student
        values(?,?,?)",(r,n,g)) conn.commit()
        conn.close()
insert(101,'Ramesh',8.2)      #
function call
```

8. **How to get the data present in the table ?**

   - You can use the SELECT statement to select data from a particular table.
   - If you want to select all the columns of the data from a table, you can use the asterisk (*).

- The syntax for this will be as follows: select * from table name
- In SQLite3, the SELECT statement is executed in the execute method of the cursor object
- For example, select all the columns of the student table, run the following code: cursorObj.execute('SELECT * FROM student ')
- If you want to select a few columns from a table, then specify the columns like the following: SELECT column1, column2 from table_name ;

Code: def view():

```
    conn =
    sqlite3.connect("student.db") cur
    = conn.cursor()
    cur.execute("SELECT * FROM
    student") rows = cur.fetchall()
    conn.cl
ose() view() #
function call
```

- Fetch all data: The select statement selects the required data from the database table, and if you want to fetch the selected data, the fetchall() method of the cursor object is used.
- The fetchall() method of the cursor object to store the values into a variable. It returns a list object
- After that, we will loop through the variable and access all values, or we can simply print the list
- In the following example, we define a view function which retrieves all the values from the student table

9. How to filter the require data or rows from the table ?

If you want to fetch specific data from the database , you can use the WHERE clause.

For example, we want to fetch the rollnos of those students whose gpa is greater than n, we use the following view function:

Code: def view(n)

```
conn =
sqlite3.connect("student.db")
cur = conn.cursor()
cur.execute(“SELECT * FROM student where gpa>?
”,(n,)) rows = cur.fetchall()
conn.close()
return rows
view(7)         # function call
```

10.How to delete the rows from the table?

- The DELETE statement is used to delete rows from a table
- The DELETE statement without any condition (a where clause), will delete all the rows in the table
- In order to specify the row to be deleted, the where clause must be specified.

- The syntax of delete statement is:

DELETE FROM table_name WHERE condition.

We should be cautious about mentioning a condition in where clause otherwise It deletes all the rows instead of particular rows.

Code: def delete(r):

```
conn =
sqlite3.connect("student.db")
cur = conn.cursor()
cur.execute("DELETE FROM student where
rollno=?",(r,)) conn.commit()
conn.close()

delete(102)      # function call
```

11.How to update the existing data in the table?

Use the UPDATE statement in the execute() method. Suppose that we want to update the gpa of a student. The following function called update(g,r) is used to update a row. The update statement in the execute() method is:

Code: def update(r,g):

```
        conn = sqlite3.connect("student.db")
        cur = conn.cursor() cur.execute("UPDATE student SET
        gpa=? WHERE rollno=?",(g,r))
        conn.commit()
        conn.close()
update(8,103)          # function calling
```

12 . How to drop the table from the database?

In order to permanently remove a table from the database, the DROP TABLE statement is used.

Its syntax is: DROP TABLE table_name

For example, the following method permanently removes the student table:

Code:

```
  def drop():
        conn =
        sqlite3.connect("student.db")
        cur = conn.cursor()
        cur.execute("DROP TABLE
        student") conn.commit()
        conn.close()
  drop()                  # function call
```