UNIT-1

MCQ

1. Which of these in not a core data type?

a) Lists

b) Dictionary

c) Tuples

d) Class

Answer: d

Explanation: Class is a user defined data type.

2. What will be the output of the following Python code? 1. >>>str="hello" 2. >>>str[:2] 3. >>>

a) he

b) lo

c) olleh

d) hello

Answer: a Explanation: We are printing only the 1st two bytes of string and hence the answer is “he”

3.What data type is the object below? L = [1, 23, 'hello', 1]

a) list

b) dictionary

c) array

d) tuple

Answer: a Explanation: List data type can store any values within it

4. Which of the following results in a SyntaxError?

a) ‘”Once upon a time…”, she said.’

b) “He said, ‘Yes!'”

c) ‘3\’

d) ”’That’s okay”’

Answer: c Explanation: Carefully look at the colons.

5. What is the average value of the following Python code snippet?

1. >>>grade1 = 80 2. >>>grade2 = 90 3. >>>average = (grade1 + grade2) / 2

a) 85.0

b) 85.1

c) 95.0

d) 95.1

Answer: a Explanation: Cause a decimal value of 0 to appear as output.

6)What is the type of inf?

a)Boolean

b) Integer

c) Float

d) Complex

Answer: c

Explanation: Infinity is a special case of floating point numbers. It can be obtained by float('inf')

7. Which of the following is incorrect?

a) x = 0b101

b) x = 0x4f5

c) x = 19023

d) x = 03964

Answer: d

8. What does 3 ^ 4 evaluate to?

a) 81

b) 12

c) 0.75

d) 7

Answer: d

Explanation: ^ is the Binary XOR operator

9. What will be the value of the following Python expression? 4 + 3 % 5

a) 4

b) 7

c) 2

d) 0

Answer: b

Explanation: The order of precedence is: %, +. Hence the expression above, on simplification results in 4 + 3 = 7. Hence the result is 7.

10.W is the value of the following expression? 8/4/2, 8/(4/2)

a) (1.0, 4.0)
b) (1.0, 1.0)
c) (4.0. 1.0)
d) (4.0, 4.0)
Answer: a
Explanation: The above expressions are evaluated as: 2/2, 8/2, which is equal to (1.0, 4.0).

11. 11. What is the value of the following expression? float(22//3+3/3)

a) 8

b) 8.0

c) 8.3

d) 8.33

Answer: b

Explanation: The expression shown above is evaluated as: float(7+1) = float(8) = 8.0. Hence the result of this expression is 8.0.

12. Is Python case sensitive when dealing with identifiers?

a) yes

b) no

c) machine dependent

d) none of the mentioned

Answer: a yes

13. What is the maximum possible length of an identifier?

a) 31 characters

b) 63 characters

c) 79 characters

d) none of the mentioned

Answer: a 31

14. Which of the following is an invalid variable?

a) my_string_1

b) 1st_string

c) foo

d) _

Answer: b

15. Which of the following is not a keyword?

a) eval

b) assert

c) nonlocal

d) pass

Answer: a
Explanation: eval can be used as a variable.

16. Which of the following is an invalid statement?

a) abc = 1,000,000

b) a b c = 1000 2000 3000

c) a,b,c = 1000, 2000, 3000

d) a_b_c = 1,000,000

Answer: b
Explanation: Spaces are not allowed in variable names.

17. What will be the output of the following Python expression if x=15 and y=12? x & y

a) b1101

b) 0b1101

c) 12

d) 1101

Answer: d

Explanation: The symbol '&' represents bitwise AND. This gives 1 if both the bits are equal to 1, else it gives 0. The binary form of 15 is 1111 and that of 12 is 1100. Hence on performing the bitwise AND operation, we get 1100, which is equal to 12

18. What will be the output of the following Python expression? 0x35 | 0x75

a) 115

b) 116

c) 117

d) 118

Answer: c

Explanation: The binary value of 0x35 is 110101 and that of 0x75 is 1110101. On OR-ing these two values we get the output as: 1110101, which is equal to 117. Hence the result of the above expression is 117.

19. What will be the value of the following Python expression? bin(10-2)+bin(12^4)

a) 0b10000

b) 0b10001000

c) 0b1000b1000

d) 0b10000b1000

Answer: d

Explanation: The output of bin(10-2) = 0b1000 and that of bin(12^4) is ob1000. Hence the output of the above expression is: 0b10000b1000.

20. What will be the output of the ~100 Python expression?

a)101
b)-101
c)100
d)-100
View Answer
Answer: b
Explanation: Suppose we have an expression ~A. This is evaluated as: -A – 1. Therefore, the expression ~100 is evaluated as -100 – 1, which is equal to -101.

Chapter 1 short

1. What is the difference between list and tuples in Python?

Sr. No.	Key	List	Tuple
1	Type	List is mutable.	Tuple is immutable.
2	Iteration	List iteration is slower and is time consuming.	Tuple iteration is faster.
3	Appropriate for	List is useful for insertion and deletion operations.	Tuple is useful for readonly operations like accessing elements.
4	Memory Consumption	List consumes more memory.	Tuples consumes less memory.
5	Methods	List provides many in-built methods.	Tuples have less in-built methods.
6	Error prone	List operations are more error prone.	Tuples operations are safe.

2. What are the key features of Python?

Ans Easy to code: **Python** is a high-level programming language. ...

- Free and Open Source: ...
- Object-Oriented Language: ...
- GUI Programming Support: ...
- High-Level Language: ...
- Extensible **feature**: ...
- **Python** is Portable language: ...
- **Python** is Integrated language

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

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

4. How is Python an interpreted language?

Python is an "**interpreted**" **language**. This means it uses an **interpreter**. An **interpreter** is very different from the compiler. An **interpreter** executes the statements of code "one-by-one" whereas the compiler executes the code entirely and lists all possible errors at a time.

5.What is pep 8?

PEP 8, sometimes spelled **PEP8** or **PEP-8**, is a document that provides guidelines and best practices on how to write Python code. It was written in 2001 by Guido van Rossum, Barry Warsaw, and Nick Coghlan. The primary focus of **PEP 8** is to improve the readability and consistency of Python code

6. How is memory managed in Python?

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

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

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference. Complex() math()

8. What are local variables and global variables in Python?

A local variable is a type of variable declared within programming block or subroutines. It can only be used only inside that subroutine or code block in which they were declared.

Global variables are defined outside of a subroutine or function. The global variable will hold its value throughout the lifetime of a program. They can be accessed within any function defined for the program.

9. 9. What is type conversion in Python?

Type Conversion is the **conversion** of object from one data **type** to another data **type**. Implicit **Type Conversion** is automatically performed by the **Python** interpreter. **Python** avoids the loss of data in Implicit **Type Conversion**.

10. Is indentation required in python?

One of the most distinctive features of **Python** is its use of **indentation** to mark blocks of code. ... In most other programming languages, **indentation** is used only to help make the code look pretty. But in **Python**, it is **required** for indicating what block of code a statement belongs to

11. What is the difference between Python Arrays and lists?

Lists

- Python lists are very flexible and can hold arbitrary data.
- Lists are a part of Python's syntax, so they do not need to be declared first.

Arrays

- Python arrays are just a thin wrapper on C arrays.
- Arrays need to first be imported, or declared, from other libraries (i.e. `numpy`).

12. What are the built-in types of python?

The principal built-in types are numerics, sequences, mappings, classes, instances and exceptions.

Some collection classes are mutable. The methods that add, subtract, or rearrange their members in place, and don't return a specific item, never return the collection instance itself but `None`

13. Explain the need for Unicodes.

LONG ANSWER

1.Explain the different string formats available in Python with examples

2. Discuss the int(), float(), str(), chr() and complex() type conversion functions with examples

1. int(a,base) : This function converts **any data type to integer**. 'Base' specifies the **base in which string is** if data type is string.
2. float() : This function is used to convert **any data type to a floating point number**
3.str() : Used to **convert integer into a string.**
4.complex(real,imag) : : This function **converts real numbers to complex(real,imag) number.**
5.chr(number) : : This function **converts number to its corresponding ASCII character.**

```
# Convert ASCII value to characters
a = chr(76)
b = chr(77)

print(a)
print(b)


Another example

y = 2.8 # float
z = 1j # complex

#convert from int to float:
a = float(x)

#convert from float to int:
b = int(y)

#convert from int to complex:
c = complex(x)

print(a)
print(b)
print(c)

print(type(a))
print(type(b))
print(type(c))
```

3.Discuss the ord(), hex(), oct(), complex() and float() type conversion functions with examples.

