Python Interview Questions

Brush up for Your Next Python Interview with 240+ Solutions on Most Common Challenging Interview Questions

SWATI SAXENA

Python Interview Questions

Brush up for Your Next Python Interview with 240+ Solutions on Most Common Challenging Interview Questions

SWATI SAXENA

Python Interview Questions

Brush up for Your Next Python Interview with 240+ Solutions on Most Common Challenging Interview Questions

Swati Saxena

www.bpbonline.com

FIRST EDITION 2021

ISBN: 978-93-89898-460

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author’s and publisher’s knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

Distributors:

BPB PUBLICATIONS

20, Ansari Road, Darya Ganj

New Delhi-110002

Ph: 23254990/23254991

MICRO MEDIA

Shop No. 5, Mahendra Chambers,

150 DN Rd. Next to Capital Cinema,

V.T. (C.S.T.) Station, MUMBAI-400 001

Ph: 22078296/22078297

DECCAN AGENCIES

4-3-329, Bank Street,

Hyderabad-500195

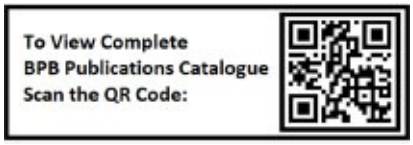
Ph: 24756967/24756400

BPB BOOK CENTRE

376 Old Lajpat Rai Market,

Delhi-110006

Ph: 23861747



Published by Manish Jain for BPB Publications, 20 Ansari Road, Darya Ganj, New Delhi-110002 and Printed by him at Repro India Ltd, Mumbai

www.bpbonline.com

Dedicated to

I would like to dedicate this book to my parents, without whom my life would not have any meaning. Thank you so much for all your unconditional support, love, and education.

Everything I am, I owe it to my parents

About the Author

Swati Saxena is an expert in computer programming and coding. She is an MCA, OCJP (Oracle Certified Java Professional), and ADST, having in-depth knowledge of the subject and very vast experience in developing and training.

She has been awarded by Rajasthan Women Achievement Award and Pratibha Samman for her writing work. She is an achiever of a Certificate of Excellence by MyGov.

She has written “C programming and coding Question Bank with solution”,”Java-A Complete Practical Solution”, “Kotlin at a Glance”, and “Cracking Kotlin Interview” for BPB publications.

Her alumni are well placed in many reputed organizations all over India.

About the Reviewer

Shayank Jain is a software developer, data analyst, and author. He is strongly passionate about coding and architectural design. He has more than seven years of professional experience in developing scalable software solutions for various organizations. He has been programming since the age of 16 and has developed softwares for mobile, web, hardware gaming, and standalone applications. After getting his hands dirty with programming, he found many new ways to debug and deploy the code successfully, with minimal time constraints. After reading and implementation, he found out that many critical concepts can be implemented easily in programming with correct and focused thinking. His research interests include information security, cryptography, analysis, design, and implementation of algorithms. He has extensively worked with python and implemented new ideas on various projects in his free time. He is also active in the computer science and education community. Through this book, he wants to share these methodologies and tricks with the beginners.

Apart from work, Shayank spends his spare time helping, coaching, and mentoring young people in taking up careers in technology.

Acknowledgement

I am thankful to **BPB Publications** who inspired me and asked me to work on this book "*Python Interview Questions*."

I thank Mr Naveen Saxena for his constant support during the preparation of this book.

I am thankful for Prof. Nidhi Saxena, ex-assistant professor at PESIT University, Bangalore, for her guidance and support.

Lastly, I thank the management, editorial, and production staff of BPB Publications, New Delhi, for bringing out this book in record time.

I am deeply grateful to all the contributors who have helped me to move this book from a dream to reality.

—Swati Saxena

Preface

Python was originally conceptualized by Guido van Rossum in the late 1980s as a member of the National Research Institute of Mathematics and Computer Science. Initially, it was designed as a response to the ABC programming language that was also foregrounded in the Netherlands.

Python solved questionnaire covers all the possible interview questions and coding in Python. As all the interviewers do not follow the same pattern, that's why I have written theory as well as practical questions. Questions are jumbled and compiled.

Features of this book are:

- Easy language for a quick understanding of the topics
- Questions are jumbled as mostly asked in interviews

Python Solved Questionnaire is a helping hand for those appearing for an interview or examination.

Practical questions may help you to understand the logic and will help you to fight the technical round. Simple questions with deep coding are the hallmark of this book.

With the 227 questions in this book, you will be able to crack your Python interview. The book covers the following topics:

Variable, Datatype, type conversion, Operators, if-else, loops, List, Tuples, Set, Dictionary, Functions, Array classes and objects, constructor, Inheritance, Encapsulation, keywords, regular expression, Random Module, Sys Module, OS Module, Statistics Module, widgets of Tkinter, Multithreading, other GUI Framework, work on multiple Tkinter windows, File Input-output, file handling with GUI, MySQL, SQLite, MongoDB, Redis, connectivity with GUI, Matplotlib Library, Django, Flask.

I hope this book will fulfill all the needs of students and learners.

It is advisable to practice "C programming and coding question bank with the solution" and "Java A complete practical solution' when practicing this book for a strong programming concept.

Despite the fact that ample care has been taken, the possibility of minor

inaccuracies cannot be ruled out. So, your suggestions, if any, are highly solicited.

Lastly, a big thanks to all students who have faith in me.

All the best.

—Swati Saxena

Downloading the coloured images:

Please follow the link to download the
Coloured Images of the book:

https://rebrand.ly/74d4f

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

BPB is searching for authors like you

If you're interested in becoming an author for BPB, please visit **www.bpbonline.com** and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Python-Interview-Questions**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at **https://github.com/bpbpublications**. Check them out!

PIRACY

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**.

REVIEWS

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Table of Contents

Python Solved Questionnaire

Learning objective:

This book will help you to learn:

- The core concept of Python
- The OOPs concept
- Modules in Python
- Python GUI (Tkinter)
- File Handling
- Python database
- NumPy, Pandas
- Django, Flask

Let us begin!

1. **What is the history behind Python?**

   Ans. Python was released in 1991 by *Guido van Rossum*. The history behind the name is, in the 1970s, there was a popular BBC comedy TV show called *Monty Python's Fly Circus* and *Van Rossum* happened to be the big fan of that show. So, when Python was developed, Rossum named the project *Python*.

2. **What is Python?**

   Ans. Python is a high-level, interpreted, interactive, and object-oriented scripting language. It is designed to be highly readable:

   - Points to know about Python -It supports functional and structured programming methods as well as OOP.
   - It supports automatic garbage collection.
   - It can be used as a scripting language and can be compiled to byte-code for building large applications.
   - It provides very high-level dynamic data types and supports dynamic type checking.

- It can be easily integrated with
    i. C,
    ii. C++,
    iii. COM,
    iv. ActiveX,
    v. CORBA,
    vi. Java.

3. **In which application area Python can be used?**

   Ans. Python can be built in the following areas:

   - GUI-based desktop applications
   - Web applications
   - Games
   - Scientific and computational applications
   - Language development
   - Enterprise and business applications development
   - Operating systems

4. **What are the benefits of Python?**

   Ans. Here are the benefits of Python:

   - **Open source:** Python language is developed under an OSI-approved open source license, which makes it free to use and distribute, including for commercial purposes.
   - **Easy learning:** Python offers excellent readability and uncluttered simple-to-learn syntax which helps beginners to utilize this programming language.
   - **Extensive support library:** Python provides a large standard library that includes areas such as Internet protocols, string operations, web services tools, and operating system interfaces. It reduces the length of code to be written significantly.
   - **User-friendly data structure:** Python has built-in list and dictionary data structures that can be used to construct fast runtime data structures.
   - **Productivity and speed:** Python has clean object-oriented design,

provides enhanced process control capabilities and possesses strong integration and text processing capabilities and its unit testing framework, all of which contribute to the increase in its speed and productivity.

5. **Explain memory management in Python?**

   Ans. Memory management is the process of efficiently allocating, de-allocating, and coordinating memory so that all the different processes run smoothly and can optimally access different system resources. Memory management also involves cleaning the memory of objects that are no longer being accessed.

   In Python, the memory manager is responsible for these kinds of tasks by periodically running to clean up, allocate, and manage the memory. Unlike C, Java, and other programming languages, Python manages objects by using reference counting. This means that the memory manager keeps track of the number of references to each object in the program.

   Lets understand memory management by following points:

   - Python memory is managed by Python private heap space. All Python objects and data structures are located in a private heap. The programmer does not have access to this private heap and the interpreter takes care of this Python private heap.
   - The allocation of Python heap space for Python objects is done by the Python memory manager. The core API gives access to some tools for the programmer to code.
   - The Python memory manager manages chunks of memory called "*Blocks*". A collection of blocks of the same size makes up the "*Pool*". Pools are created on Arenas, chunks of 256kB memory allocated on `heap=64` pools. If the objects get destroyed, the memory manager fills this space with a new object of the same size.
   - 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.

6. **What are the different environment variables in Python? And what's the use of these variables?**

   Ans. Here are the different environment variables and its uses:

   - `PYTHONPATH`: It is the same as the PATH variable. The Python

interpreter uses it to search the module files.

- `PYTHONSTARTUP`: It stores the path of an initialization script containing the Python code. It gets to run every time the Python interpreter starts.
- `PYTHONCASEOK`: In Windows, it makes the Python find the first case-insensitive match in an import statement. You need to set it for activation.

7. **Is Python Scripting language?**

   Ans. Yes, Python is a scripting language. It is also an interpreted and high-level programming language for the purpose of general programming requirements. It was designed and developed by the Software Developer named Guido van Rossum.

   It was first released in the year 1991. It is a dynamic type of discipline and has strong typing too. Filename extensions for python scripting language are of different types such as `.py, .pyc, .pyd, .pyo, .pyw, .pyz`.

8. **List the datatypes supported by Python?**

   Ans. Here are the datatypes:

   - **Text type:** `str`
   - **Numeric types:** `int, float, complex`
   - **Sequence types:** `list, tuple, range`
   - **Mapping type:** `dict`
   - **Set types:** `set, frozenset`
   - **Boolean type:** `bool`
   - **Binary types:** `bytes, bytearray, memoryview`

9. **What is the output of the following?**

```
str=”swati computers”?
print (str)
```

   Ans. It would print the complete string “`swati computers`”

10. **What is the output of the following**

```
str=”swati computers”?
print str[0]
```

   Ans. It would print “`s`”

11. **What is the output of the following**

```
str= “swati computers!” ?
print str* 2
```

Ans. It will print the string two times. `Swati computers! Swati computers!`

12. **What is the output of the following?**

```
list = [109, ‘swati’]?
print list * 2
```

Ans. It will print the list two times. Output would be `[109, ‘swati’, 109, ‘swati’]`.

13. **How to check keywords in Python?**

Ans. Type the following code:

```
import keyword
print(keyword.kwlist)
```

14. **What is pep 8?**

Ans. PEP stands for Python Enhancement Proposal. It is a set of rules that specify how to format Python code for maximum readability.

15. **What is the output of following?**

```
str=”{s}{c}{j}”.format(j=’Jaipur’,s=’Swati’,c=’Computers’)
print(str)
```

Ans. `SwatiComputersJaipur`

16. **What will be the output of the following?**

```
str=”apple#banana#kiwi#orange”
print(str.split(“#”,2))
```

Ans. `[‘apple’, ‘banana’, ‘kiwi#orange’]`

17. **What are python modules? Name some commonly used built-in modules in Python?**

Ans. Python modules are files containing Python code. This code can either be function classes or variables. A Python module is a `.py` file containing executable code. Some of the commonly used built-in modules are:

- `Os`
- `sys`
- `math`

- `random`
- `data time`
- `json`

18. **What are the Local and Global variables in Python?**

    Ans. Global variables are defined outside of any module and function. These variables can be accessed by any function in the program. Local variables are defined inside the function where it is used.

19. **What is the meaning of type conversion in Python?**

    Ans. Type conversion refers to the conversion of one data type into another:

    - `int()`: Converts any data type into the integer type.
    - `float()`: Converts any data type into float type.
    - `ord()`: Converts characters into integer.
    - `hex()`: Converts integers to hexadecimal.
    - `oct()`: Converts an integer to the octal.
    - `tuple()`: This function is used to convert to a tuple.
    - `set()`: This function returns the type after converting to set.
    - `list()`: This function is used to convert any data type to a list type.
    - `dict()`: This function is used to convert a tuple of order (key, value) into a dictionary.
    - `str()`: Used to convert an integer into a string.
    - `complex(real,imag)`: This function converts real numbers to `complex(real,imag)` number.

    Example: 1

```
num_int = 100
num_flo = 100.23
num_new = num_int + num_flo

print("datatype of num_int:",type(num_int))
print("datatype of num_flo:",type(num_flo))

print("Value of num_new:",num_new)
print("datatype of num_new:",type(num_new))
```

Solution:

```
datatype of num_int: <class 'int'>
datatype of num_flo: <class 'float'>
Value of num_new: 200.23000000000002
datatype of num_new: <class 'float'>
```

Example: 2

```
print('ASCII value of "S" is: ' + str(ord('S')))
print('Hexadecimal value of 253 is: ' + str(hex(253)))
print('Octal value of 57 is: ' + str(oct(57)))
print('Binary value of 54 is: ' + str(bin(54)))
```

Solution:

```
ASCII value of "S" is: 83
Hexadecimal value of 253 is: 0xfd
Octal value of 57 is: 0o71
Binary value of 54 is: 0b110110
```

Note: You can try on Jupyter notebook for every datatype given.

20. **Explain List in Python?**

    Ans. Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of the different data types.

    The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 at the beginning of the list and working their way to end -1.

Note: Please check Q. 22 for example

21. **What do the signs + and * mean in List in Python?**

    Ans. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

22. **Give an example of Python List?**

    Ans.

```
list = ['swati', 109, 1]
```

```
print (list) # Prints complete list
print (list[0])# Prints first element of the list
print (list[1:3])# Prints elements starting from 2nd till 3rd
print (list[2:])# Prints elements starting from 3rd element
print (list* 2)# Prints list two times
print (list + list)# Prints concatenated lists
```

23. **What are the tuples?**

Ans. A tuple is a datatype similar to the Python list. It consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses ().

Note: Please check Q.26 for example

24. **Difference between List and tuple?**

Ans. The following table shows the required difference:

List	Tuple
It is mutable in nature	It is immutable
Values in lists are enclosed in []	Values are enclosed in ()

Table 1

Note: Everything in Python is Object. Mutable objects can be changed after it is created while the immutable object cannot change. Objects of built-in types like (`int`, `float`, `bool`, `str`, `tuple`, `unicode`) are immutable. Objects of built-in types like (`list`, `set`, `dict`) are mutable. Custom classes are generally mutable.

25. **What is the use of zip () in python ?**

Ans. The `zip()` returns an iterator and takes iterable as argument. These iterables can be list, tuple, dictionary etc. It maps similar index of every iterable to make a single entity.

Example 1:

```
name=[“swati”,”shweta”]
age=[10,20]
new_entity=zip(name,age)
new_entity=set(new_entity)
print(new_entity)
```

The output is **{('shweta', 20), ('swati', 10)}**

Example 2:

```
a=["1","2","3"]
b=["a","b","c"]
c=[x+y for x, y in zip(a,b)]
print(c)
```

The output is: **['1a', '2b', '3c']**

If the passed iterators have different number of argument then the iterator with the least items decides the length of the new iterator.

Example 3:

```
name=["swati","shweta","Sweety"]
age=[10,20]
new_entity=zip(name,age)
new_entity=set(new_entity)
print(new_entity)
```

The output is: **('shweta', 20), ('swati', 10)}**

26. **Example of the tuple?**

Ans.

```
tuple = ('swati', 109, 1)

print (tuple)# Prints complete list
print (tuple[0])# Prints first element of the list
print (tuple[1:3])# Prints elements starting from 2nd till 3rd
print (tuple[2:])# Prints elements starting from 3rd element
print (tuple * 2)# Prints list two times
print (tuple + tuple)# Prints concatenated lists
```

The output is as follows:

```
In [5]: tuple = ( 'swati', 109 , 1 )


        print( tuple    )
        print (tuple[0] )
        print (tuple[1:3])
        print (tuple[2:] )
        print (tuple * 2 )
        print (tuple + tuple)

('swati', 109, 1)
swati
(109, 1)
(1,)
('swati', 109, 1, 'swati', 109, 1)
('swati', 109, 1, 'swati', 109, 1)
```

Figure 1

27. **What is the dictionary in Python?**

Ans. Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. Dictionaries are enclosed by curly braces ({}) and values can be assigned and accessed using square braces ([]).

Note: Please check example 29

28. **Create a two (multi) dimensional list?**

Ans.

```
two_dim_list = [[0] * 3]*3

print(two_dim_list)

two_dim_list[0][0] = 1
two_dim_list[0][1] = 2
two_dim_list[0][2] = 3

two_dim_list[1][0] = 4
two_dim_list[1][1] = 5
two_dim_list[1][2] = 6

two_dim_list[2][0] = 7
two_dim_list[2][1] = 8
two_dim_list[2][2] = 9

print(two_dim_list)
```

Take a look at the following output:

```
In [14]: two_dim_list = [[0] * 3]*3

         print(two_dim_list)

         two_dim_list[0][0] = 1
         two_dim_list[0][1] = 2
         two_dim_list[0][2] = 3

         two_dim_list[1][0] = 4
         two_dim_list[1][1] = 5
         two_dim_list[1][2] = 6

         two_dim_list[2][0] = 7
         two_dim_list[2][1] = 8
         two_dim_list[2][2] = 9

         print(two_dim_list)


         [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
         [[7, 8, 9], [7, 8, 9], [7, 8, 9]]
```

Figure 2

29. **Example of the dictionary?**

Ans.

```
dict = {}
dict['one'] = "This is one"
dict[2] = "This is two"

keyVal = {'name': 'swati','age':10, 'dept': 'software'}

print(dict['one']) # Prints value for 'one' key
print(dict[2]) # Prints value for 2 key
print(keyVal)# Prints complete dictionary
print(keyVal.keys()) # Prints all the keys
print(keyVal.values()) # Prints all the values
```

The output is as follows:

```
In [6]: dict = {}
        dict['one'] = "This is one"
        dict[2]     = "This is two"

        keyVal = {'name': 'swati','age':10, 'dept': 'software'}


        print(dict['one'])       # Prints value for 'one' key
        print(dict[2])           # Prints value for 2 key
        print(keyVal)            # Prints complete dictionary
        print(keyVal.keys())     # Prints all the keys
        print(keyVal.values())   # Prints all the values


        This is one
        This is two
        {'name': 'swati', 'age': 10, 'dept': 'software'}
        dict_keys(['name', 'age', 'dept'])
        dict_values(['swati', 10, 'software'])
```

Figure 3

30. **Write a program to convert a string into int?**

    Ans.

```
age = "18"

print(age)

# Converting string to integer
int_age = int(age)

print(int_age)
```

31. **What is frozenset() in Python?**

    Ans. A Set is an unordered collection data type that is iterable, mutable, and has no duplicate elements. The `frozenset()` is an inbuilt function in Python that takes an iterable object as input and makes them immutable. In Python, frozenset is the same as set except its elements are immutable.

```
# tuple of numbers
num = (1, 2, 3, 4, 5)

# converting tuple to frozenset
fnum = frozenset(num)

# printing details
print("frozenset Object is : ", fnum)
```

Answer: frozenset Object is : `frozenset({1, 2, 3, 4, 5})`

32. **Example to convert int to string?**

Ans.

```
a=10
# printing integer converting to string
c = str(a)
print (“After converting integer to string : “,end=””)
print (c)
```

Answer: After converting integer to string : `10`

33. **What is _____name____ in Python?**

Ans. Python has no inbuilt main function to start the execution of the program. Python has special variable `__name__`; it provides the functionality of the main function. Here is the output:

```
def func1():
    print ("The value of __name__ is " + __name__)

if __name__ == '__main__':
    func1()


The value of __name__ is __main__
```

Figure 4

34. **Explain different operators in Python?**

Ans. The Python language supports the following types of operators:

- **Arithmetic operators:** +, -, *, /, ** (Exponent), // floor division, % (Modulus)

  For example: 9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0
- **Relational operators** :<, >, <=, >=, !=, <>, ==
- **Assignment operators:** =, +=, - =, * =, /=, //=, %=
- **Logical operators:** and, or, not
- **Bitwise operators:** &, |, ~, <<, >>, ^
- **Membership operators:** Python’s membership operators test for membership in a sequence, such as strings, lists, or tuples.

Operator	Description	Example
`in`	Evaluates to true if it finds a variable in the specified sequence	x in y, herein results in a 1 if x is a member of sequence y.

	and false otherwise.	
not in	Evaluates to true if it does not find a variable in the specified sequence and false otherwise.	x not in y, here “not in” results in a 1 if x is not a member of sequence y.

Table 2

```
a = 10
b = 20
list = [1, 2, 3, 4, 5];

if (a in list):
 print(“Line 1 - a is available in the given list”)
else:
 print (“Line 1 - a is not available in the given list”)

if (b not in list):
 print (“Line 2 - b is not available in the given list”)
else:
 print (“Line 2 - b is available in the given list”)
```

- **Identity operators:** Identity operators compare the memory locations of two objects.

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here are results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here are not results in 1 if id(x) is not equal to id(y).

Table 3

```
a = 20
b = 20

if (a is b):
 print (“Line 1 - a and b have same identity”)
else:
 print(“Line 1 - a and b do not have same identity”)

if (id(a) == id(b)):
 print (“Line 2 - a and b have same identity”)
else:
 print (“Line 2 - a and b do not have same identity”)

b = 30
if (a is b):
```

```
 print (“Line 3 - a and b have same identity”)
else:
 print(“Line 3 - a and b do not have same identity”)

if (a is not b):
 print (“Line 4 - a and b do not have same identity”)
else:
 print (“Line 4 - a and b have same identity”)
```

When you execute the preceding program, it produces the following result:

```
Line 1 - a and b have same identity
Line 2 - a and b have same identity
Line 3 - a and b do not have same identity
Line 4 - a and b do not have same identity
```

35. **Write a program to convert all string in the list to an integer?**

Ans.

```
test_list = [‘10’, ‘14’, ‘1’, ‘6’, ‘7’]

# Printing original list
print (“Original list is : “ + str(test_list)) # conversion
revise, here you need not to
#convert

# perform conversion
For i in range(0, len(test_list)):
 test_list[i] = int(test_list[i])

# Printing modified list
print (“Modified list is : “ + str(test_list))
```

The output is as follows:

```
Original list is: [‘10’, ‘14’, ‘1’, ‘6’, ‘7’]
Modified list is: [10, 14, 1, 6, 7]
```

36. **Remove empty strings from list of strings?**

Ans.

```
# initializing list
test_list = [“”, “swatiComputers”, “”, “is”, “best”, “”]

# Printing original list
print (“Original list is : “ + str(test_list))

# using remove() to
# perform removal
```

```
while(“” in test_list) :
 test_list.remove(“”)

# Printing modified list
 print (“Modified list is : “ + str(test_list))
```

The output is as follows:

```
Original list is: [‘’, ‘swatiComputers’, ‘’, ‘is’, ‘best’, ‘’]
Modified list is: [‘swatiComputers’, ‘is’, ‘best’]
```

37. **How is Python an interpreted language?**

Ans. An interpreter is a kind of program that executes other programs. When you write Python programs, it converts source code written by the developer into the intermediate language which is again translated into the native language/machine language that is executed. The Python code you write is compiled into Python bytecode, which creates a file with extension `.pyc`. The compilation is simply a translation step, and byte code is a lower-level, and platform-independent, representation of your source code.

The `.pyc` file, created in the compilation step, is then executed by appropriate virtual machines. The Virtual Machine is the runtime engine of Python and it is always present as part of the Python system and is the component that truly runs the Python scripts. Technically, it’s just the last step of what is called the Python interpreter.

38. **What are the .pyc files?**

Ans. The `.py` files contain the source code of a program, whereas the `.pyc` file contains the bytecode of your program. We get bytecode after compilation of the `.py` file (source code). The `.pyc` files are not created for all the files that you run. It is only created for the files that you import.

39. **Write a program to ask two numbers and print their addition?**

Ans.

```
num1 = int(input(“Enter first number))
num2 = int(input(“Enter second number”))

# printing the sum in integer
print(num1 + num2)
```

40. **Write a program to input N numbers in a list?**

Ans.

```
nums= []
```

```
# number of elements as input
n = int(input(“Enter number of elements : “))

# iterating till the range
fori in range(0, n):

 ele = int(input(“Enter number”))
 nums.append(ele) # adding the element

print(nums)
```

Here is the screen with output:

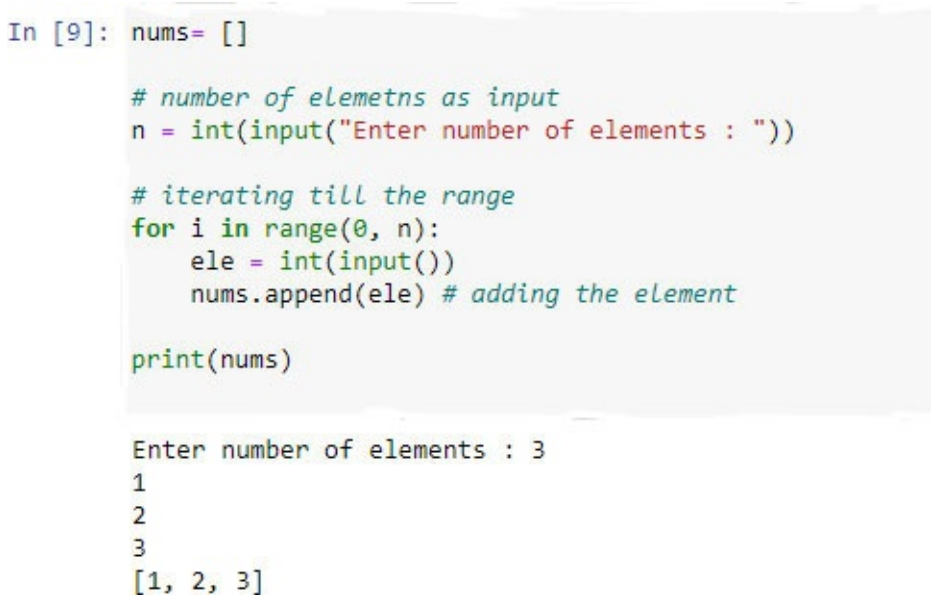


Figure 5

41. **Write a program to find out greater out of three?**

Ans.

```
#input values or assign
a=2
n=3
c=4

#code for check greater number

if (a>b)and (a>c) :
 print “A is greater”
elif (b>a) and (b>c) :
 print “B is greater”
else:
 print “C is greater”
```

Or

```
Take input from user
a=int(input(“enter num”))
b=int(input(“enter num”))
c=int(input(“enter num”))
```

42. **Write a program to find out prime number between 1 to 100?**

Ans.

```
for num in range(1,100):
 for i in range(2,num):
  if num%i == 0:
   j=num/i
   print (‘not prime’)
   break #to move to the next number, the #first FOR
  else:
   print(num, ‘is a prime number’)
```

43. **Write a program to print values of tuple?**

Ans.

```
name = [‘swati’, ‘sweety’, ‘shweta’]

for id in range(len(name)):
 print (‘Hello :’,name[id])
```

The output is as follows:

```
Hello : swati
Hello : sweety
Hello : shweta
```

44. **Write a program to print letters of the name?**

Ans.

```
name=”swati”

fori in name :
print (i)
print (‘\n’)
```

45. **Write the code to display the following pattern:**

Ans.

```
for i in range (1:5):
 for j in range(5,i,-1):
  print(end=” “)
 for k in range(1:i):
  print(“* “,end=” “)
 print(“\r”)
```

46. **Write a code to display the following character pattern:**

```
A
B B
C CC
D DDD
E EEEE
```

Ans.

```
num = 65
fori in range(0, 5):
for j in range(0, i+1):
ch = chr(num)
    # printing char value
print(ch, end=” “)
num = num + 1
   # ending line after each row
print(“\r”)
```

Following image shows the code and output:

```
In [13]: num = 65

for i in range(0, 5):

    for j in range(0, i+1):
        ch = chr(num)

            # printing char value
        print(ch, end=" ")

        num = num + 1

        # ending line after each row
    print("\r")
```

```
A
B C
D E F
G H I J
K L M N O
```

Figure 6

47. **Write a program to print 1 to 10 using a while loop?**

Ans.

```
count = 1
while (count < 11):
print('num:', count)
count = count + 1
```

48. **Explain and Print a random number?**

Ans. Python number method and `range()` returns a randomly selected element from range (`start, stop, step`):

- `start`: Start point of the range. This would be included in the range.
- `stop`: Stop point of the range. This would be excluded from the range.
- `step`: Steps to be added in a number to decide a random number.

```
import random

# Select an even number in 100 <= number < 1000
print ("randrange(100, 1000, 2) : ", random.randrange(100,
1000, 2))

# Select another number in 100 <= number < 1000
```

```
print (“randrange(100, 1000, 3) : “, random.randrange(100, 1000, 3))
```

The output for the code will be as follows:

```
randrange(100, 1000, 2) : 976
randrange(100, 1000, 3) : 520
```

49. **How to call the base class inherited method in the child class?**

Ans. Using super(): Refer to the following screenshot:

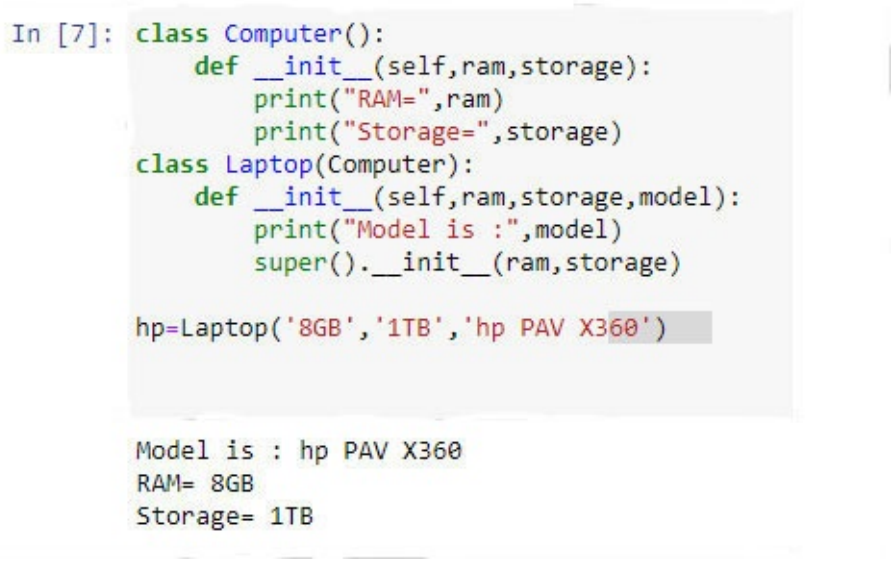


Figure 7

50. **As in Java “instanceof” is used to check whether a class is subclass of given class or not, which function or keyword is used in Python to check subclass?**

Ans. Syntax:

```
issubclass(subClassName,superClassName)
```

Example:

```
issubclass(equation,Addition) # it will return true
issubclass(subtraction,equation) # it will return false
```

51. **Is there any destructor in Python?**

Ans. Destructors are called when an object gets destroyed. Python has a garbage collector that handles memory management automatically. The `__del__()` method is known as a destructor method in Python. It is called when all references to the object have been deleted.

Example:

```
class Student:

def __init__(self):
 print('welcome student…')

def __del__(self):
 print('Destructor called, student deleted')

obj = Student()
#del obj
```

52. **How does Python Garbage collector works?**

Ans. Python deletes unwanted objects (built-in types or class instances) automatically to free the memory space. Python's garbage collector runs during program execution and is triggered when an object's reference count reaches zero. An object's reference count changes as the number of aliases that points changes. An object's reference count increases when it is assigned a new name or placed in a container (list, tuple, or dictionary). The object's reference count decreases when it's deleted with del, its reference is reassigned, or its reference goes out of scope. When an object's reference count reaches zero, Python collects it automatically. You can also call `del ()` to delete the unused object.

Example:

```
Point is a class, and pt is an object
pt = Point()
del pt
```

53. **Write a code to create a for loop that iterates through a tuple of class objects. Then call the methods without being concerned about which class type each object is?**

Ans.

```
class shape:
 def __init__(self):
  print("I am a shape")

class rect(shape):
 def myself(self):
  print("Myself rectangle")

class sqr(shape):
 def myself(self):
```

```
  print("Myself square")

r=rect()
s=sqr()

for sh in (r,s):
 sh.myself()
```

54. **Give an example of overriding.**

Ans.

```
class shape:
def __init__(self):
print("I am a shape")
def area(self):
print("Every shape has area")
class rect(shape):
def myself(self):
print("Myself rectangle")
def area(self):
print("Area of rectangle")

class sqr(shape):
def myself(self):
print("Myself square")
def area(self):
print("Area of square")

r=rect()
s=sqr()

for sh in (r,s):
sh.myself()
sh.area()
```

```
In [7]: class shape:
            def __init__(self):
                print("I am a shape")
            def area(self):
                print("Every shape has area")

        class rect(shape):
            def myself(self):
                print("Myself rectangle")
            def  area(self):
                print("Area of rectangle")

        class sqr(shape):
            def myself(self):
                print("Myself square")
            def area(self):
                print("Area of square")

        r=rect()
        s=sqr()

        for sh in (r,s):
            sh.myself()
            sh.area()

I am a shape
I am a shape
Myself rectangle
Area of rectangle
Myself square
Area of square
```

Figure 8

Note: Overriding is the concept of OOPs(Object-oriented programming).

Pillars of OOPs:

- **Abstraction:** To hide complexity and show which is necessary to a current problem scenario.
- **Encapsulation:** Combine data of the similar type of objects into a single unit.
- **Polymorphism:** Many forms of a single object.

Polymorphism is of two types:

- **Early binding:** Example - Overloading

- **Late Binding:** Example - Overriding
- **Inheritance:** Use for code reusability, where child (sub) class can use properties of the parent (base) class.

To know more about OOPs, refer *"Java -A complete practical solution"* by BPB Publications.

55. **Create a class Robot and set the robot name by the setter method, constructor?**

Ans.

```
class Robot:

def __init__(self,name=None):
   self.name = name

def say_hi(self):
if self.name:
print("Hello, I am " + self.name)
else:
print("Hello, I am a robot without a name")

def set_name(self, name):
   self.name = name

def get_name(self):
return self.name

 x = Robot("Swati")
 y = Robot()
 z = Robot()

 y.set_name("Swati")

 x.say_hi()
 y.say_hi()
 z.say_hi()
```

```
2]: class Robot:

        def __init__(self,name=None):
            self.name = name

        def say_hi(self):
            if self.name:
                print("Hello, I am " + self.name)
            else:
                print("Hello, I am a robot without a name")

        def set_name(self, name):
            self.name = name

        def get_name(self):
            return self.name

    x = Robot("Swati")
    y = Robot()
    z = Robot()

    y.set_name("Swati")

    x.say_hi()
    y.say_hi()
    z.say_hi()
```

```
Hello, I am Swati
Hello, I am Swati
Hello, I am a robot without a name
```

Figure 9

56. **Create a queue in Python and print all its elements.**

Ans.

```
import queue

# Queue is created as an object 'obj'
obj = queue.Queue(maxsize=10)

# Data is inserted in 'obj' at the end using put()
obj.put(19)
obj.put(64)
obj.put(73)
obj.put(46)

#while not obj.empty:
#print(obj.get())

print(obj.get())
print(obj.get())
print(obj.get())
```

```
print(obj.get())
```

The output is as follows:

```
19
64
73
46
```

57. **What is GUI programming? How to create GUI applications in Python?**

Ans. Modern applications are not restricted to console input/output, they have a user-friendly graphical user interface. These applications can receive inputs through mouse clicks and can enable the user to choose from alternatives with the help of radio buttons, dropdown lists, and other GUI elements (or widgets). Such applications are developed using one of the various graphics libraries available.

A graphics library is a software toolkit having a collection of classes that define the functionality of various GUI elements. These graphics libraries are generally written in C/C++.

Python provides various options for developing GUI, one of them is Tkinter. Python provides the standard library Tkinter for creating the graphical user interface for desktop-based applications.

Here are the steps to create a Tkinter window:

i. Import the Tkinter module.
ii. Create the main application window.
iii. Add widgets such as labels, buttons, and frames to the window.
iv. Call the main event loop so that the actions can take place on the user's computer screen.

Example:

```
from tkinter import *
top=Tk()
top.mainloop()
```

Here's the screenshot:

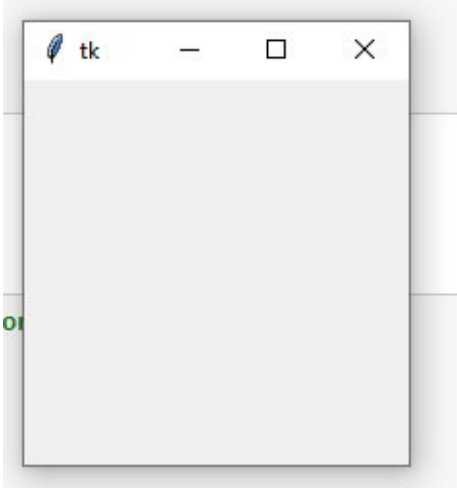


Figure 10

58. Explain Tkinter widgets.

SN	Widget	Description
1	`Button`	Click `Button`. The button is used to add buttons to the Python application.
2	`Canvas`	The `Canvas` widget is used to draw the canvas on the window.
3	`Checkbutton`	The `Checkbutton` is used to display the CheckBox on the window.
4	`Entry`	The entry widget is used to display the single-line text field to the user. It is commonly used to accept user values.
5	`Frame`	It can be defined as a container to which, another widget can be added and organized.
6	`Label`	A label is a text used to display some message or information about the other widgets.
7	`ListBox`	The `ListBox` widget is used to display a list of options to the user.
8	`Menubutton`	The `Menubutton` is used to display the menu items to the user.
9	`Menu`	It is used to add menu items to the user.
10	`Message`	The `Message` widget is used to display the message-box to the user.
11	`Radiobutton`	The `Radiobutton` is different from a checkbutton. Here, the user is provided with various options and the user can select only one option among them.
12	`Scale`	It is used to provide the slider to the user.
13	`Scrollbar`	It provides the scrollbar to the user so that the user can scroll the window up and down.

14	`Text`	It is different from entry because it provides a multi-line text field to the user so that the user can write the text and edit the text inside it.
14	`Toplevel`	It is used to create a separate window container.
15	`Spinbox`	It is an entry widget used to select from options of values.
16	`PanedWindow`	It is like a container widget that contains horizontal or vertical panes.
17	`LabelFrame`	`LabelFrame` is a container widget that acts as the container.
18	`MessageBox`	This module is used to display the message-box in the desktop-based applications.

Table 4

59. **Create a Tkinter window and change its title, height, width, position from the top and left.**

Ans.

```
from tkinter import *
top = Tk()
top.title("My Window")
top.geometry("300x300+200+200")
top.mainloop()
#top.geometry(width x height + from left + from top)
```

The following is the output:

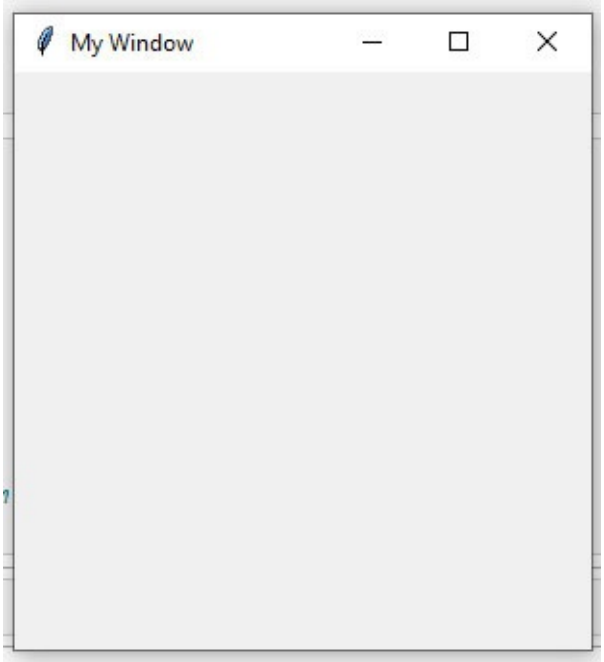


Figure 11

60. **How to restrict the window resize at runtime?**

    Ans.