. ord() : This function is used to convert a **character to integer.**

hex() : This function is to convert **integer to hexadecimal string**.

oct() : This function is to convert **integer to octal string**.
complex(real,imag) : : This function **converts real numbers to complex(real,imag) number.**
float() : This function is used to convert **any data type to a floating point number**

4. Describe the is and is not operators and type() function. Also, discuss why Python is called as dynamic and strongly typed language.

Ans-**'is' operator –** Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.

```
if (type(x) is int):
    print("true")
else:
    print("false")
```

'is not' operator – Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.

```
if (type(x) is not int):
    print("true")
else:
    print("false")
```

Python is strongly typed as the interpreter keeps track of all variables types. It's also very dynamic as it rarely uses what it knows to limit variable usage. In Python, it's the program's responsibility to use built-in functions like isinstance() and issubclass() to test variable types and correct usage. Python tries to stay out of your way while giving you all you need to implement strong type checking.

People often use the term strongly-typed language to refer to a language that is both statically typed (types are associated with a variable declaration -- or, more generally, the compiler can tell which type a variable refers to, for example through type inference, without executing the program) and strongly-typed (restrictive about how types can be intermingled).

5. What is Python? Describe its features and applications?

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming.
In Python, we don't need to declare the type of variable because it is a dynamically typed language.

1. Easy to code:

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.

2. Free and Open Source:

Python language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword.

Download Python

Since it is open-source, this means that source code is also available to the public. So you can download it as, use it as well as share it.

3. Object-Oriented Language:

One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

4. GUI Programming Support:

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python.

PyQt5 is the most popular option for creating graphical apps with Python.

5. High-Level Language:

Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

6. Extensible feature:

Python is a **Extensible** language. We can write us some Python code into C or C++ language and also we can compile that code in C/C++ language.

7. Python is Portable language:

Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

8. Python is Integrated language:

Python is also an Integrated language because we can easily integrated python with other languages like c, c++, etc.

5. Differentiate between lists and tuples in Python?

List	**Tuple**		
1	Type	List is mutable.	Tuple is immutable.
2	Iteration	List iteration is slower and is time consuming.	Tuple iteration is faster.
3	Appropriate for	List is useful for insertion and deletion operations.	Tuple is useful for readonly operations like accessing elements.
4	Memory Consumption	List consumes more memory.	Tuples consumes less memory.

List	Tuple		
5	Methods	List provides many in-built methods.	Tuples have less in-built methods.
6	Error prone	List operations are more error prone.	Tuples operations are safe.

7.Explain in detail about Python type conversion and type casting?

Python defines type conversion functions to directly convert one data type to another which is useful in day to day and competitive programming. This article is aimed at providing the information about certain conversion functions.

1. int(a,base) : This function converts **any data type to integer**. ‘Base’ specifies the **base in which string is** if data type is string.
2. float() : This function is used to convert **any data type to a floating point number**
3. ord() : This function is used to convert a **character to integer.**

4. hex() : This function is to convert **integer to hexadecimal string**.
5. oct() : This function is to convert **integer to octal string**.
6. tuple() : This function is used to **convert to a tuple**.
7. set() : This function returns the **type after converting to set**.
8. list() : This function is used to convert **any data type to a list type**.

Casting is when you convert a variable value from one type to another. That is, in Python, done with functions like int() or float() or str(). A pattern is that you convert a number as a string into a number.

Python includes two types of type conversion.

- Explicit Conversion
- Implicit Conversion

In Explicit Type Conversion, users convert the data type of the item to needed data type. We use the predefined roles such as int(), float(), str(), etc to execute explicit type conversion.

This procedure does not require any user participation. Let us see an instance where Python boosts the conversion of a lower data type (integer) to the greater data type (floats) to prevent data loss.

8. What are operators in Python? Describe specifically about identity membership operator?

Python Operators: Arithmetic, Logical, Comparison, Assignment, Bitwise & Precedence. **Operators** are used to perform **operations** on values and variables. **Operators** can manipulate individual items and returns a result. The data items are referred as operands or arguments.

Membership operators are operators used to validate the membership of a value. It test for membership in a sequence, such as strings, lists, or tuples.

1. **in operator :** The 'in' operator is used to check if a value exists in a sequence or not. Evaluates to true if it finds a variable in the specified sequence and false otherwise

2.'not in' operator- Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.

In Python are used to determine whether a value is of a certain class or type. They are usually used to determine the type of data a certain variable contains.
There are different identity operators such as

1. **'is' operator –** Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.

'is not' operator – Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.

CHAPTER – 2

Mcq

2. What is the output of print(k) in the following Python code snippet?

```
k = [print(i) for i in my_string if i not in "aeiou"]
print(k)
```

a) all characters of my_string that aren’t vowels
b) a list of Nones
c) list of Trues
d) list of Falses
View Answer
Answer:b
Explanation: print() returns None.

2) What will be the output of the following Python code snippet? x = [i**+1 for i in range(3)]; print(x);
a) [0, 1, 2]

b) [1, 2, 5]

c) error, **+ is not a valid operator

d) error, ‘;’ is not allowed

Answer: a

3. What will be the output of the following Python code snippet? print([[i+j for i in "abc"] for j in "def"])
a) [‘da’, ‘ea’, ‘fa’, ‘db’, ‘eb’, ‘fb’, ‘dc’, ‘ec’, ‘fc’]

b) [[‘ad’, ‘bd’, ‘cd’], [‘ae’, ‘be’, ‘ce’], [‘af’, ‘bf’, ‘cf’]]

c) [[‘da’, ‘db’, ‘dc’], [‘ea’, ‘eb’, ‘ec’], [‘fa’, ‘fb’, ‘fc’]]

d) [‘ad’, ‘ae’, ‘af’, ‘bd’, ‘be’, ‘bf’, ‘cd’, ‘ce’, ‘cf’]

Answer: b

Explanation: The inner list is generated once for each value of j

4) 2. What will be the output of the following Python code? x = ['ab', 'cd'] for i in x: x.append(i.upper()) print(x)

a) [‘AB’, ‘CD’]

b) [‘ab’, ‘cd’, ‘AB’, ‘CD’]

c) [‘ab’, ‘cd’]

d) none of the mentioned

Answer: d

Explanation: The loop does not terminate as new elements are being added to the list in each iteration.

5. What will be the output of the following Python code? i = 1 while True: if i%2 == 0: break print(i) i += 2

a) 1

b) 1 2

c) 1 2 3 4 5 6 …

d) 1 3 5 7 9 11 …

Answer: d

Explanation: The loop does not terminate since i is never an even number.

6. What will be the output of the following Python code? True = False while True: print(True) break
a) True

b) False

c) None

d) none of the mentioned View Answer

Answer: d

Explanation: SyntaxError, True is a keyword and it’s value cannot be changed.

7) 4. What will be the output of the following Python code? x = "abcdef" i = "i" while i in x: print(i, end=" ")

a) no output

b) i i i i i i …

c) a b c d e f

d) abcdef

Answer: a Explanation: “i” is not in “abcdef”

8) . What will be the output of the following Python code? x = "abcdef" i = "i" while i in x: print(i, end=" ")

a) no output

b) i i i i i i …

c) a b c d e f

d) abcdef

Answer: a Explanation: “i” is not in “abcdef”.

9)What will be the output of the following Python code? for i in range(int(2.0)): print(i)

a) 0.0 1.0

b) 0 1

c) error

d) none of the mentioned

Answer: b Explanation: range(int(2.0)) is the same as range(2).