    Function:

```
Resizable (true/false, true / false)
from tkinter import *
top = Tk()
top.title(“My Window”)
top.geometry(“300x300+200+200”)
top.resizable(0,0)
top.mainloop()
```

61. **Change the background color of the Tkinter window.**

    Ans.

```
from tkinter import *
top = Tk()

top.title(“My Window”)
top.geometry(“300x300+200+200”)

# top.configure(bg=’blue’)

top[‘background’]=’#856ff8’

top.resizable(0,0)

top.mainloop()
```

    Here is the screenshot:

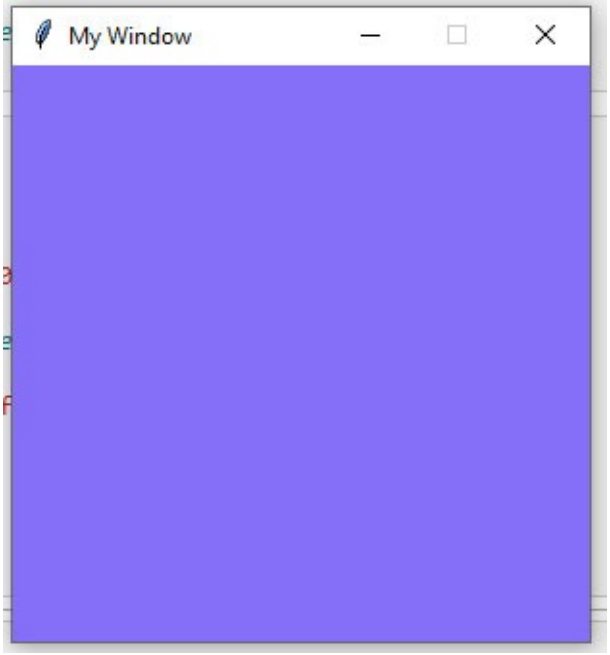


Figure 12

62. **Open the color picker dialog.**

    Ans.

```
from tkinter import *
from tkinter.colorchooser import *

top = Tk()

top.title("My Window")
top.geometry("300x300+200+200")

color=askcolor()

print(color[0])

#top['background']='#856ff8'
top.resizable(0,0)
top.mainloop()
```

    Take a look at the following image:

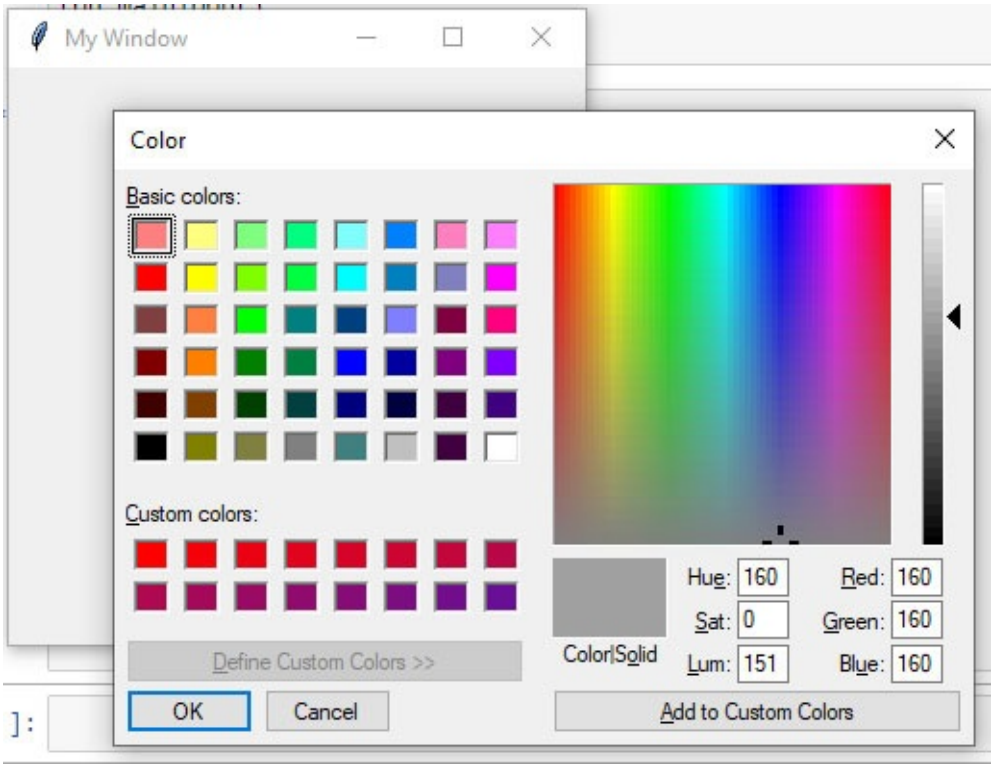


Figure 13

63. **Create a login box in Tkinter Python.**

Ans.

```
from tkinter import *
from tkinter.colorchooser import *
top = Tk()

top.title("Login Box")
top.geometry("300x200+200+200")
top.configure(bg='green')

uname = Label(top, text =
"Username",fg='yellow',bg='green').place(x = 30,y = 50)

#creating label
password = Label(top, text =
"Password",bg='green',fg='yellow').place(x = 30, y = 90)

sbmitbtn = Button(top, text = "Submit",bg= "yellow", fg=
"green").place(x = 100, y = 120)

e1 = Entry(top,width = 20).place(x = 100, y = 50)

e2 = Entry(top, width = 20).place(x = 100, y = 90)
top.mainloop()
```

Here is the output screenshot:

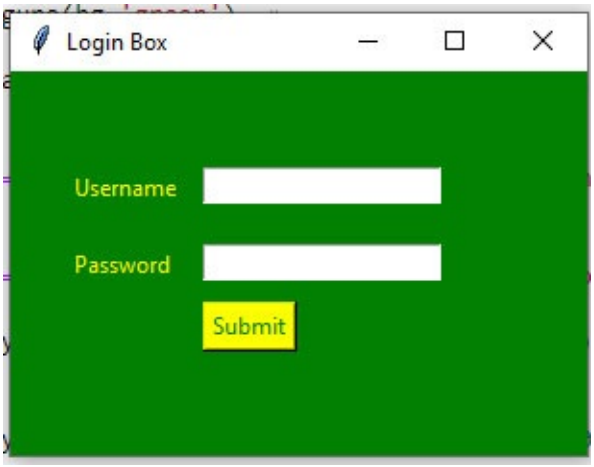


Figure 14

64. **Create a frame to add and multiply two given numbers.**

Ans.

```
import tkinter as tk
from functools import partial

def addi(answer, n1, n2):
 num1 = (n1.get())
 num2 = (n2.get())
result = int(num1)+int(num2)
answer.config(text="Result = %d" % result)
return

def multi(answer, n1, n2):
 num1 = (n1.get())
 num2 = (n2.get())
result = int(num1)*int(num2)
answer.config(text="Result = %d" % result)
return

root = tk.Tk()
root.geometry('400x200+100+200')
root.title('Calculator')

number1 = tk.StringVar()
number2 = tk.StringVar()

labelNum1 = tk.Label(root, text="A").grid(row=1, column=0)
labelNum2 = tk.Label(root, text="B").grid(row=2, column=0)

answer = tk.Label(root)
answer.grid(row=7, column=2)
```

```
entryNum1 = tk.Entry(root, textvariable=number1).grid(row=1,
column=2)
entryNum2 = tk.Entry(root, textvariable=number2).grid(row=2,
column=2)

addi= partial(addi, answer, number1, number2)
multi= partial(multi, answer, number1, number2)

buttonAdd = tk.Button(root, text="Add",
command=addi).grid(row=3, column=0)
buttonMulti = tk.Button(root, text="Multiply",
command=multi).grid(row=3, column=10)
root.mainloop()
```

Here is the screenshot:

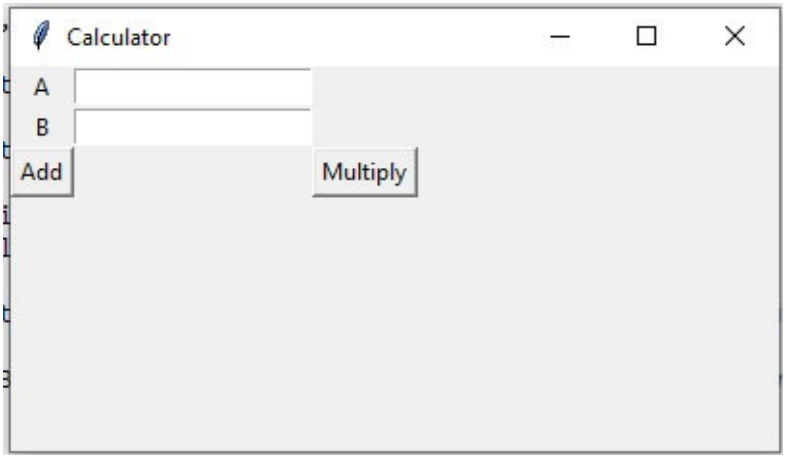


Figure 15

65. **What is the use of 'partial' in the preceding example?**

Ans.

As discussed in above program:

```
addi= partial(addi, answer, number1, number2)
multi= partial(multi, answer, number1, number2)

buttonAdd = tk.Button(root, text="Add",
command=addi).grid(row=3, column=0)

buttonMulti = tk.Button(root, text="Multiply",
command=multi).grid(row=3, column=10)
```

Partial is being used as one strategy for passing the parameters into the function you'd like the button to do (by not having parameters at all, and building a new function that just has the parameters baked into it).

If you do not need to pass an argument, avoid using partial.

Example:

```
def disp():
answer.config(text=”Hello user “)
return

buttonMulti = tk.Button(root, text=”Msg”,
command=disp).grid(row=3, column=7)
```

66. **Write a program to get selected fruit from the listbox in Python?**

Ans.

```
from tkinter import *
from functools import partial

top = Tk()

def display(answer,mylist):
 a=mylist.get(ACTIVE)
 answer.config(text=a)
 return

sb = Scrollbar(top)
sb.pack(side = RIGHT, fill = Y)
Lb=Label(top,text=”select Favourite Fruit”)
mylist = Listbox(top, yscrollcommand = sb.set)
answer=Label(top)

display=partial(display,answer,mylist)
b=Button(top,text=”Show”,command=display)

mylist.insert(1,”Kiwi”)
mylist.insert(2,”Mango”)
mylist.insert(3,”papaya”)
mylist.insert(4,”Orange”)
mylist.insert(5,”Apple”)

Lb.pack()
mylist.pack(side = LEFT)
sb.config(command = mylist.yview)
b.pack(side=LEFT)
answer.pack(side=RIGHT)

mainloop()
```

Here is the screenshot:

Figure 16

67. **Write a program to create three checkboxes for languages such as English, Hindi, French, and two radio buttons for gender selection, and on clicking on the thanks button, the window should close.**

Ans.

```
from tkinter import *

top = Tk()
top.geometry("400x200")

L=Label(top,text="Language known").place(x=30,y=30)

c1=Checkbutton(top,text="English").place(x=20,y=60)
c2=Checkbutton(top,text="Hindi").place(x=20,y=100)
c3=Checkbutton(top,text="French").place(x=20,y=140)

L=Label(top,text="Select Gender").place(x=180,y=30)

r1=Radiobutton(top,text="male").place(x=180,y=60)
r2=Radiobutton(top,text="female").place(x=180,y=100)

b=Button(top,text="Thanks",command=top.destroy).place(x=200,y=1
top.mainloop()
```

Here is the output:

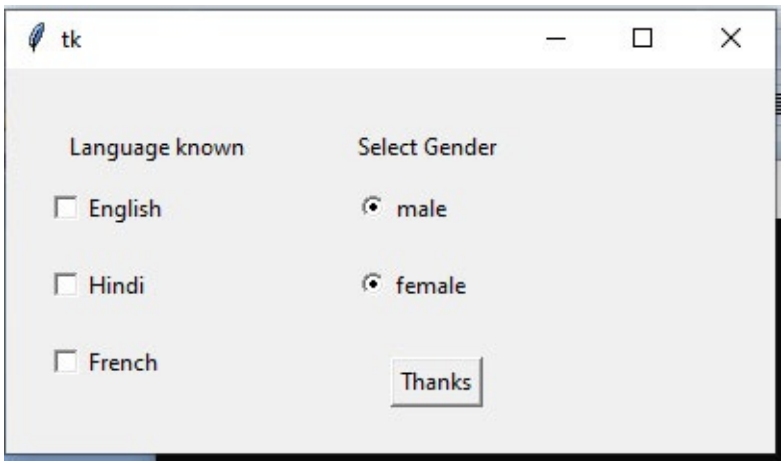


Figure 17

68. **Write a program to select radio buttons from different groups.**

Ans.

```
from tkinter import *

top = Tk()
top.geometry("400x200")

var=StringVar()
var1=StringVar()

L=Label(top,text="Qualification").place(x=30,y=30)

r1 = Radiobutton(top, text='Graduate', variable=var,
value='A').place(x=30,y=50)

r2 = Radiobutton(top, text='Under Graduate', variable=var,
value='B').place(x=30,y=80)

r3 = Radiobutton(top, text='Post Graduate', variable=var,
value='C').place(x=30,y=110)

L=Label(top,text=" Gender").place(x=180,y=30)

r1=Radiobutton(top,text="male",variable=var1,
value='male').place(x=180,y=60)

r2=Radiobutton(top,text="female",variable=var1,
value='female').place(x=180,y=100)

b=Button(top,text="Thanks",command=top.destroy).place(x=200,y=1

top.mainloop()
```

The following image is the output:

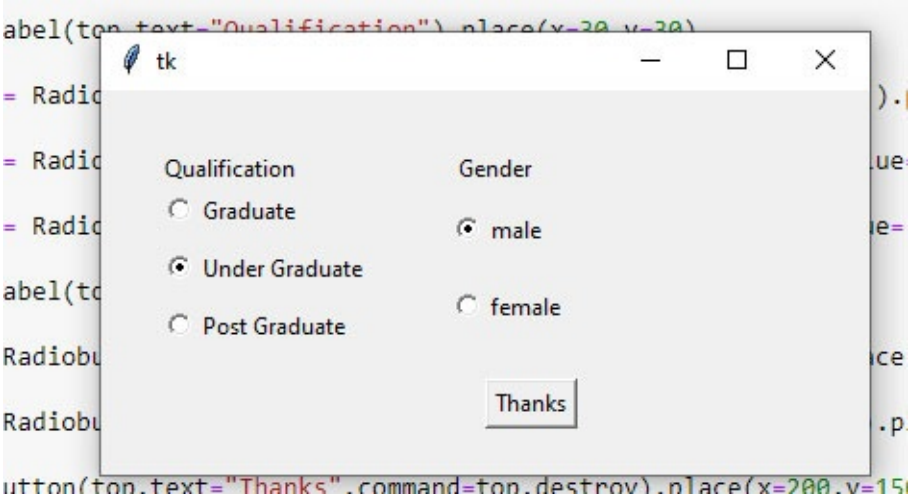


Figure 18

69. **Create a scale to select the value from 10 to 1000.**

Ans.

```
from tkinter import *

def sel():
 selection = "Value = " + str(var.get())
 label.config(text = selection)

root = Tk()
root.geometry("300x300")
var = DoubleVar()

scale = Scale(root, variable = var,from_=10,to=1000)
scale.pack(anchor=CENTER)

button = Button(root, text="Get Value", command=sel)
button.pack(anchor=CENTER)

label = Label(root)
label.pack()

root.mainloop()
```

The following is the output screenshot:

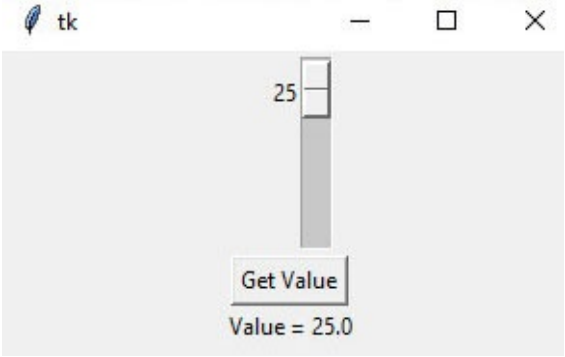


Figure 19

70. **Write a program to change the width of the label by changing the scale value?**

Ans.

```
from tkinter import *

def sel():
 selection = “Value = “ + str(var.get())
 w= int(var.get())
 label.config(width=w)

root = Tk()
root.geometry(“300x300”)
var = DoubleVar()

scale = Scale(root, variable = var,from_=10,to=1000)
scale.pack(anchor=CENTER)

button = Button(root, text=”Change Width”, command=sel)
button.pack(anchor=CENTER)

label = Label(root,bg=”RED”)
label.pack()

root.mainloop()
```

The output screenshot is as follows:

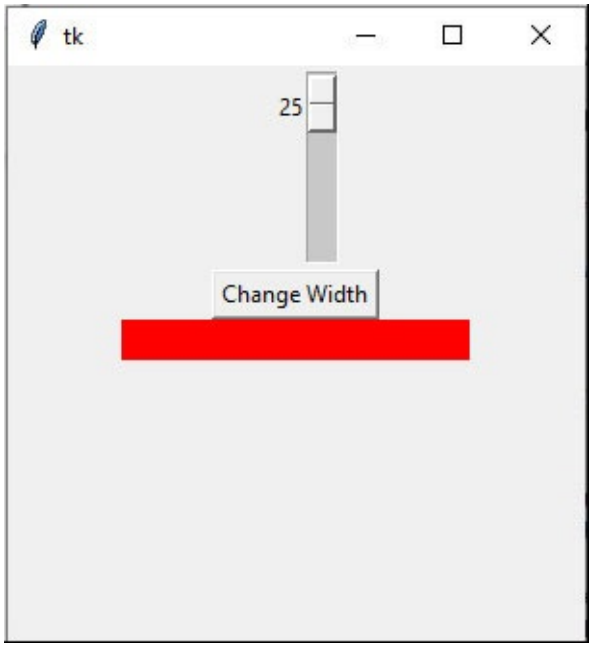


Figure 20

71. **Write a program to change label text on the select radio button?**

Ans.

```
import tkinter as tk
from functools import partial

top=tk.Tk()
top.geometry("300x300")

r11=tk.StringVar()

def gen():
 #print(r11.get())
 a=r11.get()
 label.config(text=a)

#gen=partial(gen,L)

r1=tk.Radiobutton(top,text="male",value="male",
command=gen,variable=r11).place(x=20,y=20)

r2=tk.Radiobutton(top,text="female",value="female",
variable=r11,command=gen).place(x=20,y=40)

label = tk.Label(top,bg="RED")
label.pack()

top.mainloop()
```

You will get the following screen:

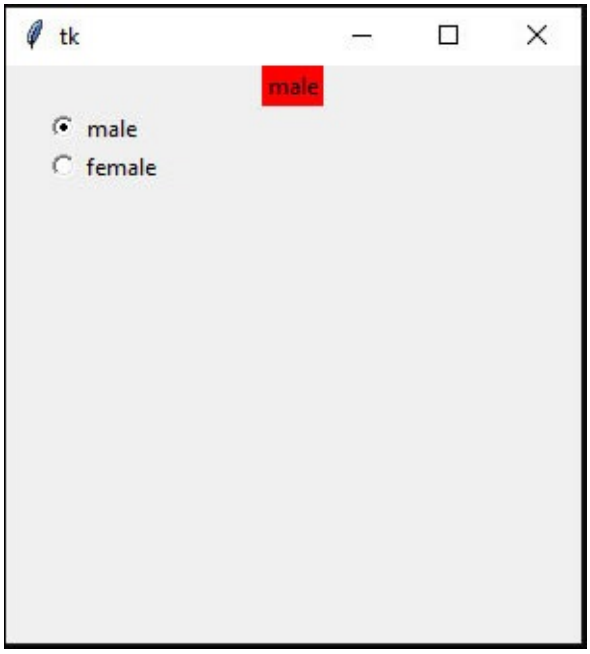


Figure 21

72. **List options of Tkinter Entry.**

Ans. The following table lists the options:

SNO	Option	Description
1	`Bg`	The background color of the widget.
2	`Bd`	The border width of the widget in pixels.
3	`Cursor`	The mouse pointer will be changed to the cursor type set to the arrow, dot, etc.
4	`Export selection`	The text written inside the entry box will be automatically copied to the clipboard by default. We can set `exportselection` to 0 to not copy this.
5	`Fg`	It represents the color of the text.
6	`Font`	It represents the font type of the text.
7	`Highlight background`	It represents the color to display in the traversal highlight region when the widget does not have the input focus.
8	`Highlight color`	It represents the color to use for the traversal highlight rectangle that is drawn around the widget when it has the input focus.
9	`Highlight thickness`	It represents a non-negative value indicating the width of the highlight rectangle to draw around the outside of the widget when it has the input focus.
	`Insert`	

10	`background`	It represents the color to use as a background in the area covered by the insertion cursor. This color will normally override either the normal background for the widget.
11	`Insert borderwidth`	It represents a non-negative value indicating the width of the 3-D border to draw around the insertion cursor. The value may have any of the forms acceptable to `Tk_GetPixels`.
12	`Insert offtime`	It represents a non-negative integer value indicating the number of milliseconds the insertion cursor should remain off in each blink cycle. If this option is zero, then the cursor doesn't blink: it is on all the time.
13	`Insert ontime`	Specifies a non-negative integer value indicating the number of milliseconds the insertion cursor should remain on in each blink cycle.
14	`Insertwidth`	It represents the value indicating the total width of the insertion cursor. The value may have any of the forms acceptable to `Tk_GetPixels`.
15	`Justify`	It specifies how the text is organized if the text contains multiple lines.
16	`Relief`	It specifies the type of the border. Its default value is `FLAT`.
17	`Select background`	The background color of the selected text.
18	`Selectbor derwidth`	The width of the border to display around the selected task.
19	`Select foreground`	The font color of the selected task.
20	`Show`	It is used to show the entry text or some other type instead of the string. For example, the password is typed using stars (*).
21	`Textvariable`	It is set to the instance of the `StringVar` to retrieve the text from the entry.
22	`Width`	The width of the displayed text or image.
23	`Xscroll command`	The entry widget can be linked to the horizontal scrollbar if we want the user to enter more text than the actual width of the widget.

Table 5

73. Create a Notepad window in Tkinter.

Ans.

```
from tkinter import Button, Tk, Menu, Label, Text
```

```
top = Tk()
def command():
 master2 = Tk()
master2.geometry(“300x300+0+2”)
master2.resizable(0,0)
master2.title(“New Window”)
 #label = Label(master2, text=”This is the new window”)
 #label.pack()
text=Text(top).place(x=0,y=0)
master2.mainloop()
menubar = Menu(top)
file = Menu(menubar, tearoff=0)
file.add_command(label=”New”,command=command)
file.add_command(label=”Open”)
file.add_command(label=”Save”)
file.add_command(label=”Save as…”)
file.add_separator()
file.add_command(label=”Close”)

file.add_command(label=”Exit”, command=top.quit)

menubar.add_cascade(label=”File”, menu=file)
edit = Menu(menubar, tearoff=0)
edit.add_command(label=”Undo”)

edit.add_command(label=”Cut”)
edit.add_command(label=”Copy”)
edit.add_command(label=”Paste”)
edit.add_separator()
edit.add_command(label=”Delete”)
edit.add_command(label=”Select All”)

menubar.add_cascade(label=”Edit”, menu=edit)

format=Menu(menubar)
format.add_command(label=”font”)
format.add_command(label=”wordWrap”)
menubar.add_cascade(label=”Format”,menu=format)
help = Menu(menubar, tearoff=0)
help.add_command(label=”About Notepad”)
menubar.add_cascade(label=”Help”, menu=help)

top.config(menu=menubar)
top.mainloop()
```

The following is the output:

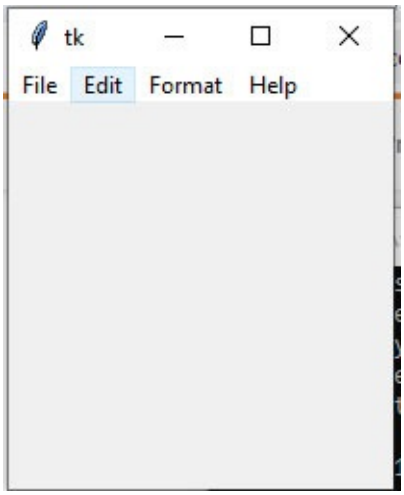


Figure 22

74. **Open Gmail on a button click.**

Ans.

```
from tkinter import Button, Tk, Menu, Label, Text

top = Tk()

def mail():
import webbrowser
webbrowser.open('https://gmail.com',new=1)
menubar = Menu(top)
file = Menu(menubar, tearoff=0)
file.add_command(label="Open Gmail",command=mail)

menubar.add_cascade(label="Gmail", menu=file)

top.config(menu=menubar)
top.mainloop()
```

The following screen will be visible:

Figure 23

75. **Write a program to calculate age, create a spin box or entry box to select the day, month and year, then subtract the year from the current year.**

Ans.

```
from tkinter import *
from datetime import date

top = Tk()

def age():

a=IntVar()
b=IntVar()
a=int(spin2.get())
b=int(today.year)
c=b-a
print(c)
msg=Label(top,text="Age : "+str(c))
msg.pack()

top.geometry("400x240")
today = date.today()

label=Label(top,text="select DOB")
label1=Label(top,text="select Day")

spin = Spinbox(top, from_= 1, to = 31)

label2=Label(top,text="select Month")
spin1 = Spinbox(top, from_= 1, to = 12)

label3=Label(top,text="select year")
spin2 = Spinbox(top, from_= 1970, to = 2020)

label.pack()
label1.pack()
spin.pack()
label2.pack()
spin1.pack()
label3.pack()
spin2.pack()

label4=Label(top,text="Today's Date :"+str(today))
label4.pack()

b=Button(top,text="calculate age in year",command=age)
b.pack()
top.mainloop()
```

The following is the output screenshot:

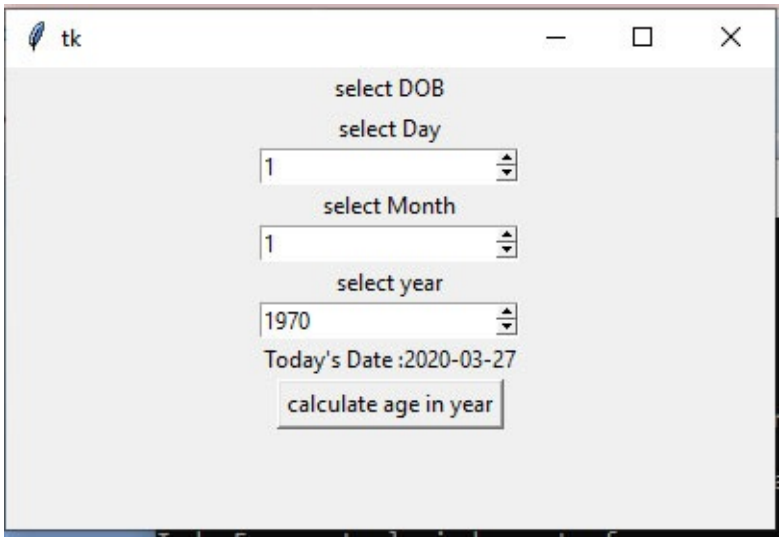


Figure 24

76. **Write a program to open different color windows on the click of the button?**

Ans.

```
from tkinter import *

class A:
 def CB(self):
 B()
def CC(self):
 C()
def CD(self):
 D()
def __init__(self):
 top = Tk()
 top.title("My Window")
 top.geometry("300x300+200+200")
 # top.configure(bg='blue')
 top['background']='#856ff8'

bt=Button(top,text="Green",command=self.CB)
bt.pack()

bt1=Button(top,text="Blue",command=self.CC)
bt1.pack()

bt2=Button(top,text="Red",command=self.CD)
bt2.pack()

top.resizable(0,0)
top.mainloop()
```

```
class B(object):

def __init__(self):
 top = Tk()
 top.title(“My Window”)
 top.geometry(“300x300+200+200”)
 # top.configure(bg=’red’)
 top[‘background’]=’#678423’
 top.resizable(0,0)
 top.mainloop()

class C(object):

def __init__(self):
 top = Tk()
 top.title(“My Window”)
 top.geometry(“300x300+200+200”)
 # top.configure(bg=’red’)
 top[‘background’]=’#345689’
 top.resizable(0,0)
 top.mainloop()

class D(object):

def __init__(self):
 top = Tk()
 top.title(“My Window”)
 top.geometry(“300x300+200+200”)
 # top.configure(bg=’red’)
 top[‘background’]=’#982432’
 top.resizable(0,0)
 top.mainloop()

def main():
 A()
 main()
```

The following is the output screen:

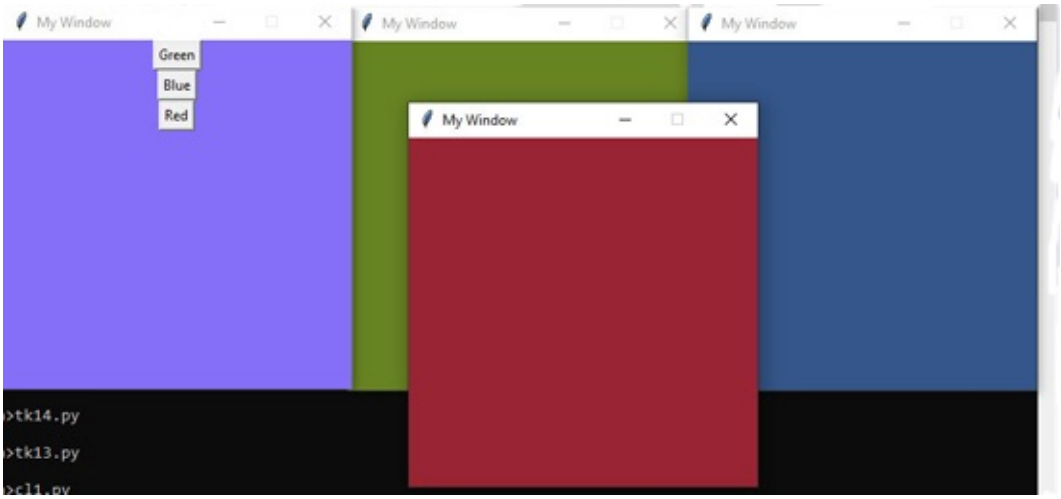


Figure 25

77. **Write a program to select an item from the combo box and add it to the listbox. You can remove items from the list box.**

Ans.

```
from tkinter import *
from tkinter.ttk import Combobox
window=Tk()

data=(“Apple”, “Kiwi”, “Papaya”, “Grapes”,”Banana”,”Orange”)

def add():
a=StringVar()
a=cb.get()
lb.insert(END,a)

def remove():
a=lb.curselection()
lb.delete(a[0])

var = StringVar()

lbl=Label(window,text=”Select favourite
fruit”).place(x=150,y=20)
bt=Button(window,text=”>>”,command=add).place(x=200,y=70)
bt=Button(window,text=”<<”,command=remove).place(x=200,y=110)
lb=Listbox(window, height=5, selectmode=’multiple’)

lb.place(x=250, y=50)

cb=Combobox(window, values=data)
cb.place(x=30, y=50)

window.title(‘Hello Python’)
window.geometry(“400x300+10+10”)
```

```
window.mainloop()
```

The following is the output:

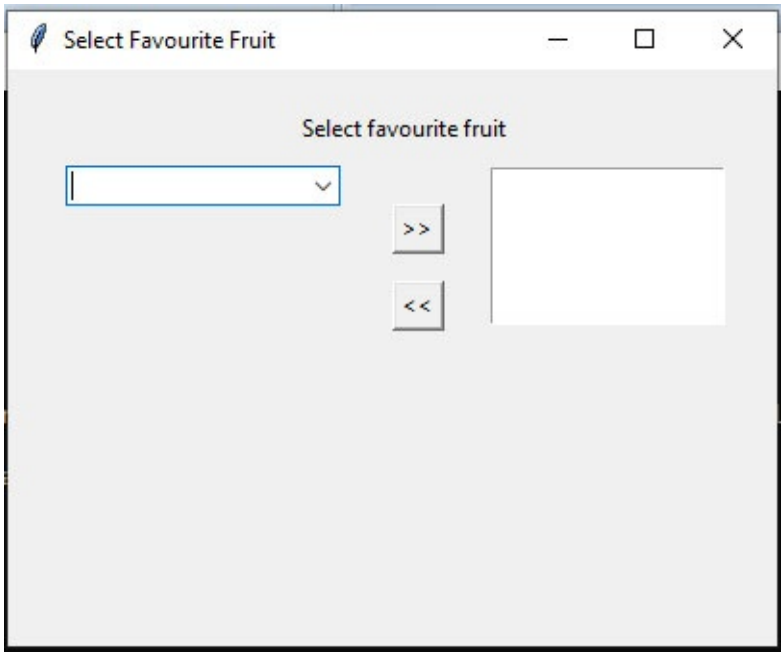


Figure 26

78. **Is indentation required in Python?**

    Ans. Indentation is necessary for Python. It specifies a block of code. All code within loops, classes, functions, etc. is specified within an indented block. It is usually done using four space characters. If your code is not indented necessarily, it will not execute accurately and will throw errors as well.

79. **How is Multithreading achieved in Python?**

    Ans.

    - Python has a multi-threading package but if you want to multi-thread to speedyour code up, then it's usually not a good idea to use it.
    - Python has a construct called the **Global Interpreter Lock (GIL).** The GIL makes sure that only one of your 'threads' can execute at any one time. A thread acquires the GIL, does a little work, then passes the GIL onto the next thread.
    - This happens very quickly so to the human eye, it may seem like your threads are executing in parallel, but they are just taking turns using the same CPU core.

- All this GIL passing adds overhead to execution. This means that if you want to make your code run faster, then using the threading package often isn't a good idea.

80. **Write some disadvantages of multi-threading? (Multi-threading is a concept and same in all language)**

    Ans.

    - On a single processor system, multithreading won't hit the speed of computation. The performance may downgrade due to the overhead of managing threads.
    - Synchronization is needed to prevent mutual exclusion while accessing shared resources. It directly leads to more memory and CPU utilization.
    - Multithreading increases the complexity of the program, thus also making it difficult to debug.
    - It raises the possibility of potential deadlocks.
    - It may cause starvation when a thread doesn't get regular access to shared resources. The application would then fail to resume its work.

81. **Write a program to create a popup menu?**

    Ans.

```
from tkinter import *

top = Tk()

w = Label(top, text="Right-click to display menu", width=40,
height=20)
w.pack()

# create a menu
popup = Menu(top, tearoff=0)
popup.add_command(label="Cut")
popup.add_command(label="Copy")
popup.add_command(label="Paste")
popup.add_separator()
popup.add_command(label="Select All")
popup.add_command(label="Delete")

def do_popup(event):
 # display the popup menu
try:
   popup.tk_popup(event.x_root, event.y_root, 0)
```

```
finally:
   popup.grab_release()
w.bind("<Button 3>", do_popup)

mainloop()
```

Take a look at the following output screen:

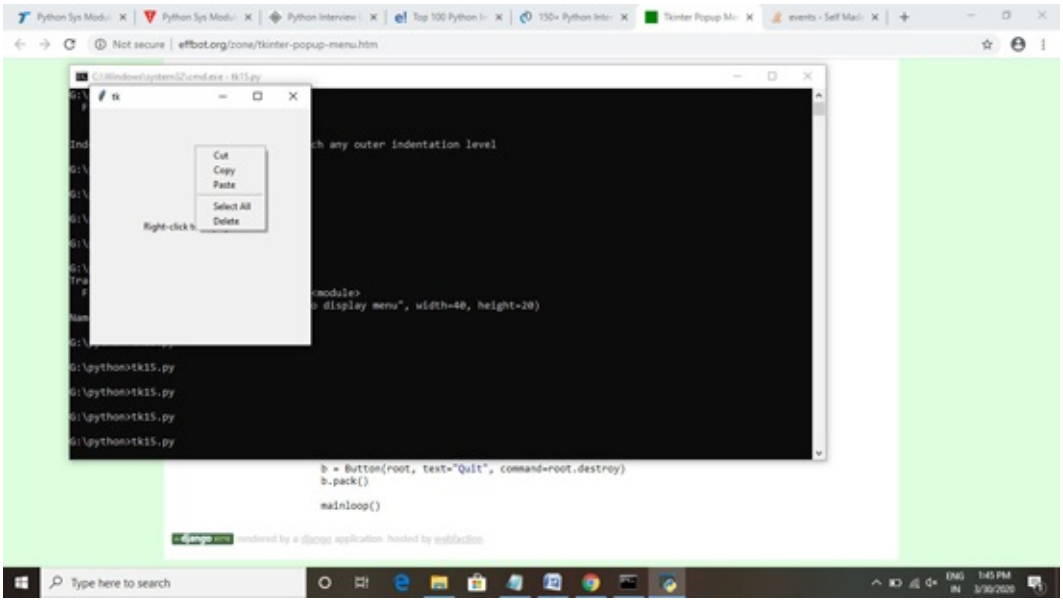


Figure 27

82. **What is the Dogpile effect?**

Ans. The Dogpile effect is referred to as the event when the cache expires, and websites are hit by the multiple requests made by the client at the same time. This effect can be prevented by using a semaphore lock. In this system when value expires, the first process acquires the lock and starts generating new value.

83. **What is a Frame in Python GUI?**

Ans. A Frame in Python can be related as a storage holder for other Graphical User Interface or GUI elements such as Label, Text Entry, Text Box, Check Button, RadioButton, etc. The following is the output screen:

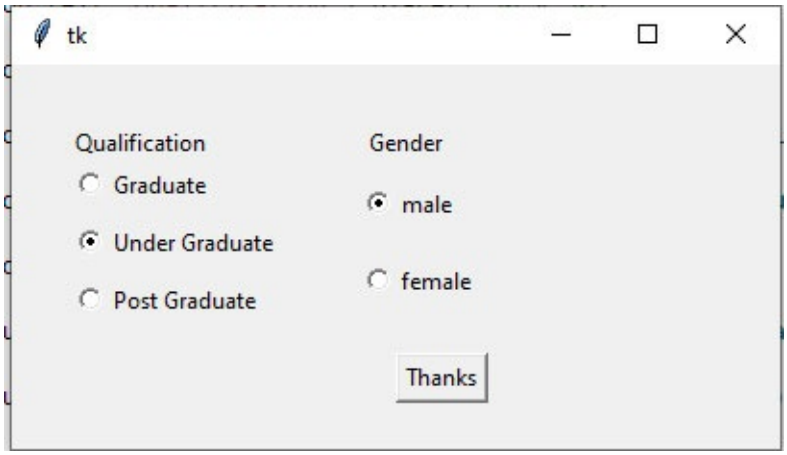


Figure 28

84. **What is the difference between the input() method and the raw_input() method?**

Ans. `Theraw_input()` method returns string values whereas the `input()` method returns integer values. The `input()`method was used in Python 2.x versions whereas Python 3.x and later versions use `theraw_input()` method. However, the `input()` method has been replaced by the `raw_input()` method in Python 3.x.

85. **What is the difference between a Lambda and Def?**

Ans. A `Def` is a function that can contain multiple expressions whereas a Lambda can contain only one single expression. A `Def` method can contain `return` statements whereas a Lambda cannot contain `return` statements. A Lambda can be used inside lists and dictionaries.

Example of Lambda:

```
 x = lambda a: a + 10
 print(x(5))
```

Example of def:

```
def addi(answer, n1, n2):
 num1 = (n1.get())
 num2 = (n2.get())
result = int(num1)+int(num2)
answer.config(text="Result = %d" % result)
return
```

86. **How do you define the dimensions of a window in a Python Graphics Program?**

Ans. It can be defined using the `geometry()` method. It takes in two parameters: `width` and `height` respectively.

Example: `geometry("width * height")`

87. **What is a DocString in Python and what is it used for?**

Ans. A DocString represents Document String that is used to Document Python Modules, Classes, and Methods.