10)What will be the output of the following Python code snippet? for i in [1, 2, 3, 4][::-1]: print (i)

a) 1 2 3 4

b) 4 3 2 1

c) error

d) none of the mentioned

Answer: b Explanation: [::-1] reverses the list

11. What will be the output of the following Python code snippet? for i in ''.join(reversed(list('abcd'))): print (i)

a) a b c d

b) d c b a

c) error

d) none of the mentioned

Answer: b Explanation: ‘ ‘.join(reversed(list(‘abcd’))) reverses a string

12.What will be the output of the following Python code? for i in range(5): if i == 5: break else: print(i) else: print("Here")

a) 0 1 2 3 4 Here

b) 0 1 2 3 4 5 Here

c) 0 1 2 3 4

d) 1 2 3 4 5

Answer: a Explanation: The else part is executed if control doesn’t break out of the loop.

13. What will be the output of the following Python code? x = (i for i in range(3)) for i in x: print(i) for i in x: print(i)

a) 0 1 2

b) error

c) 0 1 2 0 1 2

d) none of the mentioned

Answer: a

Explanation: We can loop over a generator object only once

14. What will be the output of the following Python code snippet? a = [0, 1, 2, 3] for a[-1] in a: print(a[-1])

a) 0 1 2 3

b) 0 1 2 2

c) 3 3 3 3

d) error V

Answer: b

Explanation: The value of a[-1] changes in each iteration

15. What will be the output of the following Python statement? 1. >>>"abcd"[2:]

a) a

b) ab

c) cd

d) dc

Answer: c

Explanation: Slice operation is performed on string.

16.print(0xA + 0xB + 0xC):

a) 0xA0xB0xC

b) Error

c) 0x22

d) 33

Answer: d

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

>>>max("what are you")

a) error

b) u

c) t

d) y

Answer:

18. Suppose x is 345.3546, what is format(x, “10.3f”) (_ indicates space).

a) __345.355

b) ___345.355

c) ____345.355

d) _____345.354

Answer: b

Explanation: Execute in the shell to verify

19.What will be the output of the following Python code? print('*', "abcdef".center(7), '*')

a) * abcdef *

b) * abcdef *

c) *abcdef *

d) * abcdef*

Answer: b

20. What will be the output of the following Python code? print("abcdef".center(7, '1'))

a) 1abcdef

b) abcdef1

c) abcdef

d) error

Answer: a

21) What will be the output of the following Python code? advertisement
print("xyyzxyzxzxyy".count('yy', 2))

a) 2

b) 0

c) 1

d) none of the mentioned

Answer: c

22) . What will be the output of the following Python code? elements = [0, 1, 2] def incr(x): return x+1 print(list(map(incr, elements)))

a) [1, 2, 3]

b) [0, 1, 2]

c) error

d) none of the mentioned

Answer: a

23.What will be the output of the following Python code? x = ['ab', 'cd'] print(list(map(list, x)))

a) [‘ab’, ‘cd’]

b) [2, 2]

c) [‘2’, ‘2’]

d) none of the mentioned

Answer:

24) In ________________ copy, the base address of the objects are copied. In ________________ copy, the base address of the objects are not copied.

a) deep. Shallow

b) memberwise, shallow

c) shallow, deep

d) deep, memberwise

Answer:C

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

[1, 2, 3, 4, 5] lambda x: (b (x[1:]) + x[:1] if x else []) print(b (a))

a) 1 2 3 4 5

b) [5,4,3,2,1]

c) [] d) Error, lambda functions can’t be called recursively

Answer: c

Short answer

1. What are functions in Python?

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

2. What is __init__?

"__init__" is a reseved method in python classes. It is called as a constructor in object oriented terminology. This method is called when an object is created from a class and it allows the class to initialize the attributes of the class.

3. What is a lambda function?

A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

4. What is self in Python?

self represents the instance of the class. By using the “self” keyword we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.

5. How does break, continue and pass work?

- **Pass**. Let's start with something relatively easy (but not straightforward). Nothing happens when **pass is** executed. ...
- **Break**. The **break** statement allows you to leave a for or while loop prematurely. ...
- **Continue**. The **continue** statement ignores the rest of the statements inside a loop, and continues with the next iteration.

6. What does [::-1} do?

7.How can you randomize the items of a list in place in Python?

The **shuffle()** method randomizes the items of a list in place.

Following is the syntax for **shuffle()** method −

```
shuffle (lst,[random])
```

Parameters

- **lst −** This could be a list or tuple.
- **random −** This is an optional 0 argument function returning float between 0.0 - 1.0. Default is None

8. What are python iterators?

An iterator is an object that contains a countable number of values.

An iterator is an object that can be iterated upon, meaning that you can traverse through all the values.

Technically, in Python, an iterator is an object which implements the iterator protocol, which consist of the methods `__iter__()` and `__next__()`.

9. How do you write comments in python?

To write a comment in Python, simply put the hash mark # before your desired comment:

```
# This is a comment
```

10. What is pickling and unpickling?

"**Pickling**" is the process whereby a Python object hierarchy is converted into a byte stream, and "**unpickling**" is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

11. What are docstrings in Python?

Python documentation strings (or **docstrings**) provide a convenient way of associating documentation with **Python** modules, functions, classes, and methods. It's specified in source code that is used, like a comment, to document a specific segment of code.

12. What is the purpose of is, not and in operators?

13 . What is the usage of help() and dir() function in Python?

The **dir function** returns a list of all the attributes within in an object - these could be either data values or **methods**, and the list includes private and magic **methods**. The **help function** works by formatting all of the docstrings from the object with other data into a hopefully useful **help** page.

14. . What is a dictionary in Python?

A **dictionary** is a collection which is unordered, changeable and indexed.
In **Python** dictionaries are written with curly brackets, and they have keys and values.

15. How can the ternary operators be used in python?

Many programming languages support **ternary operator**, which basically define a conditional expression. Similarly the **ternary operator** in **python** is **used to** return a value based on the result of a binary condition. It takes binary value(condition) as an input, so it looks similar **to** an “if-else” condition block

16. What does this mean: *args, **kwargs? And why would we use it?

Kwargs allow **you** to pass keyword **arguments** to **a** function. They **are used** when **you are** not sure of the number of keyword **arguments** that **will** be passed in the function. **Kwargs can** be **used** for unpacking dictionary key, value pairs.
This **is** done **using** the double asterisk notation (**).

17. What does len() do?

The **len()** function returns the number of items in an object. When the object **is** a string, the **len()** function returns the number of characters in the string.

18. What are negative indexes and why are they used?

Negative indexes are a way to allow you to **index** into a list, tuple or other indexable container relative to the end of the container, rather than the start. **They** are **use** d because **they** are more efficient and are considered by much more readable.

19. How to import modules in python?

import module_name
When import is used, it searches for the module initially in the local scope by calling __import__() function. The value returned by the function are then reflected in the output of the initial code.

```
import math
print(math.pi)
```

Output:

```
3.141592653589793
```

20. What are Python libraries? Name a few of them

A **Python library** is a reusable chunk of code that you may want to include in your programs/ projects. Compared to languages like C++ or C, a **Python libraries** do not pertain to any specific context in **Python**. Here, a '**library**' loosely describes a collection of core modules

- **Pandas**. ...
- **Keras**. ...
- **SciKit-Learn**. ...
- PyTorch. ...
- **TensorFlow**. ..

21. What is module and package in Python?

A **package** is a collection of **Python modules**: while a **module** is a single **Python** file, a **package** is a directory of **Python modules** containing an additional __init__**.py** file, to distinguish a **package** from a directory that just happens to contain a bunch of **Python** scripts.

22. How can you share global variables across modules?

The best way **to share global variables across modules across** a single program is **to** create a config **module**. Just import the config **module in all modules** of your application; the **module** then becomes available as a **global** name

CHAPTER-2 LONG ANSWER

1. **Illustrate the different types of control flow statements available in Python with flowcharts.**

There are 6 different types of flow control statements available in Python:

1. *if-else*
2. *Nested if-else*
3. *for*
4. *while*
5. *break*
6. *Continue*

if-else

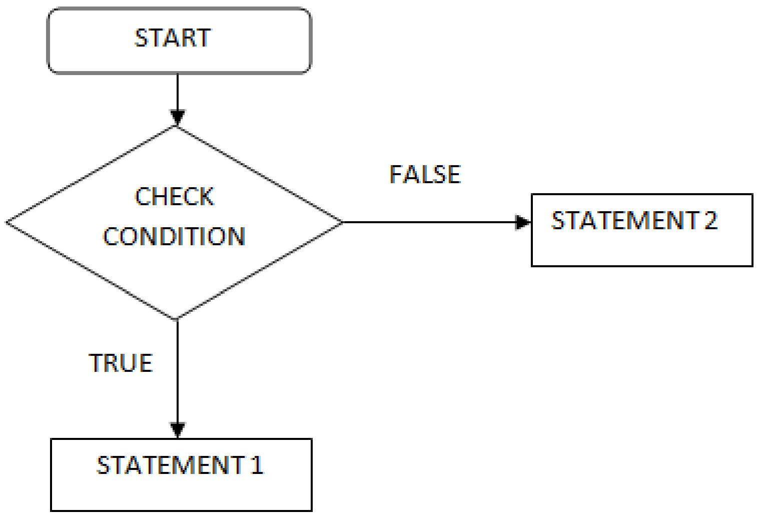
If-else is used for decision making; when code statements satisfy the condition, then it will return non-zero values as true, or else zero or NONE value as false, by the Python interpreter.

Syntax

```
if(<condition>):
    Statement 1
    ...
```

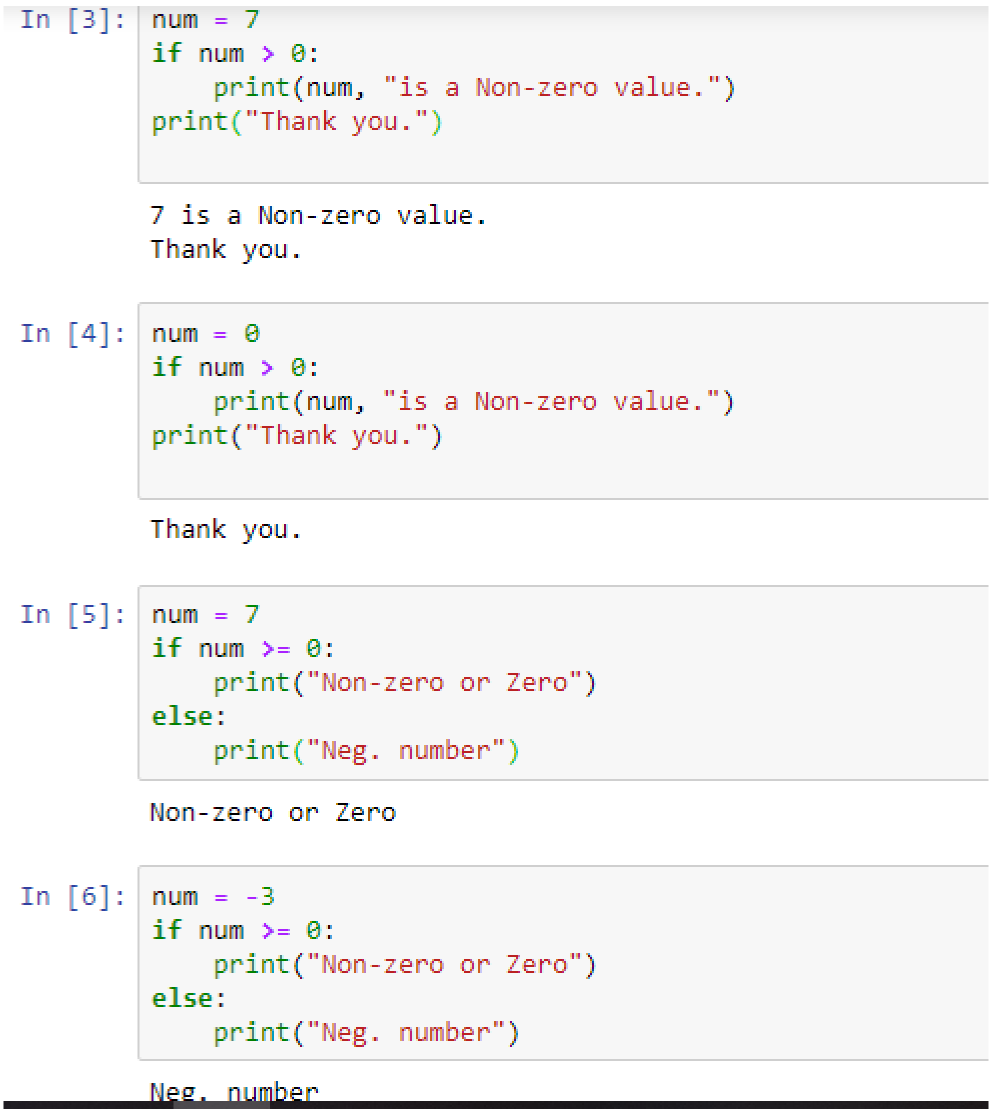
```
else:
    Statement 2
    ...
```

Understand this statement with a flow chart.



Example

Check In[3] and In[4] and also In[5] and In[6].



```
In [3]: num = 7
        if num > 0:
            print(num, "is a Non-zero value.")
        print("Thank you.")

        7 is a Non-zero value.
        Thank you.

In [4]: num = 0
        if num > 0:
            print(num, "is a Non-zero value.")
        print("Thank you.")

        Thank you.

In [5]: num = 7
        if num >= 0:
            print("Non-zero or Zero")
        else:
            print("Neg. number")

        Non-zero or Zero

In [6]: num = -3
        if num >= 0:
            print("Non-zero or Zero")
        else:
            print("Neg. number")

        Neg. number
```

Nested if-else

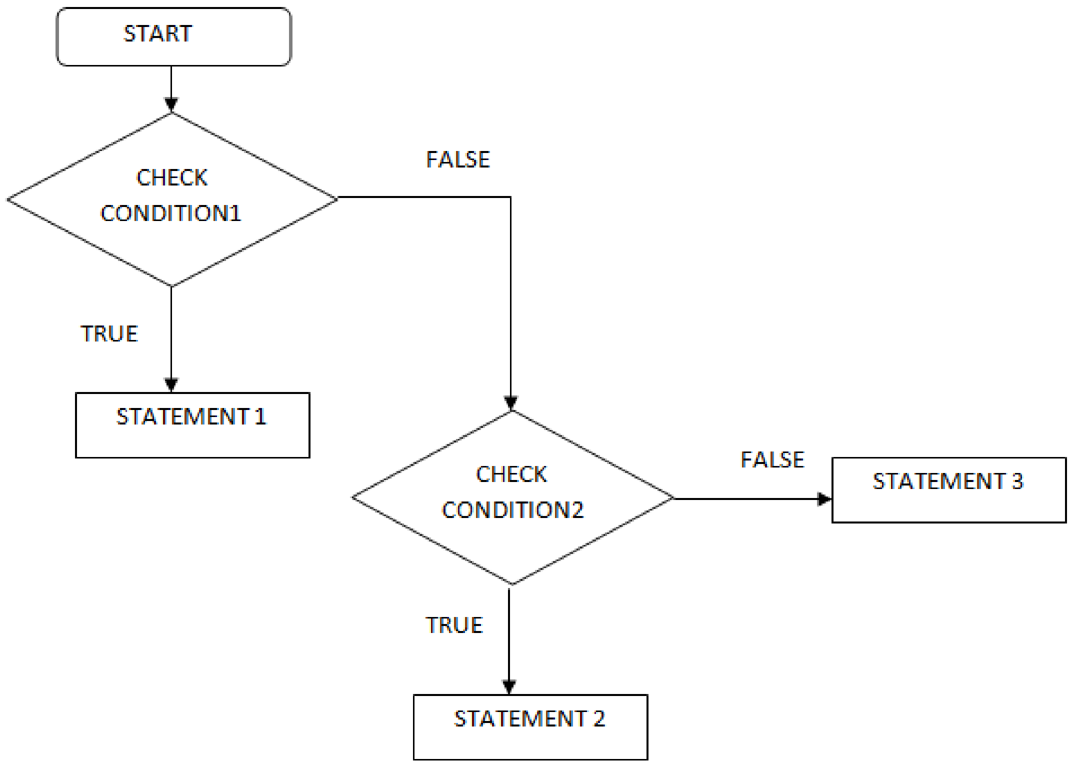
With if...elif...else, elif is a shortened form of else if. It works the same as 'if' statements, where the if block condition is false then it checks to elif blocks. If all blocks are false, then it executes an else statement. There are multiple elif blocks possible for a Nested if...else.