88. **How to add an image in Tkinter?**

Ans.

```
from tkinter import *

from PIL import Image, ImageTk

class Window(Frame):

def __init__(self, master=None):
   Frame.__init__(self, master)

   self.master = master
self.pack(fill=BOTH, expand=1)

load = Image.open("1.jpg")
render = ImageTk.PhotoImage(load)
img = Label(self, image=render)
   img.image = render
   img.place(x=0, y=0)

root = Tk()
app = Window(root)
root.wm_title("Tkinter window")
root.geometry("400x420")
root.mainloop()
```

The following is the output:

Figure 29

89. **List any six Python Library.**

    Ans. Here are six Python Library:

    - `Pandas`
    - `Numpy`
    - `Keras`
    - `Eli5`
    - `SciPy`
    - `Theano`

90. **If someone wants to halt the program flow and display text, how to add a delay in execution?**

    Ans. The Sleep function plays a very important role in a situation where we want to halt the program flow and let other executions happen. It basically adds a delay to the execution and it will only pause the current thread and not the whole program:

```
import time
sleep_time = 1
print('Hello')
time.sleep(sleep_time)
```

```
print(“Students!”)
time.sleep(sleep_time)
print(“Hope you all are doing well!”)
```

91. **Explain threads in Python.**

Ans. Threads are a single execution part of a program. Several threads can run concurrently, they share the same memory space and are lightweight processes and, they do not require much memory space. It can temporarily be put on hold (also known as **sleeping**) while other threads are running (called **yielding**).

The methods provided by the Thread class are as follows:

- `run()`: The `run()` method is the entry point for a thread.
- `start()`: The `start()` method starts a thread by calling the run method.
- `join([time])`: The `join()` waits for threads to terminate.
- `isAlive()`: The `isAlive()` method checks whether a thread is still executing.
- `getName()`: The `getName()` method returns the name of a thread.
- `setName()`: The `setName()` method sets the name of a thread.

Example:

```
import threading
import time

exitFlag = 0

class myThread (threading.Thread):

def __init__(self, threadID, name, counter):
   threading.Thread.__init__(self)
   self.threadID = threadID
   self.name = name
   self.counter = counter

def run(self):
print (“Starting “ + self.name)
   print_time(self.name, 5, self.counter)
print (“Exiting “ + self.name)

def print_time(threadName, counter, delay):

while counter:
```

```
if exitFlag:
threadName.exit()
time.sleep(delay)
print (“%s: %s” % (threadName, time.ctime(time.time())))
counter -= 1

# Create new threads
thread1 = myThread(1, “First”, 1)
thread2 = myThread(2, “Second”, 2)

# Start new Threads
thread1.start()
thread2.start()
```

92. **Write a program to check whether a phone number is valid or not using regX? It should contain only ten digits.**

Ans.

```
import re

phone = “1234567890”

if re.search(‘\d{10}’, phone):
print (“Phone Num : “,phone)
else:
print(“invalid”)
```

Take a look at the following output:

```
import re

phone = "1234567890"
#phone = "123567890"


if re.search('\d{10}',  phone):
    print ("Phone Num : ",phone)
else:
    print("invalid")


Phone Num :  1234567890
```

Figure 30

93. **Validate e-mail using regular expression in Python.**

Ans.

```
import re
email = “swaticomputers01@gmail.com”
```

```
if re.search('^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$',
email):
print("Correct")
else:
print("invalid")
```

The output will be:

```
Correct
```

94. **What is Jython?**

Ans. Jython is a Java implementation of Python. Jython is freely available for both commercial and non-commercial use and is distributed with source code under the PSF License v2.

95. **Why Jython?**

Ans.

- **Rapid application development:** Python programs are typically shorter than the equivalent Java program. This translates directly to increased programmer productivity. The seamless interaction between Python and Java allows developers to freely mix the two languages both during development.
- **Embedded scripting:** Java programmers can add the Jython libraries to their system to allow end-users to write simple or complicated scripts that add functionality to the application.
- **Interactive experimentation:** Jython provides an interactive interpreter that can be used to interact with Java packages or with running Java applications.

Some of the projects where Jython is embedded are TigerJython, IBM Websphere, Apache PIG, etc. To embed Python code into Java, you need to import required libraries. Download https://www.jython.org/download and work on it.

96. **What is wxPython?**

Ans. wxPython is a cross-platform GUI toolkit for the Python programming language. It allows Python programmers to create programs with a robust, highly functional graphical user interface, simply, and easily. wxPython is an Open source and a cross-platform toolkit. This means that the same program will run on multiple platforms without modification. Currently, supported platforms are Microsoft Windows, Mac OS X and

macOS, and Linux or other Unix-like systems with GTK2 or GTK3 libraries.

97. **What is PyCharm and wxPython?**/p>

    Ans. Pycharm is an editor and integrated development environment by Jetbrains.com. Wxpython is a GUI library used for making GUI apps written in Python/C++.

98. **List some GUI frameworks for Python?**

    Ans.

Tkinter	Tkinter is commonly bundled with Python, using Tk and is Python's standard GUI framework. It is popular for its simplicity and graphical user interface. It is open source and available under the Python License.
wxPython	WxPython is an open-source wrapper for cross-platform GUI library WxWidgets (earlier known as WxWindows) and implemented as a Python extension module. With WxPython, you as a developer can create native applications for Windows, Mac OS, and Unix.
PyGUI	PyGUI is a graphical application cross-platform framework for Unix, Macintosh, and Windows. It is simple and lightweight.
PySide	PySide is a free and cross-platform GUI toolkit. PySide currently supports Linux/X11, Mac OS X, Maemo, and Windows, and support for Android is in the plans for the near future. PySide provides tools to works with multimedia, XML documents, network, databases, and GUI.
PyQt	PyQt is a cross-platform Python GUI Framework implementing the Qt library for the Qt (owned by Nokia) application development framework. Currently, PyQT is available for Unix/Linux, Windows, Mac OS X, and Sharp Zaurus.
Kivy	Kivy is an OpenGL ES 2 accelerated framework for the creation of new user interfaces. It supports multiple platforms namely Windows, MacOSX, Linux, Android-iOS, and Raspberry Pi. It is an open source.

Table 6

    Qt initiated and sponsored by Nokia, Qt is a UI framework and a cross-platform application.

99. **What is the need for File Handling?**

    Ans. Sometimes, data may be large enough and we need to store for future reference. Then we use the local file system to store data so that it can be accessed in the future. Just like in C and other programming languages you can create, read, or write files.

Syntax:

```
file_object = open(file_name [, access_mode][, buffering])
```

100. **Explain the File access mode.**

Ans.

SN	Access mode	Description
1	`R`	It opens the file to read-only. The file pointer exists at the beginning.
2	`Rb`	It opens the file to read-only in the binary format.
3	`r+`	It opens the file to read and write both.
4	`rb+`	It opens the file to read and write both in the binary format.
5	`W`	It opens the file to write only. It overwrites the file if previously exists or creates a new one if no file exists with the same name.
6	`Wb`	It opens the file to write only in the binary format
7	`w+`	It opens the file to write and read both. It is different from `r+` in the sense that it overwrites the previous file if one exists whereas `r+` doesn't overwrite the previously written file.
8	`wb+`	It opens the file to write and read both in the binary format.
9	`A`	It opens the file in the append mode.
10	`Ab`	It opens the file in the append mode in the binary format.
11	`a+`	It opens a file to append and read both.
12	`ab+`	It opens a file to append and read both in the binary format.

Table 7

101. **Write a program to read a line from an existing file?**

Ans.

```
# Open a file
fo = open("tk21.py", "r")

str = fo.read(10);
print ("Read String is : ", str)

fo.close()
```

Take a look at the following screenshot:

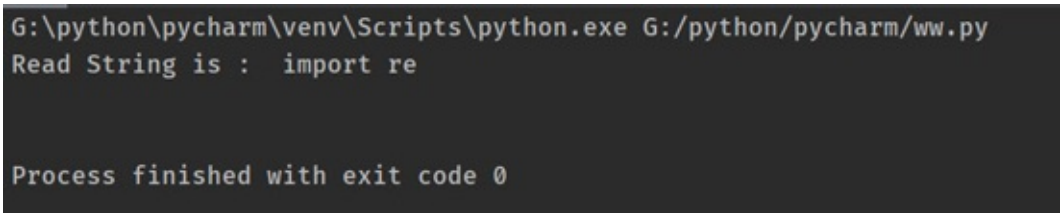


Figure 31

102. **Write a program to read the complete file?**

Ans.

```
# Open a file
fo = open(“tk21.py”, “r”)
str = fo.read(10)
while (str):
print (str)
str = fo.read(10)

fo.close()
```

The output is as follows:

```
G:\python\pycharm\venv\Scripts\python.exe  G:/
python/pycharm/ww.py
import re

phone = “
swaticompu
ters01@gma
il.com”

#
 Remove an
ything oth
er than di
gits
if re
.search(‘^
\w+([\.-]?
\w+)*@\w+(
[\.-]?\w+)
*(\.\w{2,3
})+$’, pho
ne):
p
rint(“Phon
e Num : “,
phone)
el
se:
pr
int(“inval
id”)

Process finished with exit code 0
```

103. **Write a program to read five names from the console and write it into the file?**

Ans.

```
# Open a file
```

```
fo = open(“name.txt”, “w”)

for i in range (1,6):

print (“Enter name”)
name=input()
fo.write(“Name\t”+name+”\n”)

# Close opend file
fo.close()
```

The output is as follows:

```
Name Swati
Name Sweety
Name Shweta
Name Nidhi
Name Aavya
```

104. **Which Python module is used to create a new folder, removing, and renaming the file?**

Ans. The Python `os` module is used. The `os` module in Python provides functions for interacting with the operating system. `os` comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality.

```
import os
print(os.name)
```

```
nt
```

Figure 32

105. **Write the output of an existing file into another file without using read/write operations of file?**

Ans. Here we are writing the output of `tk21.py` into `output.txt`. The Python code is in `tk24.py`:

```
Tk21.py
import re

email = “swaticomputers01@gmail.com”

if re.search(‘^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$’,
email):
print(“Email : “, email)
else:
print(“invalid”)
```

```
tk24.py
import subprocess

with open(“output.txt”, “wb”) as f:
 subprocess.check_call([“python”, “tk21.py”], stdout=f)
```

Note: Difference between “open” and “with open” ?

The open method syntax:

```
file_handle=open(“name”,”mode”)

with open syntax:
with open (“name of file”, “mode”) as keyword :
……
…….
```

Open methods take two arguments and you have to explicitly close the file after read/write operation, if you forget to close the garbage collector will close, but we are not sure about this. While with open will close, the file itself after the with block close:

```
output.txt
Email: swaticomputers01@gmail.com
```

106. **Create a GUI form (Tkinter), enter user name and password, and save them into a file.**

Ans.

```
from tkinter import *
from tkinter.colorchooser import *
from functools import partial

top = Tk()

def xx(a, b):
 u = a.get()
 p = b.get()
 f = open(“users.txt”, “a”)
f.write(“Uname: “ + u + “\tPwd: “ + p + “\n”)
f.close()
a.set(“”)
b.set(“”)

top.title(“Login Box”)
top.geometry(“300x200+200+200”)
top.configure(bg=’blue’)

uname = Label(top, text=”Username”, fg=’white’,
```

```
bg='blue').place(x=30, y=50)

a = StringVar()
b = StringVar()

password = Label(top, text="Password", fg='white',
bg='blue').place(x=30, y=90)

e1 = Entry(top, width=20, textvariable=a).place(x=100, y=50)
e2 = Entry(top, width=20, textvariable=b).place(x=100, y=90)

xx = partial(xx, a, b)
sbmitbtn = Button(top, text="Submit", bg="white", fg="blue",
command=xx).place(x=100, y=120)

top.mainloop()
```

107. **Explain the function of the tell() method.**

Ans.

- It tells the current position within the file.
- This indicates that the next read or write will occur at that many bytes from the beginning of the file.

108. **Explain the function of the seek() method.**

Ans. Move the current file position to a different location at a defined offset.

109. **What will be the output of the following code?**

Ans.

```
fo = open("myfile.txt", "w+")
print ("Name of the file: ", fo.name)

# Assuming that the file contains these lines
# PythonQuestionnair
# Hello Viewers!!

seq=" PythonQuestionnair \nHello Viewers!!"
fo.writelines(seq)

fo.seek(0,0)

for line in fo:
print (line)

fo.close()
```

The output will be as follows:

```
Name of the file: myfile.txt
PythonQuestionnair
 Hello Viewers!!
```

110. **What is pickling in Python?**

Ans.

- It is a process to convert a Python object into a byte stream.
- It is done using two methods dump and load.
- Serialization is an alternate name for pickling.

111. **Guess output of the following.**

```
with open("hello.txt", "w") as f:
 f.write("Good Morning Students")
```

```
with open('hello.txt', 'r') as f:
 data = f.readlines()
 for line in data:
  words = line.split()
 print (words)
 f.close()
```

Ans. The output is as follows:

```
['Good', 'Morning', 'Students']
```

112. **Create a table "friends" with "name, phone" column and enter value in it.**

Ans.

```
import mysql.connector

# databse name is "db" contains "friends" table

mydb = mysql.connector.connect(
host="localhost",
user="root",
password="computers ",
database="db"
)

mycursor = mydb.cursor()
mycursor.execute("create table friends(fname varchar(20),phone
Bigint(10))")

n = input("Enter nameof friend")
p = int(input("phone no"))
```

```
sql = “insert into friends(fname,phone) values(%s,%s)”
val = (n, p)

mycursor.execute(sql,val)
mydb.commit()

mydb.close()
```

Table:

Fname	Phone
Swati	1234567899

Table 8

113. **In the preceding example, add another column birthday after the phone column.**

Ans.

```
import mysql.connector
mydb = mysql.connector.connect(
host=”localhost”,
user=”root”,
password=”computers “,
database=”db”
)

mycursor = mydb.cursor()
mycursor.execute(“alter table friends add column bday date”)

mydb.commit()
mydb.close()
```

Fname	Phone	Bday
Swati	1234567899	

Table 9

114. **Update the preceding table and set Bday of a friend.**

Ans.

```
import mysql.connector

mydb = mysql.connector.connect(
host=”localhost”,
user=”root”,
password=”computers “,
database=”db”
```

```
)

mycursor = mydb.cursor()

n = input(“Enter nameof friend”)
dt: str = input(“Enter date of birth”)

sql = “update friends set bday=%s where fname=%s”
val = (dt,n)

mycursor.execute(sql, val)
mydb.commit()
mydb.close()

# Create a databse and table and run above code for output.
```

115. **Delete the desired record from the preceding table.**

Ans.

```
import mysql.connector

mydb = mysql.connector.connect(
host=”localhost”,
user=”root”,
password=”computers “,
database=”db”
)

mycursor = mydb.cursor()

n = input(“Enter nameof friend”)
sql = “delete from friends where fname=%s”
val = (n,)

mydb.commit()
mydb.close()

# Create a databse and table and run above code for output.
```

116. **Create table users in MySQL with uname and pwd fields. Create a login box in Python GUI, user may signup, signin, change password, delete account on button click.**

Ans.

```
from functools import partial
from tkinter import *
from tkinter import messagebox

import mysql.connector

mydb = mysql.connector.connect(
```

```
host=”localhost”,
user=”root”,
password=”computers”,
database=”db”
)

mycursor = mydb.cursor()

top = Tk()

top.title(“Login Box”)
top.geometry(“400x300+200+200”)
top.configure(bg=’pink’)

def ins(un, pw):
 a = un.get()
 b = pw.get()

sql = “insert into users(uname,pwd) values(%s,%s)”
val = (a, b)

mycursor.execute(sql, val)
messagebox.showinfo(“Message”, ‘Account Created’)
mydb.commit()

un.set(“”)

pw.set(“”)

def login(un,pw):
 a = un.get()
 b = pw.get()

sql = “select * from users where uname=%s and pwd=%s”
val = (a, b)
mycursor.execute(sql, val)

result=mycursor.fetchall()
if result:
messagebox.showinfo(“Message”,’successfully logIn into
account’)
else:
messagebox.showerror(“Message”,”Try Again!”)

un.set(“”)
pw.set(“”)

def dels(un,pw):
 a = un.get()
 b = pw.get()

sql = “delete from users where uname=%s and pwd=%s”
```

```
val = (a, b)
mycursor.execute(sql, val)

messagebox.showinfo(“Message”,’Account Deleted’)

mydb.commit()
un.set(“”)
pw.set(“”)

def chngP(un, pw,cpw,npw):
 a = un.get()
 b = pw.get()

sql = “update users set pwd=%s where uname=%s and pwd=%s”
val = (cpw.get(), a, b)

if cpw.get() == npw.get() :
mycursor.execute(sql, val)
messagebox.showinfo(“Message”, ‘Password Updated’)
mydb.commit()

else:
messagebox.showerror(“Message”,”password does not match”)

un.set(“”)
pw.set(“”)
npw.set(“”)
cpw.set(“”)

# creating label

uname = Label(top, text=”Username”, fg=’blue’,
bg=’pink’).place(x=30, y=50)
password = Label(top, text=”Password”, bg=’pink’,
fg=’blue’).place(x=30, y=90)
Confirm_password = Label(top, text=”New Password”, bg=’pink’,
fg=’blue’).place(x=30, y=190)
New_password = Label(top, text=”Confirm Password”, bg=’pink’,
fg=’blue’).place(x=30, y=230)

# creating text box
un =StringVar()
pw = StringVar()
cpw = StringVar()
npw = StringVar()

e1 = Entry(top, width=30, textvariable=un).place(x=170, y=50)
e2 = Entry(top, width=30, textvariable=pw).place(x=170, y=90)
e3 = Entry(top, width=30, textvariable=npw).place(x=170, y=190)
e4 = Entry(top, width=30, textvariable=cpw).place(x=170, y=230)
```

```
ins=partial(ins,un, pw)
login=partial(login,un,pw)
dels=partial(dels,un,pw)
chngP=partial(chngP,un,pw,npw,cpw)

# creating button
insertbtn = Button(top, text="SignUp", bg="blue", fg="pink",
width="10", command=ins).place(x=50, y=140)
signinbtn = Button(top, text="SignIn", bg="blue", fg="pink",
width="10",command=login).place(x=140, y=140)
updatebtn = Button(top, text="Change Password", bg="blue",
fg="pink", width="15",command=chngP).place(x=80, y=260)
deletebtn = Button(top, text="Delete Record", bg="blue",
fg="pink", width="15",command=dels).place(x=230, y=140)
top.mainloop()
```

The following screen will be visible:

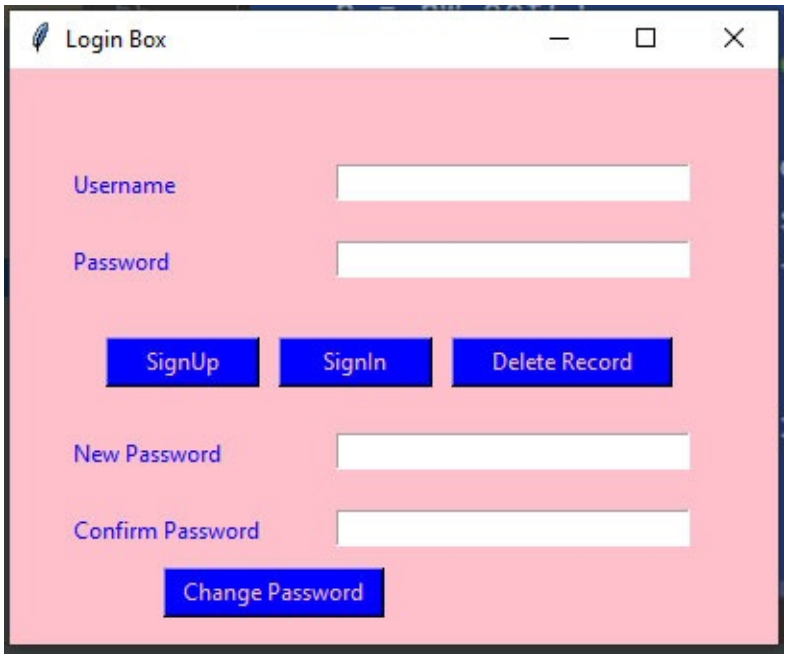


Figure 33

117. **Create a GUI form to insert the employee name and generate employee ID.**

Ans.

```
from functools import partial
from tkinter import *
import tkinter as tk
from tkinter import messagebox
import mysql.connector
```

```
top = Tk()
top.title(“Generate Id “)
top.geometry(“300x150+200+200”)
top.config(bg=”lightgray”)

mydb = mysql.connector.connect(
host=”localhost”,
user=”root”,
password=”1234”,
database=”db”
)

mycursor = mydb.cursor()
def generate(name):
 a = IntVar

mycursor.execute(“select max(eid) from employee”)
result = mycursor.fetchone()
 a = result[0] + 1
messagebox.showinfo(“Id:”,a)

sql = “insert into employee(eid,name) values(%s,%s)”
val = (a, name.get())

mycursor.execute(sql, val)
messagebox.showinfo(“Message”, ‘Id Generated’)
mydb.commit()

name = StringVar()

nam = Label(top, text=”Enter your Name..”, bg=”lightgray”,
fg=”black”).place(x=50, y=30)
e1 = Entry(top, textvariable=name, width=”30”).place(x=70,
y=50)

#lid = Label(top, text=” Please Note your ID : “, fg=’black’,

bg=’lightgray’).place(x=50, y=160)
#answer = tk.Label(top, fg=’black’).place(x=200, y=160)

generate = partial(generate, name)

butt1 = Button(top, text=”Generate ID”, width=”30”, bg=”black”,
fg=”white”, command=generate).place(x=40, y=100)
top.mainloop()
```

The following is the screenshot:

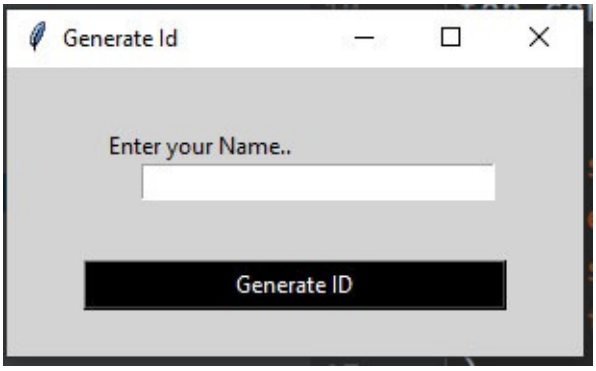


Figure 34

118. **What is SQLite?**

Ans. Like other DBMS SQLite is an RDBMS and light-weighted. It is designed by *D.Richard Hipp* for the purpose of no administration required for operating a program. SQLite is available on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT). SQLite doesn't require configuration.