Syntax

```
if  (<condition 1>):
    Statement 1
    ...
elif (<condition 2>):
    Statement 2
    ...
else
    Statement 3
```

```
9.        ...
```

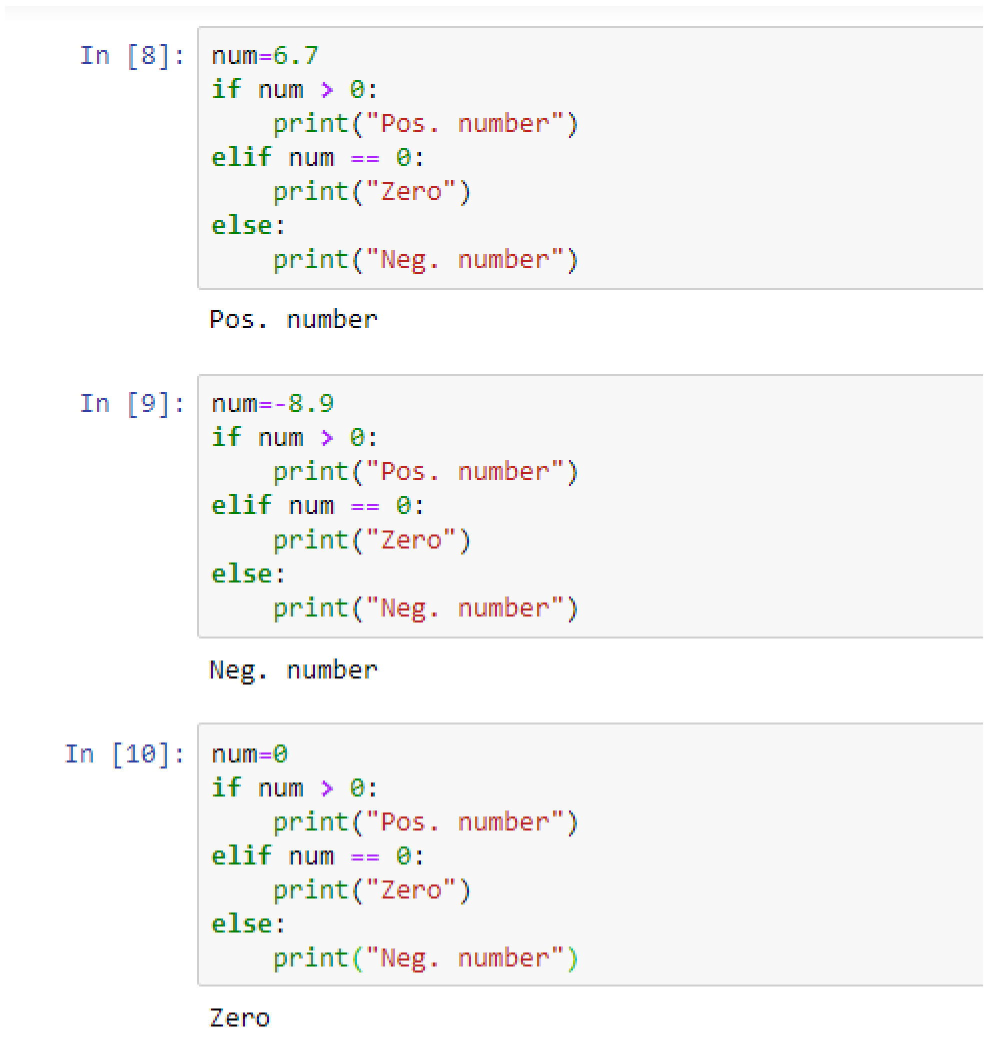
Flow chart of Nested-if else



Remember there is no condition statement associated with else part of these flow control statements. It will execute ig statements only in the case that of all conditions are false.

Example

Check In[3] and In[8] and also In[9] and In[10].



```
In [8]:  num=6.7
         if num > 0:
             print("Pos. number")
         elif num == 0:
             print("Zero")
         else:
             print("Neg. number")

         Pos. number

In [9]:  num=-8.9
         if num > 0:
             print("Pos. number")
         elif num == 0:
             print("Zero")
         else:
             print("Neg. number")

         Neg. number

In [10]: num=0
         if num > 0:
             print("Pos. number")
         elif num == 0:
             print("Zero")
         else:
             print("Neg. number")

         Zero
```

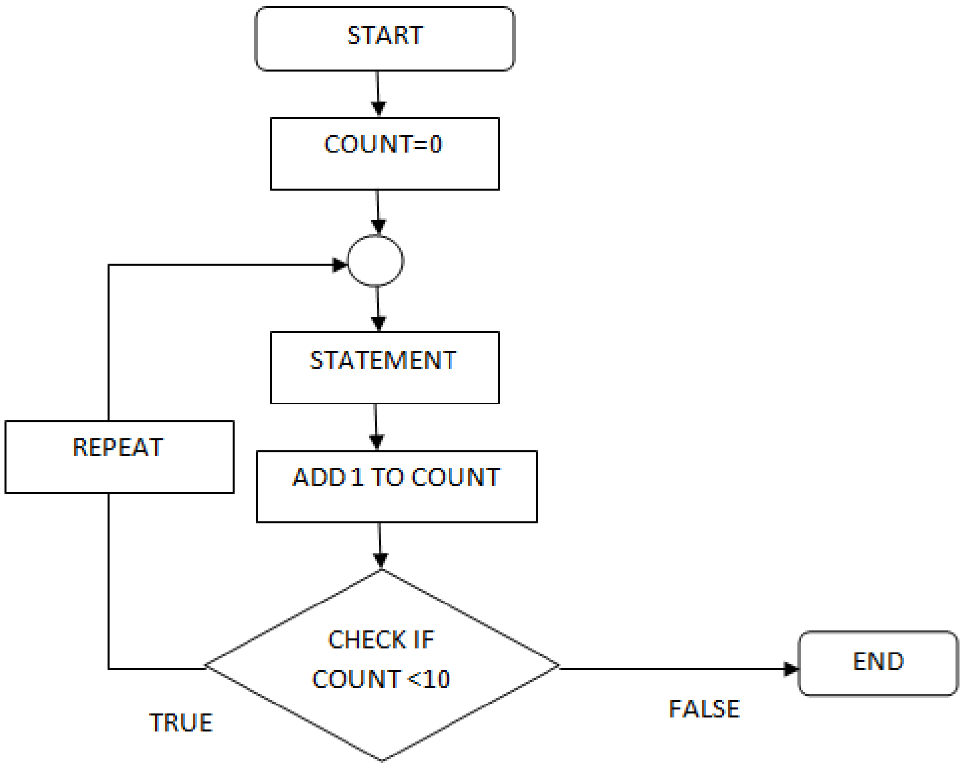
for Statement

The for loop statement has variable iteration in a sequence(list, tuple or string) and executes statements until the loop does not reach the false condition.