119. **What is the difference between SQL and SQLite?**

Ans. The main differences between SQL and SQLite are:

- SQL is Structured Query Language while SQLite is a relational database management system mostly used in android mobile devices to store data.
- SQL support stored procedures while SQLite does not support stored procedures.
- SQL is server-based, while SQLite is file-based.

120. **What is the use of the LIMIT clause with the SELECT query?**

Ans. The `LIMIT` clause is used with the `SELECT` statement when we want a limited number of fetched records.

Syntax:

```
SELECT column1, column2, columnN
FROM table_name
LIMIT [no of rows]
```

Example:

```
SELECT * FROM STUDENT LIMIT 5; #it will return five records
from student table
```

```
OFFSET is used in some cases where we have to retrieve the
records starting from a certain point.
Select 3 records form table “STUDENT” starting from 4th
position.
SELECT * FROM STUDENT LIMIT 3 OFFSET 3;
```

121. **What is the difference between UNION and UNION ALL operator?**

Ans. The `UNION ALL` operator is used to combine the result of two or more tables using the `SELECT` statement. The `UNION` operator ignores the duplicate entries while combining the results, while `UNION ALL` doesn’t ignore duplicate values.

Syntax:

```
SELECT expression1, expression2, … expression_n
FROM tables
[WHERE conditions]
UNION ALL
SELECT expression1, expression2, … expression_n
FROM tables
[WHERE conditions];
```

122. **What is SQLite JOIN? How many types of JOINS are supported in SQLite?**

Ans. The SQLite `JOIN` clause is used to combine two or more tables in a database. It combines the table by using the common values of both tables. There are mainly three types of joins supported in SQLite:

- **SQLite INNER JOIN:** SQLite INNER JOIN is the simplest and most common join. It combines all rows from both tables where the condition is satisfied.

  Syntax:

  ```
  SELECT … FROM table1 [INNER] JOIN table2 ON
  conditional_expression …
  ```

  Example:

  ```
  SELECT EMP_ID, NAME, DEPT FROM EMP INNER JOIN INCR
  ON EMP.ID = INCR.EMP_ID;
  ```

- **SQLite OUTER JOIN:** The SQLite left outer join is used to fetch all rows from the left-hand table specified in the `ON` condition and only those rows from the right table where the join condition is satisfied.

Syntax:

```
SELECT … FROM table1 LEFT OUTER JOIN table2 ON
conditional_expression
```

Example:

```
SELECT EMP_ID, NAME, DEPT FROM EMP LEFT OUTER JOIN INCR
ON EMP.ID = INCR.EMP_ID;
```

- **SQLite CROSS JOIN:** The SQLite Cross join is used to match every rows of the first table with every rows of the second table.

Syntax:

```
SELECT … FROM table1 CROSS JOIN table2
```

Example:

```
SELECT * FROM TEAM1 CROSS JOIN TEAM2;
```

123. **What is the SQLite Julianday function?**

Ans. A Julian Day is the number of days since Nov 24, 4714 BC 12:00pm Greenwich time in the Gregorian calendar. The `julianday` function returns the date as a floating-point number.

Example:

```
SELECT julianday('2020-01-10 10:55:20');
SELECT julianday('now');
```

124. **What is Primary key and Foreign Key in SQLite?**

Ans. The SQLite primary key is a simple field or a combination of fields that are used to uniquely define a record. A table can have only one primary key. A primary key should not be a NULL value. SQLite Foreign Key is used to specify that values in one table also appear in another table. It enforces referential integrity within the SQLite database. The referenced table is known as the parent table while the table with the foreign key is known as the child table. The foreign key in the child table will generally reference a primary key in the parent table.

125. **What is the NoSQL database?**

Ans. NoSQL stands for "Not Only SQL". NoSQL is a type of database that can handle and sort all types of unstructured, messy, and complicated data.

It is just a new way to think about the database. NoSQL is an open-source database. It has no approach for the backup of data. It has some negative points such as no GUI, no Backup, Open source, and has no reliable standard.

126. **What is Redis?**

Ans. Redis is a NoSQL database that follows the principle of key-value stores. Redis is a NoSQL database so it facilitates users to store huge amounts of data without the limitations of a Relational database. Redis supports various types of data structures such as strings, hashes, lists, sets, sorted sets, bitmaps, hyperlogs, and geospatial indexes with radius queries.

127. **What is the difference between Redis and RDBMS?**

Ans. The following table shows the difference:

Redis	RDBMS
Redis stores everything in primary memory.	RDBMS stores everything in secondary memory.
In Redis, Read and Write operations are extremely fast because of storing data in primary memory.	In RDBMS, Read and Write operations are slow because of storing data in secondary memory.
Primary memory is lesser in size and much expensive than secondary. Therefore, Redis cannot store large files or binary data.	Secondary memory is abundant in size and is cheaper than primary memory. Therefore, RDBMS can easily deal with these types of files.
Redis is used only to store that small textual information that needs to be accessed, modified, and inserted at a very fast rate. If you try to write bulk data more than the available memory, then you will receive errors.	RDBMS can hold large data that has less frequent usage and not required to be very fast.

Table 10

128. **What is the difference between MongoDB and Redis database?**

Ans. The following is the difference:

Comparison Index	Redis	MongoDB
Introduction	Redis is an in-memory data structure store, used as a database, cache, and message broker.	MongoDB is one of the most popular NoSQL databases that follow the document stores structure.
Primary database model	Redis follows the key-value	MongoDB follows the

	store model.	document store model.
Official Website	redis.io	**www.mongodb.com**
Developed By	Redis is developed by Salvatore Sanfilippo.	MongoDB is developed by MongoDB Inc.
Initial Release	Redis is initially released in 2009.	MongoDB is also initially released in 2009.
Cloud-based	No	No
Implementation Language	Redis is written and implemented in the C language.	MongoDB is written and implemented in the C++ language.
Server operating systems	BSD, Linux, OS X, Windows	Linux, OS X, Solaris, Windows
Data Scheme	schema-free	schema-free
Secondary Indexes	No	Yes
SQL	No	No
APIs and other access methods	Redis follows a proprietary protocol.	MongoDB follows a proprietary protocol using JSON.
Supported programming languages	C, C#, C++, Clojure, Crystal, D, Dart, Elixir, Erlang, Fancy, Go, Haskell, Haxe, Java, JavaScript (Node.js), Lisp, Lua, MatLab, Objective-C, OCaml, Perl, PHP, Prolog, Pure Data, Python, R, Rebol, Ruby, Rust, Scala, Scheme, Smalltalk, Tcl	Actionscript, C, C#, C++, Clojure, ColdFusion, D, Dart, Delphi, Erlang, Go, Groovy, Haskell, Java, JavaScript, Lisp, Lua, MatLab Perl, PHP, PowerShell, Prolog, Python, R, Ruby, Scala, Smalltalk
Server-side scripts	Lua	JavaScript
Triggers	No	No
Partitioning methods	Redis uses Sharding for partition.	MongoDB also uses Sharding for partition.
Replication methods	Redis follows master-slave replication.	MongoDB also follows master-slave replication.
MapReduce	No	Yes
Consistency concepts	Eventual Consistency and Immediate Consistency	Eventual Consistency
Foreign keys	No	No

Transaction concepts	Optimistic locking, atomic execution of commands blocks, and scripts.	No
Concurrency	Yes	Yes
MapReduce	No	Yes
Durability	Yes	Yes
In-memory capabilities	Yes	Yes
User concepts	Simple password-based access control.	Access rights for users and roles.
Special Characteristics	Redis is ranked as the world's fastest database. It reduces application complexity, simplifies development, accelerates time to market, and provides unprecedented flexibility to developers with its visionary data structures and modules.	MongoDB is considered as the next-generation database. It successfully helped many businesses to transform their industries by providing big data. The world's most sophisticated organizations, from cutting-edge startups to the largest companies, use MongoDB to create applications never before possible, at a very low cost.
Comparing Advantages	Redis is an in-memory database platform that provides support of wide range of data structures such as strings, hashes, sets, lists, sorted sets, bitmaps, hyperloglogs, and geospatial indexes. Redis provides effortless scaling in a fully automated manner by overseeing all the operations of sharding, re-sharding, and migration. It also includes persistence, instant automatic failure detection, backup and recovery, and in-memory replication across racks, zones, datacenters, regions, and cloud platforms.	MongoDB provides the best of traditional databases as well as flexibility, scale, and performance required by today's applications. MongoDB is a database of giant ideas. MongoDB keeps the most valuable features of the Relational database, that is, strong consistency, expressive query language, and secondary indexes. It facilitates developers to build highly functional applications faster than NoSQL databases.
Key Customers	Key customers of Redis are Verizon, Vodafone, Atlassian, Trip Advisor, Jet.com, Nokia, Samsung, HTC, Docker,	Key customers of MongoDB are: ADP, Adobe, AstraZeneca, BBVA, Bosch, Cisco, CERN, Department of

	Staples, Intuit, Groupon, Shutterfly, KPMG, TD Bank, UnitedHealthcare, RingCentral, The Motley Fool, Bleacher Report, HipChat, Salesforce, Hotel Tonight, Cirruspath, Itslearning.com, Xignite, Chargify, Rumble Entertainment, Scopely, Havas Digital, Revmob, MSN, Bleacher Report, Mobli, TMZ, Klarna, Shopify, etc.	Veteran Affairs, eBay, eHarmony, Electronic Arts, Expedia, Facebook's Parse, Forbes, Foursquare, Genentech, MetLife, Pearson, Sage, Salesforce, The Weather Channel, Ticketmaster, Under Armour, Verizon Wireless, etc.

Table 11

129. **In which language MongoDB is written?**

Ans. MongoDB is written and implemented in C++.

130. **Does MongoDB database have tables for storing records?**

Ans. No. Instead of tables, MongoDB uses "Collections" to store data.

131. **What is sharding in MongoDB?**

Ans. In MongoDB, Sharding is a procedure of storing data records across multiple machines. It is a MongoDB approach to meet the demands of data growth. It creates a horizontal partition of data in a database or search engine. Each partition is referred to as a shard or database shard.

132. **Which are the different languages supported by MongoDB?**

Ans. MongoDB provides official driver support for C, C++, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala, Go, and Erlang.

133. **What type of DBMS is MongoDB?**

Ans. MongoDB is a document-oriented DBMS

134. **Does MongoDB support primary-key, foreign-key relationships?**

Ans. No. By default, MongoDB doesn't support the primary key-foreign key relationship.

135. **Is there any need to create database command in MongoDB?**

Ans. You don't need to create a database manually in MongoDB because it creates automatically when you save the value into the defined collection for the first time.

136. **In which language is Redis written?**

Ans. Redis is written in ANSI C and mostly used for cache solution and

session management. It creates unique keys for store values.

137. **What are the different languages supported by Redis?**

Ans. Redis supports a variety of languages, that is, C, C++, C#, Ruby, Python, Twisted Python, PHP, Erlang, Tcl, Perl, Lua, Java, Scala, and so on. If your favorite language is not supported yet, you can write to your client library, as the Protocol is pretty simple.

138. **What is NumPy?**

Ans. NumPy stands for numeric Python, which is a Python package for the computation and processing of the multidimensional and single-dimensional array elements. You can install NumPy by the following command:

```
Python –m pip install numpy
```

139. **What is Ndarray?**

Ans. Ndarray means an–dimensional array object defined in NumPy. It stores collection of similar datatypes.

Syntax:

```
numpy.array(object, dtype = None, copy = True, order = None,
subok = False, ndmin = 0)
```

Take a look at the following table:

Parameter	Description
Object	It represents the collection object. It can be a list, tuple, dictionary, set, and so on.
Dtype	We can change the data type of the array elements by changing this option to the specified type. The default is none.
Copy	It is optional. By default, it is true which means the object is copied.
Order	There can be threepossible values assigned to this option. It can be C (column order), R (row order), or A (any).
Subok	The returned array will be base class array by default. We can change this to make the subclasses pass through by setting this option to true.
Ndmin	It represents the minimum dimensions of the resultant array.

Table 12

140. **Give an example of Ndarray in numPy.**

Ans.

```
import numpy as np

a = np.array([[10,20],[30,40],[50,60],[70,80]])
print("printing the original array..")
print(a)

a=a.reshape(2,4)
print("printing the reshaped array..")
print(a)
```

141. **Create a numpy array using an existing tuple and list.**

Ans.

```
import numpy as np
a=(1,2,3,4)
b=[10,20,30]
c=[a,b]
d=np.array(c)
print(d)
```

142. **Print a transpose numpy array.**

Ans.

```
import numpy as np

b=[[1,2,3],[10,20,30],[100,200,300]]

d=np.array(b)
print(d)

e=d.T
print(e)
```

143. **Write a program to display the output as desired (Find output)**

```
import numpy as np
c=np.array([[1,2,3,4,5],[6,7,8,9,10]])

find output : 3,5,8,10
```

Ans. Solution:

```
print(c[[0,0,1,1],[2,4,2,4])
```

or

```
print(c[:,2::2])
```

144. **What is broadcasting? Explain with example?**

Ans. When the shapes of the two arrays are not similar and therefore they cannot be multiplied together, NumPy can perform such operations by using the concept of broadcasting. Broadcasting can be applied to the arrays if the following rules are satisfied:

- All the input arrays have the same shape.
- Arrays have the same number of dimensions, and the length of each dimension is either a common length or 1.
- Array with the fewer dimension can be appended with '1' in its shape.

Example:

```
import numpy as np

a = np.array([[1,2,3,4],[2,4,5,6],[10,20,39,3]])
b = np.array([2,4,6,8])
print("\nprinting array a..")
print(a)

print("\nprinting array b..")
print(b)

print("\nAdding arrays a and b ..")
c = a + b;
print(c)

print("\nMultiplying arrays a and b ..")
d=a*b
print(d)
```

The following image explains the preceding code in detail:

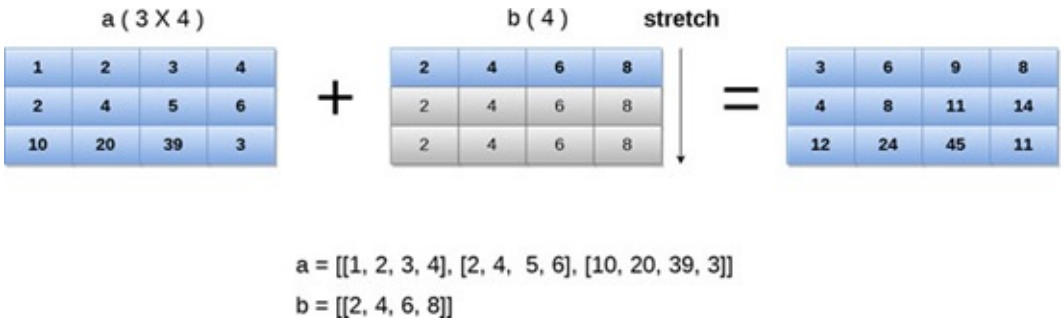


Figure 35

145. **Write a program create a numpy array using list of numbers. Display the values less than 80 otherwise 0 from numpy array?**

```
import numpy as np
```

```
a = np.array([[101,28,86,54],[22,74,95,46], [10,20,239,37]])

b=np.where(a<80,a, 0)
print(b)
```

146. **Sort array by the given age or height?**

Ans.

```
import numpy as np

dtype = [('name', 'S10'), ('height', float), ('age', int),
('gender','S10')]
values = [('Swati', 5.4, 28, 'F'), ('Aavya', 4.2, 19, 'F'),
('Nidhi', 5.8, 37, 'F'),('Deep', 6.1, 46, 'M')]

x=np.array(values, dtype=dtype)
print(x)

y=np.sort(x, order='age')
print(y)

z=np.sort(x, order=['age','height'])
print(z)
```

`dtype ()` is used to create datatype object. The output screenshot is as follows:

```
[(b'Swati', 5.4, 28, b'F') (b'Aavya', 4.2, 19, b'F')
 (b'Nidhi', 5.8, 37, b'F') (b'Deep', 6.1, 46, b'M')]
[(b'Aavya', 4.2, 19, b'F') (b'Swati', 5.4, 28, b'F')
 (b'Nidhi', 5.8, 37, b'F') (b'Deep', 6.1, 46, b'M')]
[(b'Aavya', 4.2, 19, b'F') (b'Swati', 5.4, 28, b'F')
 (b'Nidhi', 5.8, 37, b'F') (b'Deep', 6.1, 46, b'M')]
```

Figure 36

147. **Explain datatype object (dtype).**

Ans. A data type object (an instance of `numpy.dtype` class) describes how the bytes in the fixed-size block of memory corresponding to an array item should be interpreted. It describes the following aspects of the data:

- Type of the data (integer, float, Python object, to name a few)
- Size of the data (how many bytes it is in, for example, the integer)
- The byte order of the data (little-endian or big-endian)
- If the data type is a record, an aggregate of other data types, (for example, describing an array item consisting of an integer and a float)

:

- What are the names of the “fields” of the record, by which they can be accessed?
- What is the data-type of each field?
- Which part of the memory block each field takes?

148. **Explain Pandas?**

Ans. Pandas is a high-level data manipulation tool developed by *Wes McKinney.* It is built on the Numpy package and its key data structure is called DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables. It 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.

Before Pandas, Python was capable of data preparation, but it only provided limited support for data analysis. It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, for example, load, manipulate, prepare, model, and analyze. The Pandas provides two data structures for processing the data, that is, Series and DataFrame:

- **Series:** It is defined as a one-dimensional array that is capable of storing various datatypes. The row labels of series are called the index.
- **Dataframe:** It 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, that is, row index and column index.

149. **Create Series and also change its index in Python pandas.**

Ans.

```
import pandas as pd
import numpy as np

name = np.array(['s','w','a','t','i'])
a = pd.Series(name)
print(a)

a.index=["1.1","1.2","1.3","1.4","1.5"]
print(a)
```

150. **Create DataFrame from dictionary and also change its index in Python pandas.**

Ans.

```
import pandas as pd

dict = {“country”: [“India”, “Russia”, “Australia”, “United Kingdom”, “South Africa”], “area”: [9.516, 17.10, 3.286, 9.597, 12.221], “population”: [200.4, 143.5, 1252, 1357, 52.98]}

a = pd.DataFrame(dict)
print(a)

a.index=[“IN”,”RU”,”AUS”,”UK”,”SA”]
print(a)
```

151. **How to read a CSV file and perform an operation?**

Emp.CSV:

Empid,EName,Job,Salary,Deptno

101,Jerry,manager,456784,10

102,Smith,clerk,765432,20

103,Mike,admin,235678,10

104,Scott,manager,435678,10

105,James,clerk,654367,20

106,Bill,clerk,347623,10

……

Ans. Reading the preceding CSV file:

```
import pandas

df=pandas.read_csv(‘emp.csv’)
print(df)

print(“\ncount total employees”)
print(“Employees in company =”,df[‘Empid’].count())

print(“\nfind out maximum salary”)
print(“maximum salary=”,df[‘Salary’].max())

print(“\ndisplay mean, std,max,min”)
print(df.describe())

print(“\nprint departmentwise total salary”)
dg=df.groupby(‘Deptno’)
```

```
print(dg['Salary'].sum())

print("OR")
print(df.groupby('Deptno')['Salary'].sum())

print("\nPrint Name of all clerks")
print(df[df['Job']=='clerk']['EName'])

s=df['Salary'].max()

print("\nName of the employee getting highest salary")

print(df[df['Salary']==s]['EName'])
```

152. **From the preceding CSV file, find the name and department number of employees getting salary more than 50000 and less than 800000?**

Ans.

```
import pandas
df=pandas.read_csv('emp.csv')
print(df[((df['Salary']>500000)&(df['Salary'] <800000))]
[['EName','Deptno']])
```

153. **Read the following excel file and perform operations:**

Mobile.xlsx

	A	B	C	D
1	Company	Model	Color	Price
2	Nokia	6.2	black	34566
3	Samsung	Galaxy S2(	gray	97654
4	Nokia	220 4G	blue	12345
5	Xolo	ERA	black	21345
6	Nokia	7.2	white	65473
7	Redmi	Note 9 Pr(	gray	45678
8	Redmi	Note 9 Pr(	white	73455

Figure 37

Sheet Name: Detail

Ans.

```
import pandas

df = pandas.read_excel('mobile.xlsx', sheet_name='Detail')

print(df)

print("\nColumn Name:\n")
print(df.columns)
```

```
print("\nfirst three records\n")
print(df.head(3))

print("\nlast two records\n")
print(df.tail(2))

print("\nExcel to CSV\n ",df.to_csv())

print("\nFind Mobiles with price less than 60000\n")
print(df[df['Price']<60000][['Company','Model']])
```

154. **Create pivot table from excel data using pandas?**

Ans.

```
import pandas

df = pandas.read_excel('mobile.xlsx', sheet_name='Detail')

subset=df[['Company','Model','Price']]
print(subset)

print(subset.pivot_table(index='Company'))
```

155. **What kind of plots can you create in pandas?**

Ans.

- 'line': line plot (default)
- 'bar': vertical bar plot
- 'barh': horizontal bar plot
- 'hist': histogram
- 'box': boxplot
- 'kde': Kernel Density Estimation plot
- 'density': same as 'kde'
- 'area': area plot
- 'pie': pie plot
- 'scatter': scatter plot
- 'hexbin': hexbin plot

156. **Create a scatter and bar plot?**

Ans.

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
df = pd.DataFrame({
  'name':['john','scott','jerry','james',
  'bill','patrik','mike'],
  'age':[23,78,12,19,45,33,20],
  'gender':['M','F','M','M','M','F','M']
})

# a scatter plot
df.plot(kind='scatter',x='age',y='gender',color= 'red')
plt.show()

# a bar plot

df.plot(kind='bar',x='name',y='age')
plt.show()
```

The following is the output:

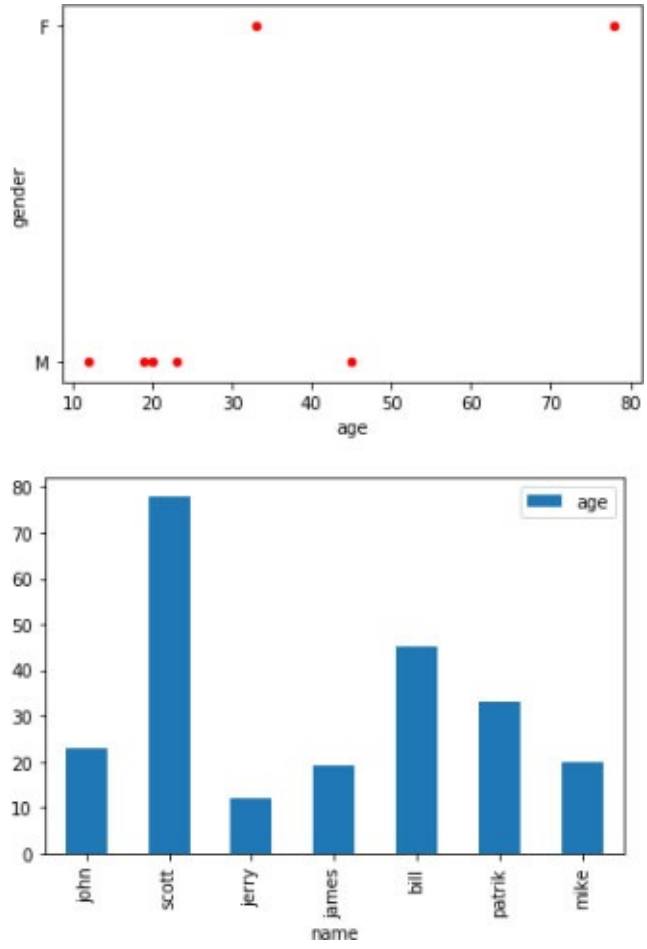


Figure 38

157. **Explain syntax of plot in pandas.**

Ans.

```
DataFrame.plot(x=None, y=None, kind='line', ax=None,
subplots=False, sharex=None, sharey=False, layout=None,
figsize=None, use_index=True, title=None, grid=None,
legend=True, style=None, logx=False, logy=False, loglog=False,
xticks=None, yticks=None, xlim=None, ylim=None, rot=None,
fontsize=None, colormap=None, table=False, yerr=None,
xerr=None, secondary_y=False, sort_columns=False, **kwds)
```

158. **Create a box plot.**

Ans.

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.rand(10, 5),
columns =['A', 'B', 'C', 'D', 'E'])

df.plot.box()
plt.show()
```

We get the following output:

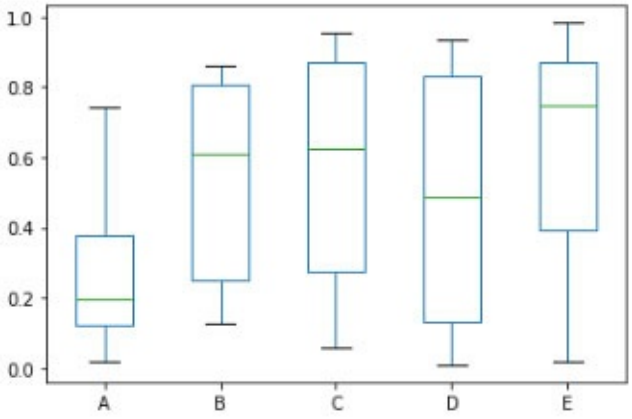


Figure 39

159. **Create a scatter plot from columns of the CSV file.**

Ans.

```
import pandas
import matplotlib.pyplot as plt
df=pandas.read_csv('emp.csv')

x=df['EName']
y=df['Salary']
plt.scatter(x,y)
plt.show()
```

160. **Create line plot on employee name and salary.**

Ans.

```
import pandas
import matplotlib.pyplot as plt
df=pandas.read_csv('emp.csv')

df.plot.line(x='EName',y='Salary')
plt.show()
```

Refer to the following output:

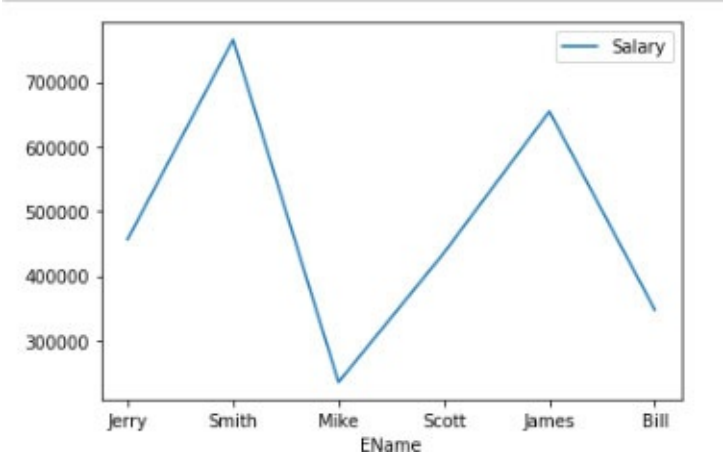


Figure 40

161. **What is the difference between loc and iloc in Pandas? Give an example.**

Ans. Using these, we can practically do any data selection task on Pandas dataframes. loc is label-based, which means that we have to specify the name of the rows and columns that we need to filter out. On the other hand, `iloc` is integer index-based. So here, we have to specify rows and columns by their integer index:

```
import pandas as pd
import numpy as np

# crete a sample dataframe

data = pd.DataFrame({
 'age' : [10, 22, 13, 21, 12, 11, 17],
 'section' : ['A', 'B', 'C', 'B', 'B', 'A', 'A'],
 'city' : ['Gurgaon', 'Delhi', 'Mumbai', 'Delhi', 'Mumbai',
 'Delhi', 'Mumbai'],
 'gender' : ['M', 'F', 'F', 'M', 'M', 'M', 'F']
})
```

```
print(data.loc[data.age >= 15])
print(data.loc[(data.age >= 12) & (data.gender == 'M')])
print(data.loc[1:3])
print("\n using iloc\n")
print(data.iloc[[0,2]])
print(data.iloc[[0,2],[1,3]])
```

The output will be as follows:

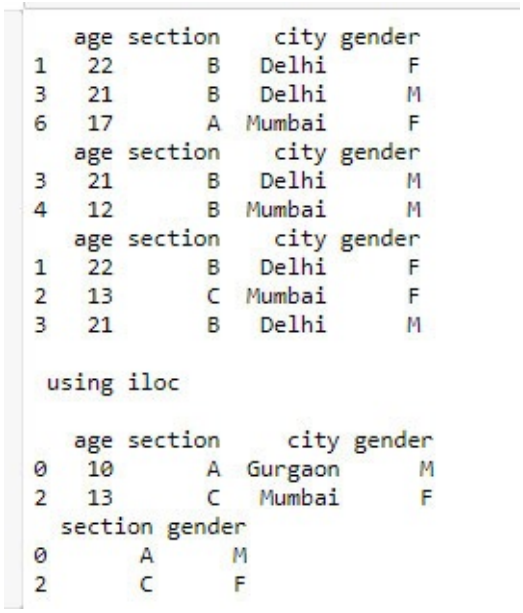


Figure 41

162. **Explain the difference between Pandas and NumPy.**

Ans. The following table shows the difference:

Basis for comparison	Pandas	NumPy
Works with	Pandas module works with the tabular data.	NumPy module works with numerical data.
Powerful Tools	Pandas have powerful tools like Series, DataFrame, and so on.	NumPy has a powerful tool like Arrays.
Organizational usage	Pandas is used in popular organizations such as Instacart, SendGrid, and Sighten.	NumPy is used in popular organizations such as SweepSouth.

Performance	Pandas have a better performance for 500K rows or more.	NumPy has a better performance for 50K rows or less.
Memory Utilization	Pandas consume large memory as compared to NumPy.	NumPy consumes less memory as compared to Pandas.

Table 13

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

Ans.

```
import pandas as pd

p1 = pd.Series([2, 4, 5, 6, 8, 9,10])
p2 = pd.Series([5, 8, 10, 12, 14, 16])

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

The output is as follows:

```
0    2
1    4
3    6
5    9
dtype: int64
```

Figure 42

164. **Create a histogram.**

Ans.

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.rand(10, 5),
columns =['A', 'B', 'C', 'D', 'E'])

df.hist()
plt.show()
```

The output will beas follows:

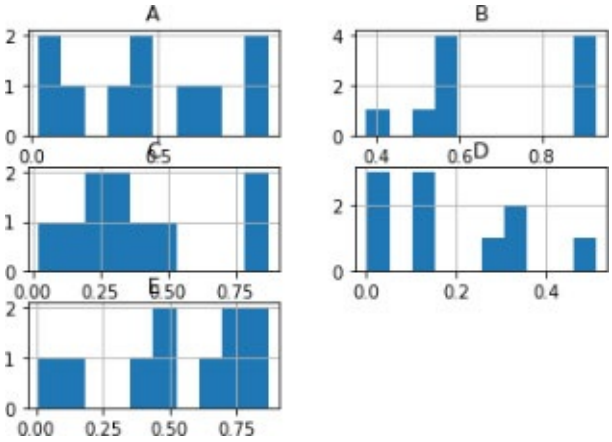


Figure 43

165. **What is a web framework?**

Ans. It is a code library that helps you to build and maintain a flexible and dynamic web application, web app, and web service. For example, Ruby on Rails for Ruby. Django is a web development framework for Python which offers a standard method for fast and effective website development.

166. **What is Django?**

Ans. Django is a web development framework written in Python. It helps you to build and maintain web applications. It is based on the **MVT (Model View Template)** design pattern. This framework uses a famous tag line: the web framework for perfectionists with deadlines.

Install Django by pip:

```
pip install django == 2.0.3
```

167. **Explain the MVT pattern.**

Ans. The **Model-View-Template (MVT)** is a different concept compared to MVC. Django itself manages the Controller part. The Model helps to handle database. It is a data access layer that handles the data. The Template is a presentation layer (HTML file) that handles the user interface part completely. The View is used to execute the business logic and interact with a model to carry data and renders a template. The following image explains the concept more clearly:

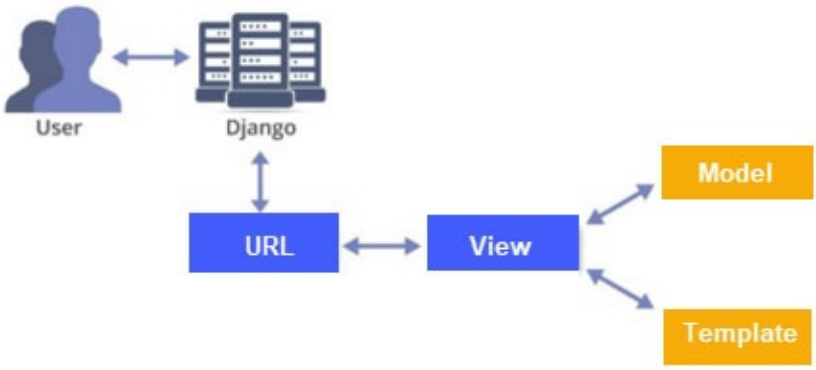


Figure 44

As seen in the preceding diagram, a user requests a resource for Django. Django acts as a controller and checks to the available resource in the URL. If URL maps, a view is called which interacts with model and template. Django then responds to the user and sends a template as a response.

168. **How to check form validity in Django?**

Ans. The `is_valid()` method is used to perform validation for each field of the form. It is defined in the Django Form class. It returns True if data is valid and place all data into a `cleaned_data` attribute.

Example:

```
def emp(request):
if request.method == “POST”:
form = EmployeeForm(request.POST)
if form.is_valid():
try:
return redirect(‘/’)
except:
pass
else:
form = EmployeeForm()
```

169. **What is Django Rest Framework (DRF)?**

Ans. Django REST is a framework that lets you create RESTful APIs rapidly. RESTful APIs are perfect for web applications since they use low bandwidth and are designed such that they work well with the communications over the Internet such as GET, POST, and PUT.

170. **What are the various files that are created when you create a Django Project? Explain briefly.**

Ans. The following table shows various files with its description:

File Name	Description
`manage.py`	A command-line utility that allows you to interact with your Django project
`__init__.py`	An empty file that tells Python that the current directory should be considered as a Python package
`settings.py`	Consists of the settings for the current project
`urls.py`	Contains the URLs for the current project
`wsgi.py`	This is an entry-point for the web servers to serve the project you have created

Table 14

171. **What is Django ORM?**

Ans. Django ORM is one of the special feature-rich tools in Django. ORM is an acronym for Object-Relational Mapper. Django ORM is the abstraction between models (web application data-structure) and the database where the data is stored. It makes it possible to retrieve, save, delete, and perform other operations over the database without ever writing any SQL code.

172. **How does Django compare to other popular frameworks such as Laravel?**

Ans. Django has many unique features. It allows for rapid development of applications without any security loopholes or performance lacking. Some more differences:

- Laravel is PHP based while Django is Pythonbased, which is clearly more powerful and robust than PHP.
- The performance of Django is better than Laravel because of the different programming languages it uses.
- Django is based on the MTV architecture, a more robust and loosely coupled architecture while Laravel is strictly based on the MVC architecture.
- Django can use RegEx while Laravel doesn’t provide you with that functionality.

173. **What is Jinja Templating?**

Ans. Django supports many popular templating engines and by default, it comes with one very powerful templating engine. Jinja Templating is a very popular templating engine for Python, the latest version in the market is Jinja 2.

174. **What is the difference between Flask and Django?**

Ans. The following table explains the difference:

Comparison Factor	Django	Flask
Project Type	Supports large projects	Built for smaller projects
Templates, Admin and ORM	Built-in	Requires installation
Ease of Learning	Requires more learning and practice	Easy to learn
Flexibility	Allows complete web development without the need for third-party tools	More flexible as the user can select any third-party tools according to their choice and requirements
Visual Debugging	Does not support Visual Debug	Supports Visual Debug
Type of framework	Batteries included	Simple, lightweight
Bootstrapping-tool	Built-it	Not available

Table 15

175. **What is the usage of Django-admin.py and manage.py?**

Ans.

- `Django-admin.py`: It is a Django's command-line utility for administrative tasks.
- `Manage.py`: It is an automatically created file in each Django project. It is a thin wrapper around the `Django-admin.py`. It has the following usage:
    - It puts your project's package on sys.path.
    - It sets the `DJANGO_SETTING_MODULE` environment variable to points to your project's `setting.py` file.

176. **Is Django stable?**

Ans. Yes, Django is quite stable. Many companies such as Instagram,

Discus, Pinterest, and Mozilla have been using Django for many years now.

177. **Is Django a CMS?**

Ans. Django is not a CMS (content-management-system). It is just a web framework, a tool that allows you to build websites.

178. **Does the Django framework scale?**

Ans. Yes. Hardware is much cheaper when compared to the development time and this is why Django is designed to make full use of any amount of hardware that you can provide it. You can add hardware at any level, that is, database servers, caching servers, or web application servers.

179. **What are the different types of Django Exception Classes?**

Ans. The `django.core.exceptions` module contains the following classes:

Exception	Description
`AppRegistryNotReady`	It is raised when attempting to use models before the app loading process.
`ObjectDoesNotExist`	The base class for `DoesNotExist` exceptions.
`EmptyResultSet`	If a query does not return any result, this exception is raised.
`FieldDoesNotExist`	It raises exception when the requested field does not exist.
`MultipleObjectsReturned`	This exception is raised by a query if only one object is expected, but multiple objects are returned.
`SuspiciousOperation`	This exception is raised when a user has performed an operation that should be considered suspicious from a security perspective.
`PermissionDenied`	It is raised when a user does not have permission to perform the action requested.
`ViewDoesNotExist`	It is raised by `django.urls` when a requested view does not exist.
`MiddlewareNotUsed`	It is raised when a middleware is not used in the server configuration.
`ImproperlyConfigured`	The `ImproperlyConfigured` exception is raised when Django is somehow improperly configured.
`FieldError`	It is raised when there is a problem with a model

	field.
`ValidationError`	It is raised when data validation fails to form or model field validation.