Syntax

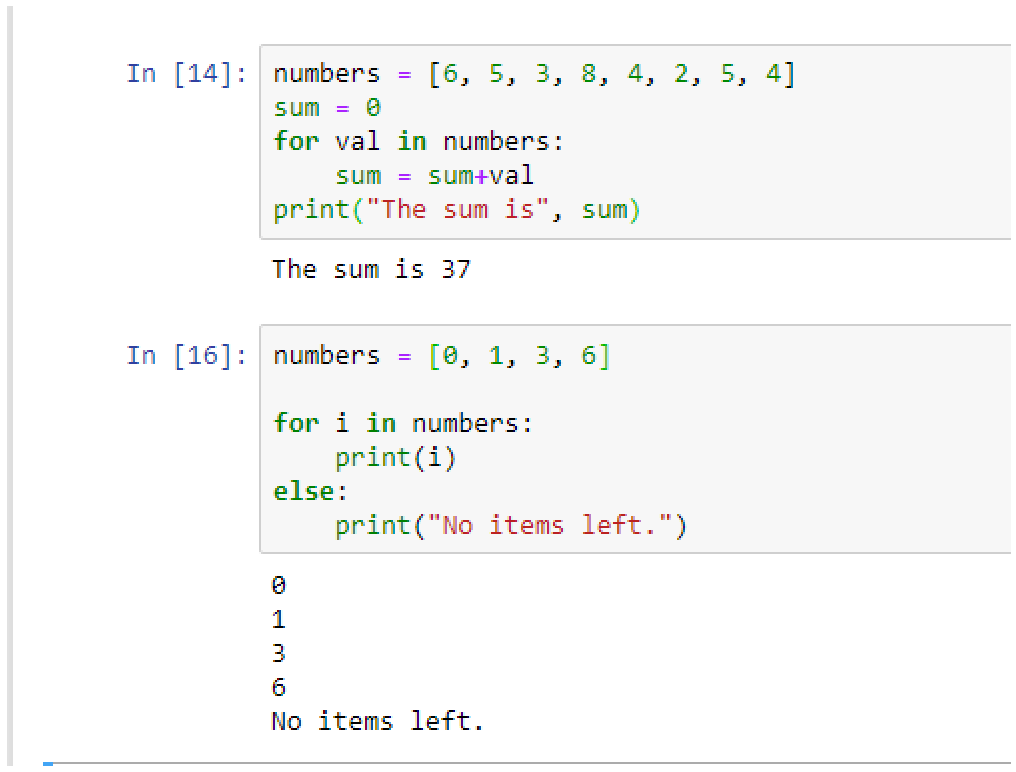
```
for value in sequence:
    ...body statement of for
```

Flow chart of for statement



Example

Check In[14] and In[16]. The continue statement is used to stop for loop, in case there is an else block missed.



```
In [14]: numbers = [6, 5, 3, 8, 4, 2, 5, 4]
         sum = 0
         for val in numbers:
             sum = sum+val
         print("The sum is", sum)
```

```
The sum is 37
```

```
In [16]: numbers = [0, 1, 3, 6]

         for i in numbers:
             print(i)
         else:
             print("No items left.")
```

```
0
1
3
6
No items left.
```

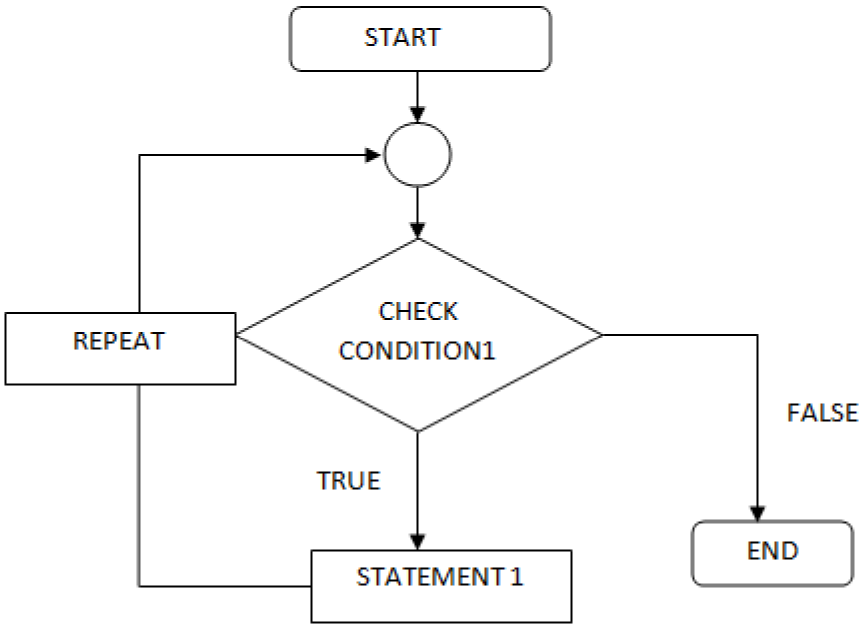
while loop

A while loop is used in python to iterate over the block of expression which matches to true. Non-zero values are True and zero and negative values are False, as interpreted in Python.

Syntax

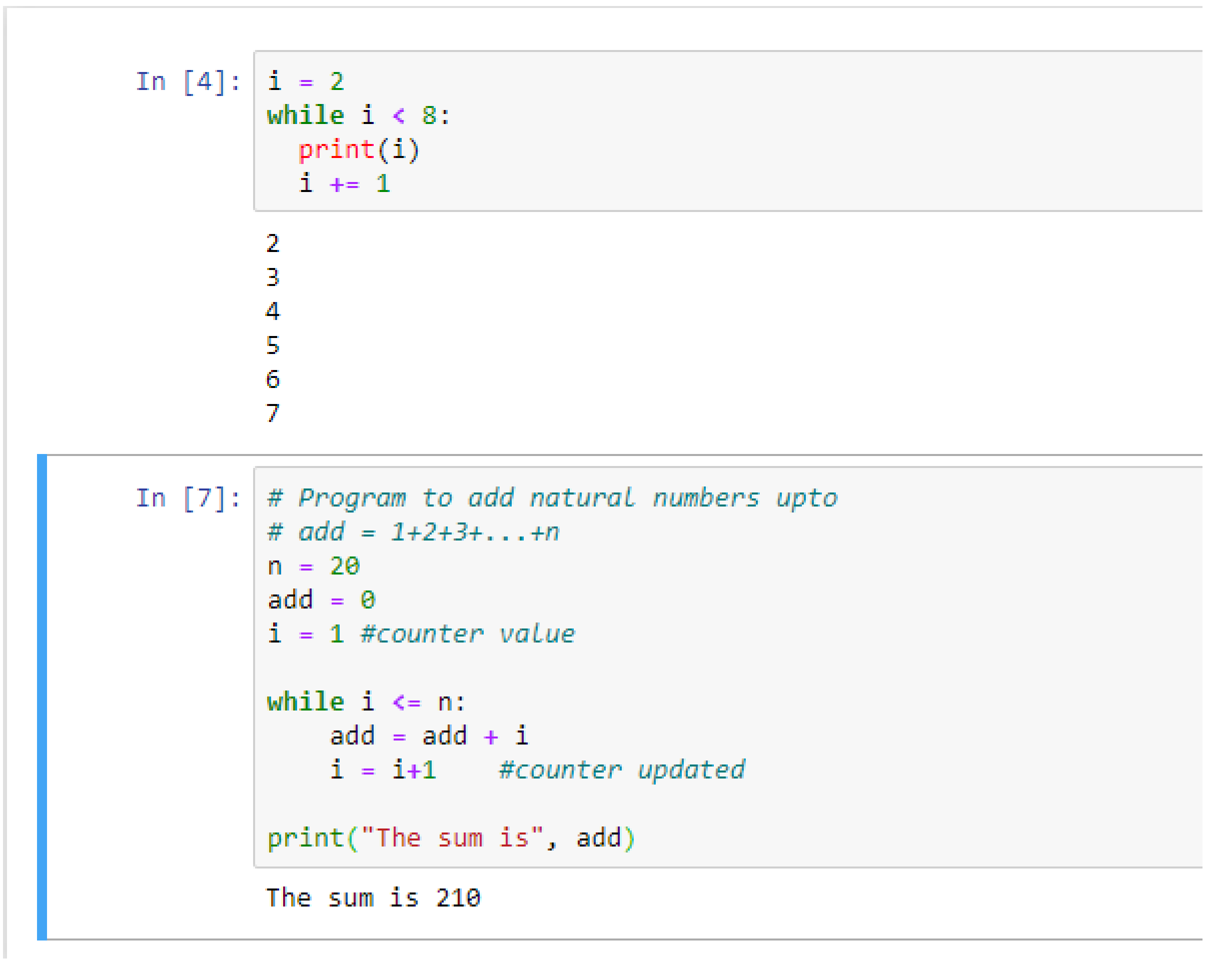
```
while(<condition>):
    statement 1..
```

Flow chart of while loop



Example

Check In[4] and In[7]. In[7]. If the user wants to add a number of his choice, then use: n = int(input("Enter number: ")) instead of n=20. Also, check-in In[3] for a while..else loop.



```
In [4]: i = 2
        while i < 8:
          print(i)
          i += 1
```

```
2
3
4
5
6
7
```

```
In [7]: # Program to add natural numbers upto
        # add = 1+2+3+...+n
        n = 20
        add = 0
        i = 1 #counter value

        while i <= n:
            add = add + i
            i = i+1     #counter updated

        print("The sum is", add)
```

```
The sum is 210
```

Break statement

The Python Break statement is used to break a loop in a certain condition. It terminates the loop. Also, we can use it inside the loop then it will first terminate the innermost loop.