Table 16

180. **Give an example to set and get a cookie.**

Ans.

```
def setcookie(request):

response = HttpResponse("Cookie Set")
 response.set_cookie('python-tutorial', 'swati computers')
return response

def getcookie(request):

tutorial = request.COOKIES['python-tutorial']
return HttpResponse("python tutorials @: "+ tutorial)
```

181. **Is it mandatory to use the model/ database layer in Django?**

Ans. No, the model/ database layer is actually decoupled from the rest of the framework.

182. **What is Python Flask?**

Ans. Flask is a web framework written in Python which provides libraries to build lightweight web applications. It is based on the WSGI toolkit and jinja2 template engine. Flask is considered as a micro framework.

183. **What is WSGI and Jinja template?**

Ans. WSGI stands for web server gateway interface which is a standard for Python web application development.Jinja2 is a web template engine that combines a template with a certain data source to render the dynamic web pages.

184. **What is the difference between Django, Pyramid, and Flask?**

Ans. Take a look at the following table for understanding the difference:

Django	Pyramid	Flask
It is a Python framework.	It is the same as Django.	It is a micro framework.
It is used to build large applications.	It is the same as Django.	It is used to create small applications.
It includes an ORM.	It provides flexibility and the right tools.	It does not require external libraries.

185. **What is Flask Sijax?**

Ans. Flask Sijax is a Simple Ajax and jQuery library. It is used to enable Ajax in web applications. It uses JSON to pass data between the server and the browser.

186. **Write a code to get querystring from Flask.**

Ans.

```
from flask import request
@app.route('/data')
def data () :
model = request.arg.get ('model_no')
```

187. **Explain cookies in Flask?**

Ans. Cookies are used to track the user's activities on the web and reflect some suggestions according to the user's choices to enhance the user's experience.

In a flask, the cookies are associated with the request object as the dictionary object of all the cookie variables and their values transmitted by the client. Flask facilitates us to specify the expiry time, path, and domain name of the website.

Syntax:

```
response.setCookie(<title>, <content>, <expiry time>)
```

```
request.cookies.get(<title>)
```

188. **What is App routing and how to do it in Flask?**

Ans. App routing is used to map the specific URL with the associated function that is intended to perform some task. It is used to access some particular page:

```
@app.route('/home')
```

Flask facilitates us to add the variable part to the URL by using the section:

```
from flask import Flask
app = Flask(__name__)
```

```
@app.route('/home/<name>')
def home(name):
return "hello,"+name;
```

```
if __name__ ==”__main__”:
app.run(debug = True)
```

189. **Which method in Flask is used to give user feedback in a box like ‘alert’ in JavaScript and message’ in Tkinter?**

Ans. The `Flash ()` method is used to generate an informative message in Flask.

190. **What is containerization? What is docker?**

Ans. Docker is an open-source application that allows administrators to create, manage, deploy, and replicate applications using containers. Containers can be thought of as a package that houses dependencies that an application requires to run at an operating system level.

191. **What is web-scraping?**

Ans. If you need to pull a large amount of data from a website without going to that website, web scraping is used. It is used in the following situations: gathering e-mail ID to send bulk messages, pull out price details from an e-commerce site to compare, details regarding job openings, interviews are collected from different websites and then listed in one place so that it is easily accessible to the user.

Web scraping helps collect these unstructured data and store it in a structured form. There are different ways to scrape websites such as online Services, APIs, or writing your own code. To extract data using web scraping with Python, you need to follow these basic steps:

i. Find the URL that you want to scrape
ii. Inspecting the page
iii. Find the data you want to extract
iv. Write the code
v. Run the code and extract the data
vi. Store the data in the required format

To scrap data from Python, we have to install some libraries:

- requests
- html5lib
- bs4

You can install them from pip install in Python.

BeautifulSoup: Beautiful Soup is a Python package for parsing HTML and XML documents. It creates parse trees that are helpful to extract the data easily.

Example

```
import requests
import urllib.request
from bs4 import BeautifulSoup

url=”https://www.monster.com/jobs/search/?q=Software-
Developer&where=India”

response= requests.get(url)

#print(response)

htmlcontent=response.content

soup = BeautifulSoup(htmlcontent, “html.parser”)

#print(soup)

#print(soup.prettify)

print(soup.findAll(‘div’))
```

192. **Write a program to change the key value of the dictionary and change the attribute value.**

Ans.

```
vehicle={
 “Tw”:{“Name”:”victor”,
 “Price”:”70K”,
 “color”:”red”
 },
 “FW”:{“Name”:”kwid”,
 “Price”:”70L”,
 “color”:”blue”
 }
}
print(vehicle)

vehicle[“TV”]=vehicle.pop(“Tw”)
print(vehicle)
print(vehicle[“TV”][“color”])
vehicle[“TV”][“color”]=”silver”
print(vehicle)

#Here I have created a dictionary Vehicle with key “TW” and
```

```
“FW”.
#I changed the key “TW” by “TV”.
#I have changed the “TV” color from red to silver
```

The following will be the output of the preceding code:

```
{'Tw': {'Name': 'victor', 'Price': '70K', 'color': 'red'}, 'FW': {'Name': 'kwid', 'Price': '70L', 'co
lor': 'blue'}}
{'FW': {'Name': 'kwid', 'Price': '70L', 'color': 'blue'}, 'TV': {'Name': 'victor', 'Price': '70K', 'c
olor': 'red'}}
red
{'FW': {'Name': 'kwid', 'Price': '70L', 'color': 'blue'}, 'TV': {'Name': 'victor', 'Price': '70K', 'c
olor': 'silver'}}
```

Figure 45

193. **Write a program to create a nested list. Then, add new values and add a new list.**

Ans.

```
a=[‘a’,[‘b’,’c’],’d’]
a[0]=10
print(a)
a[1][0]=20
print(a)
a[1][1]=[‘w’]
a[1].append(‘pop’)
print(a)
a[1][0]=’q’
print(a)
```

The following will be the output of the above code:

```
[10, ['b', 'c'], 'd']
[10, [20, 'c'], 'd']
[10, [20, ['w'], 'pop'], 'd']
[10, ['q', ['w'], 'pop'], 'd']
```

Figure 46

194. **Create a set and perform a union, intersection, and difference operation.**

Ans.

```
a={‘pomogrenate’,’banana’,’guava’,’jamun’,’litchi’}
print(type(a))
for i in a:
 print(i)
```

```
a.add(‘kiwi’)
a.add(‘grapes’)
print(a)
a.remove(“banana”)
print(a)

b={‘guava’,’banana’,’apple’,’chiku’,’cherry’,’ jamun’}
print(“\n\n”)
print(“set A=”,a)
print(“set B=”,b)
c=a.union(b)
print(c)

d=a.intersection(b)
print(“\n”,d)
i=a.difference(b)
print(i)

i=b.difference(a)
print(i)
```

The following will be the output of the preceding code:

Figure 47

195. **Create a nested list and use extend and insert functions.**

Ans.

```
n=[“1”,”2”,[“3”,”4”,],”5”]
a=[“a”,”b”]
print(n)
n.extend(a)
n.insert(1,(“q”,”p”))
print(n)
```

The following will be the output of the above code:

```
['1', '2', ['3', '4'], '5']
['1', ('q', 'p'), '2', ['3', '4'], '5', 'a', 'b']
```

Figure 48

196. **Write a program to count the occurrence of "scream" in the following sentence:**

"I scream, you scream, all scream, for ice cream"?

Ans.

```
n="I scream,you scream,all scream,for ice cream"
print(n.count("scream"))
```

The following will be the output of the preceding code:

```
3
```

197. **Write a program to create a list of numbers and display the numbers divisible by 3.**

Ans.

```
n=[12,3,5,7,9,7,5,4,12]
#1st method
for i in range(0,len(n)):
 if (n[i]%3==0):
  print(n[i],end=",")

print("\n\n")
#2nd method
for i in n:
 if (i%3==0):
  print(i,end=",")

print("\b")
```

The following will be the output of the preceding code:

```
12,3,9,12,


12,3,9,12
```

Figure 49

198. **Write a program to enter your name and display every second letter.**

Ans.

```
n=input(“Enter your name”)
print(n[0:len(n):2])
```

The following will be the output of the preceding code:

```
Enter your nameswati computers jaipur
saicmuesjiu
```

Figure 50

199. **Write a program to enter your name and display it in the reverse order.**

Ans.

```
n=input(“Enter your name”)
print(n[::-1])
for i in range(len(n)-1,-1,-1):
 print(n[i],end=””)
```

The following will be the output of the preceding code:

```
Enter your nameayush
hsuya
hsuya
```

Figure 51

200. **Create a list and perform remove(), pop (), insert(), and index().**

Ans.

```
list1=[1,4,3,4,6,2,3,4,8]
print(list1.count(4))
print(len(list1))

list1.remove(6)
print(list1)
print(list1.pop(2))
print(list1)

print(“index of value 8 is : %d”%list1.index(8))

print(list1.insert(7,100))
print(list1)
```

The following will be the output of the preceding code:

```
3
9
[1, 4, 3, 4, 2, 3, 4, 8]
3
[1, 4, 4, 2, 3, 4, 8]
index of value 8 is :  6
None
[1, 4, 4, 2, 3, 4, 8, 100]
```

Figure 52

201. **Write a program to create a list and display the square of list elements.**

Ans.

```
a=[2,3,4]
b=[x*x for x in a]
print(b)
```

The following will be the output of the preceding code:

```
[4, 9, 16]
```

Figure 53

202. **Write a program to input string as “education” and display “edon.”**

Ans.

```
#education
#edon
name=”Education”
print(name[0:2]+name[-2:])
```

The following will be the output of the preceding code:

```
Edon
```

203. **Create a tuple of five elements and unpack it in five different variables.**

Ans.

```
my_Tuple = (10, 20, 30, 40,50)
a, b, c, d, e= my_Tuple
print(a)
print(b)
```

```
print(c)
print(d)
print(e)
```

The following will be the output of the preceding code:

```
10
20
30
40
50
```

204. **Write a program to create two tuples and swap them.**

Ans.

```
tuple1 = (10, 20)
tuple2 = (100, 200)
tuple1, tuple2 = tuple2, tuple1
print(tuple2)
print(tuple1)
```

The following will be the output of the preceding code:

```
(10,20)
(100,200)
```

205. **Create a function fun() such that it can accept variable length of argument and print them.**

Ans.

```
def fun(*args):
 for i in args:
  print(i)

fun(10, 20, 30,40)
fun(200,300, 100)
fun(1,2)
```

206. **How to return two numbers from a function?**

Ans.

```
def fun():
 return 10,20
a,b=fun()
print("a={},b={}".format(a,b))
```

The following will be the output of the preceding code:

```
a=10,b=20
```

207. **Create two list and iterate both lists simultaneously such that list1 should display item in the original order and list2 in the reverse order.**

Ans.

```
list1 = [10, 20, 30, 40]
list2 = [100, 200, 300, 400]

for x, y in zip(list1, list2[::-1]):
 print(x, y)
```

The following will be the output of the preceding code:

```
10 400
20 300
30 200
40 100
```

Figure 54

208. **Write a program to remove empty elements from the list.**

Ans.

```
b=[“a”,“b”,“c”,“d”]
c=list(filter(None,b))
print(c)
```

The following will be the output of the preceding code:

```
['a', 'b', 'c', 'd']
```

Figure 55

209. **Write a program to input two string and create a new string by appending s2 in the middle of s1.**

Ans.

```
s1=”HELLO”
s2=”student”
middleIndex = int(len(s1) /2)
print(“Original Strings are {} and {}”.format(s1, s2))
middleThree = s1[:middleIndex:]+ s2 +s1[middleIndex:]
print(“After appending new string in middle”, middleThree)
```

The following will be the output of the preceding code:

```
Original Strings areHELLO and student
After appending new string in middle HEstudentLLO
```

Figure 56

210. **Create a function to count all lower case, upper case, digits, and special symbols from a given string.**

Ans.

```
def count_digit_symbol_char(input_string):
charCount = 0
 digitCount = 0
 symbolCount = 0
for char in input_string:
 if char.islower() or char.isupper():
 charCount+=1
elif char.isnumeric():
 digitCount+=1
else:
 symbolCount+=1

 print("Chars = ", charCount, "Digits = ", digitCount, "Symbol
 = ", symbolCount)

input_string = "B@eP&o3Si*4ti1V#e"
print("total counts of chars, digits,and symbols \n")

count_digit_symbol_char(input_string)
```

The following will be the output of the preceding code:

```
total counts of chars, digits,and symbols

Chars =  10 Digits =  3 Symbol =  4
```

Figure 57

211. **Write a program to count the occurrence of all characters in a given string.**

Ans.

```
str1 = "banana"
countDict = dict()
for char in str1:
count = str1.count(char)
 countDict[char]=count
print(countDict)
```

The following will be the output of the preceding code:

```
{'b': 1, 'a': 3, 'n': 2}
```

Figure 58

212. **Write a program to convert a string into a date-time object.**

Ans.

```
from datetime import datetime

date_string = “Jun 1 2020 10:20AM”
datetime_object = datetime.strptime(date_string, ‘%b %d %Y
%I:%M%p’)
print(datetime_object)
```

The following will be the output of the preceding code:

```
2020-06-01 10:20:00
```

Figure 59

213. **Print the day of the week in a given date.**

Ans.

```
from datetime import datetime

given_date = datetime(2020, 6, 1)

print(given_date.strftime(‘%A’))
```

The following will be the output of the preceding code:

```
Monday
```

214. **Display today’s date in a different format.**

Ans.

```
from datetime import date

today = date.today()
print(“Today’s date:”, today)
# dd/mm/YY
d1 = today.strftime(“%d/%m/%Y”)
print(“d1 =”, d1)

# Textual month, day and year
d2 = today.strftime(“%B %d, %Y”)
```

```
print(“d2 =”, d2)

# mm/dd/y
d3 = today.strftime(“%m/%d/%y”)
print(“d3 =”, d3)

# Month abbreviation, day and year
d4 = today.strftime(“%b-%d-%Y”)
print(“d4 =”, d4)

print(“Today’s date and time”,datetime.now())
```

The following will be the output of the preceding code:

```
Today's date: 2020-09-25
d1 = 25/09/2020
d2 = September 25, 2020
d3 = 09/25/20
d4 = Sep-25-2020
Today's date and time 2020-09-25 16:39:25.482464
```

Figure 60

215. **Print the current time in Python.**

Ans.

```
from datetime import datetime
now = datetime.now().time() # time object
print(“now =”, now)
```

216. **Calculate the difference in years between two dates.**

Ans.

The first method

```
from datetime import datetime
start_date = datetime(2010,1,1)
end_date = datetime(2012,1,1)
difference = end_date - start_date
difference_in_years = (difference.days +
difference.seconds/86400)/365.2425
print(round(difference_in_years))
```

The second method

```
from datetime import datetime
from dateutil.relativedelta import relativedelta
```

```
datetime1 = datetime(2000, 1, 1)
datetime2 = datetime(2010, 1, 1)
time_difference = relativedelta(datetime2, datetime1)
difference_in_years = time_difference. years
print(difference_in_years, “years”)
```

217. **Generate captcha. On every refresh, it should change. It must satisfy the following condition:**

    It must be eight characters long and contain two uppercase letters, one symbol, and a digit.

    Ans.

```
import random
import string

def captcha():
 randomSource = string.ascii_letters + string.digits +
 string.punctuation
 capt = random.sample(randomSource, 4)
 capt += random.sample(string.ascii_uppercase, 2)
 capt += random.choice(string.digits)
 capt+= random.choice(string.punctuation)

 captcha_list = list(capt)
 random.SystemRandom().shuffle(captcha_list)
 capt = ‘’.join(captcha_list)
 return capt

print (“Random captcha “, captcha())
```

218. **How to create a class without any variable and method and a method without the body?**

    Ans.

```
class student:
 pass

def fun():
 pass
```

219. **How to check which class an object belongs to and check its base class?**

    Ans.

```
class Vehicle:
 def __init__(self, name):
  self.name = name
```

```
class Bus(Vehicle):
 pass

class Car(Vehicle):
 pass

School_bus = Bus(“Volvo”)
cab=Car(“Bugatti”)

# use Python’s built-in type() function
print(type(School_bus))
print(isinstance(School_bus, Vehicle))
print(type(cab))
print(isinstance(cab, Vehicle))
```

The following will be the output of the preceding code:

```
<class '__main__.Bus'>
True
<class '__main__.Car'>
True
```

Figure 61

220. **Remove items from set1 that are not common to both set1 and set2.**

Ans.

```
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}
set1.intersection_update(set2)
print(set1)
```

221. **Get the key corresponding to the minimum value from the following dictionary:**

```
fruits={‘apple’:100,’kiwi’:200,’banana’: 80,’pineapple’:150}
```

Ans.

```
print(min(fruits,key=fruits.get))
```

222. **Create a 4X2 integer array and print its attribute.**

Ans.

```
import numpy

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

```
print(“Printing numpy array Attributes”)
print(“1-- Array Shape is: “, firstArray.shape)
print(“2-- Array dimensions are “, firstArray.ndim)
print(“3-- Length of each element of array in bytes is “,
firstArray.itemsize)
```

223. **Create a numpy array and print the value of the third column of all rows.**

Ans.

```
import numpy

sampleArray = numpy.array([[1,2,3], [11,22,33], [111,222,333],
[1111,2222,3333]])
print(“Printing Input Array”)
print(sampleArray)

print(“\n Printing array of items in the third column from all
rows”)
newArray = sampleArray[…,2]
print(newArray)
```

224. **Create an 8X3 integer array from a range between 1 and 24 such that the difference between each element is 1, and then split the array into four equal-sized sub-arrays.**

Ans.

```
import numpy

print(“Creating 8X3 array using numpy.arange”)
sampleArray = numpy.arange(1, 25, 1)
sampleArray = sampleArray.reshape(8,3)
print (sampleArray)

print(“\nDividing 8X3 array into 4 sub array\n”)
subArrays = numpy.split(sampleArray, 4)
print(subArrays)
```

The following will be the output of the preceding code:

```
Creating 8X3 array using numpy.arange
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]
 [13 14 15]
 [16 17 18]
 [19 20 21]
 [22 23 24]]

Dividing 8X3 array into 4 sub array

[array([[1, 2, 3],
       [4, 5, 6]]), array([[ 7,  8,  9],
       [10, 11, 12]]), array([[13, 14, 15],
       [16, 17, 18]]), array([[19, 20, 21],
       [22, 23, 24]])]
```

Figure 62

225. **Find each company's highest price car.**

company	company	price
alfa-romero	alfa-romero	16500.0
audi	audi	18920.0
bmw	bmw	41315.0
chevrolet	chevrolet	6575.0
dodge	dodge	6377.0
honda	honda	12945.0
isuzu	isuzu	6785.0
jaguar	jaguar	36000.0
mazda	mazda	18344.0
mercedes-benz	mercedes-benz	45400.0
mitsubishi	mitsubishi	8189.0
nissan	nissan	13499.0
porsche	porsche	37028.0
toyota	toyota	15750.0
volkswagen	volkswagen	9995.0
volvo	volvo	13415.0

Figure 63

Ans.

```
import pandas as pd
df = pd.read_csv("Desktop\\Automobile_data.csv")
car_Manufacturers = df.groupby('company')
mileageDf = car_Manufacturers['company','average-
mileage'].mean()
print(mileageDf)
```

226. **Write a program to find the sum of series 5 +55+555+5555 ….N.**

Ans.

```
number_of_terms = 5
start = 5
sum = 0
for i in range(0, number_of_terms):
 print(start, end=" ")
 sum += start
 start = (start * 10) + 5
print("\nSum of above series is:", sum)
```

The following will be the output of the preceding code:

```
5 55 555 5555 55555
Sum of above series is: 61725
```

Figure 64

227. **Write a program to return the sum and average of the digits that appear in the string, ignoring all other characters.**

Ans.

```
import re

inputStr = "English = 98 Science = 78 Math = 88 History = 75"
markList = [int(num) for num in re.findall(r'\b\d+\b',
inputStr)]
totalMarks = 0
for mark in markList:
 totalMarks+=mark

percentage = totalMarks/len(markList)
print("Total Marks is:", totalMarks, "Percentage is ",
percentage)
```

The following will be the output of the preceding code:

Figure 65