Syntax

I. break

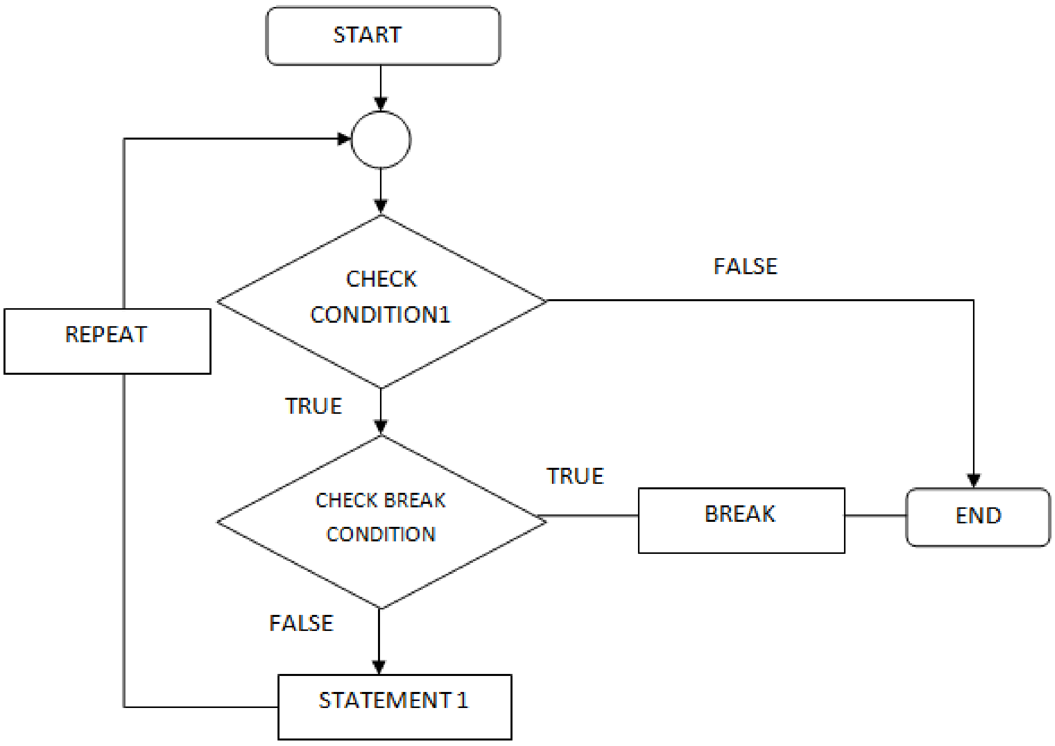
II. with for loop

```
for value in sequence:
    ...body statement of for
    if(<condition>):
       break
    ...body statement of for loop

...body statement outside of for loop
```

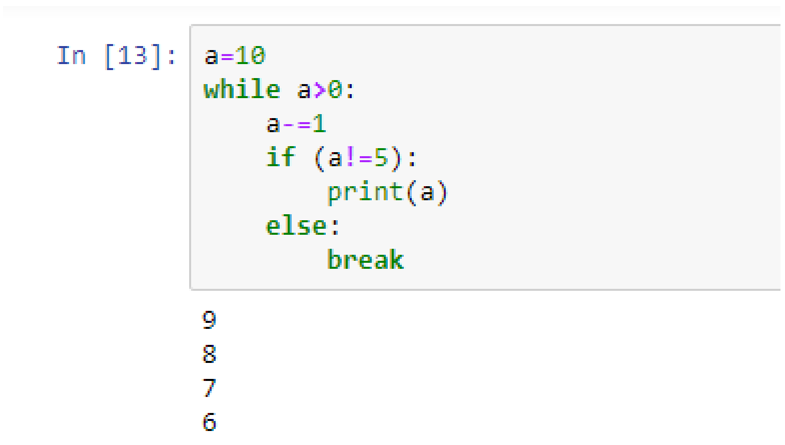
III. with a while loop

```
while(<condition>):
     statement 1...
    if(<condition>):
          break
    ...Statement of while loop

....Statements outside while loop
```

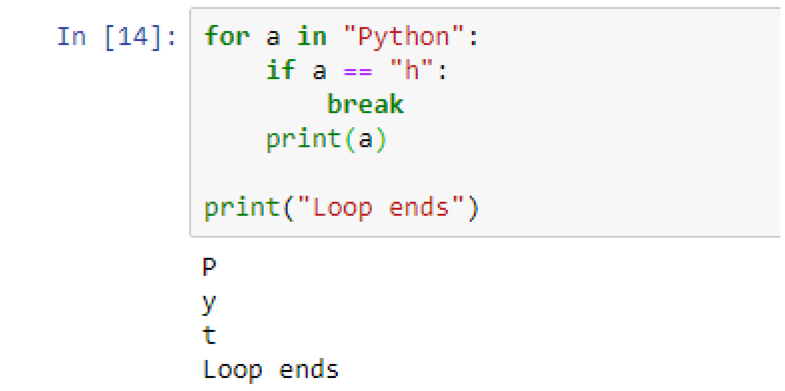
Break statement Flow Chart



Example



```
In [13]: a=10
         while a>0:
             a-=1
             if (a!=5):
                 print(a)
             else:
                 break

9
8
7
6
```



```
In [14]: for a in "Python":
             if a == "h":
                 break
             print(a)

         print("Loop ends")

P
y
t
Loop ends
```

Continue Statement

A continue statement won't continue the loop, it executes statements until the condition matches True.

Syntax

I. continue

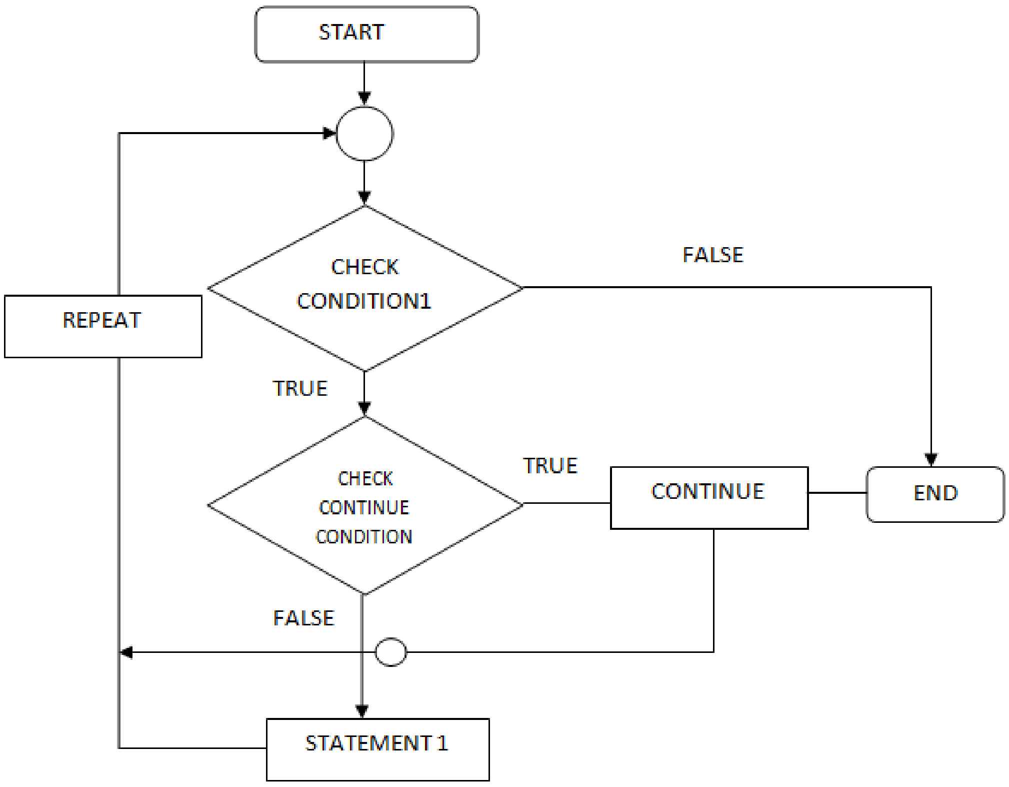
II. with for loop

```
for value in sequence:
    ...body statement of for
    if(<condition>):
        continue
    ...body statement of for loop

...body statement outside of for loop
```

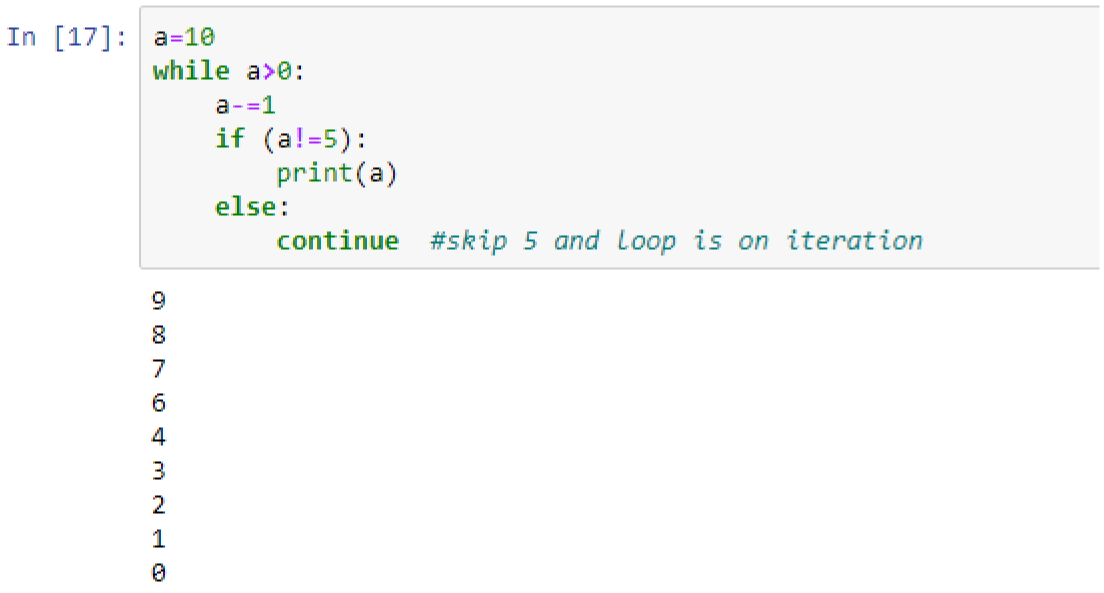
III. with the while loop

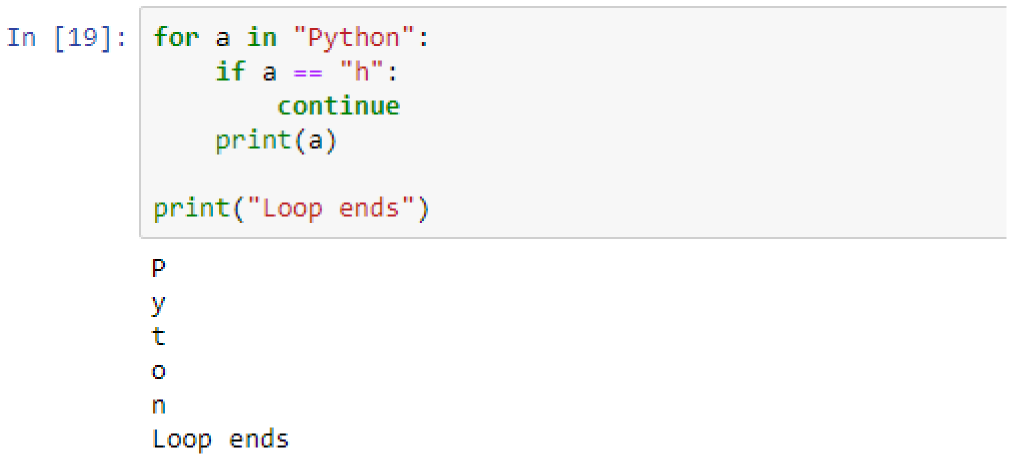
```
while(<condition>):
    statement 1...
    if(<condition>):
        continue
    ...Statement of while loop

...Statements outside while loop
```

Continue statement Flow Chart



Example


In [17]:
```python
a=10
while a>0:
    a-=1
    if (a!=5):
        print(a)
    else:
        continue  #skip 5 and loop is on iteration
```
```
9
8
7
6
4
3
2
1
0
```

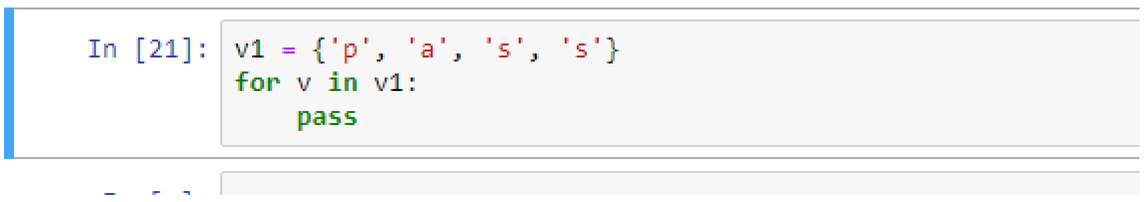

In [19]:
```python
for a in "Python":
    if a == "h":
        continue
    print(a)

print("Loop ends")
```
```
P
y
t
o
n
Loop ends
```

One more additional Flow statement is PASS.

PASS

In Python, pass, and comment both are quite similar. The pass contains a Null value. The Python interpreter ignores the comment statement, but it’s not possible to ignore a pass statement. There is nothing is going to execute when a pass is used. It is used as a Place Holder in a loop or a function.

Example



```
In [21]: v1 = {'p', 'a', 's', 's'}
         for v in v1:
             pass
```

It executes nothing.

2. Write a Program to Prompt for a Score between 0.0 and 1.0. If the Score is out of range, print an error. If the score is between 0.0 and 1.0, print a grade using the following table

```
score =
input("Enter
Score: ")
               s =  float(score)
               x = 'Error'
               if s >= 0.9:
                       x = 'A'
               elif s >=0.8:
                       x='B'
               elif s >=0.7:
                       x='C'
               elif s >= 0.6:
                       x='D'
               elif s < .6:
                       x ='F'
               else:
                       x ="Out of Range"
               print (x)
```

3. Write a program to display the fibonacci sequences up to nth term where n is provided by the user.

```
# take input from the user
nterms = int(input("How many terms? "))

# first two terms
n1 = 0
n2 = 1
count = 2

# check if the number of terms is valid
if nterms <= 0:
   print("Plese enter a positive integer")
elif nterms == 1:
   print("Fibonacci sequence:")
   print(n1)
else:
```

```
print("Fibonacci sequence:")
print(n1,",",n2,end=', ')
while count < nterms:
    nth = n1 + n2
    print(nth,end=' , ')
    # update values
    n1 = n2
    n2 = nth
    count += 1
```

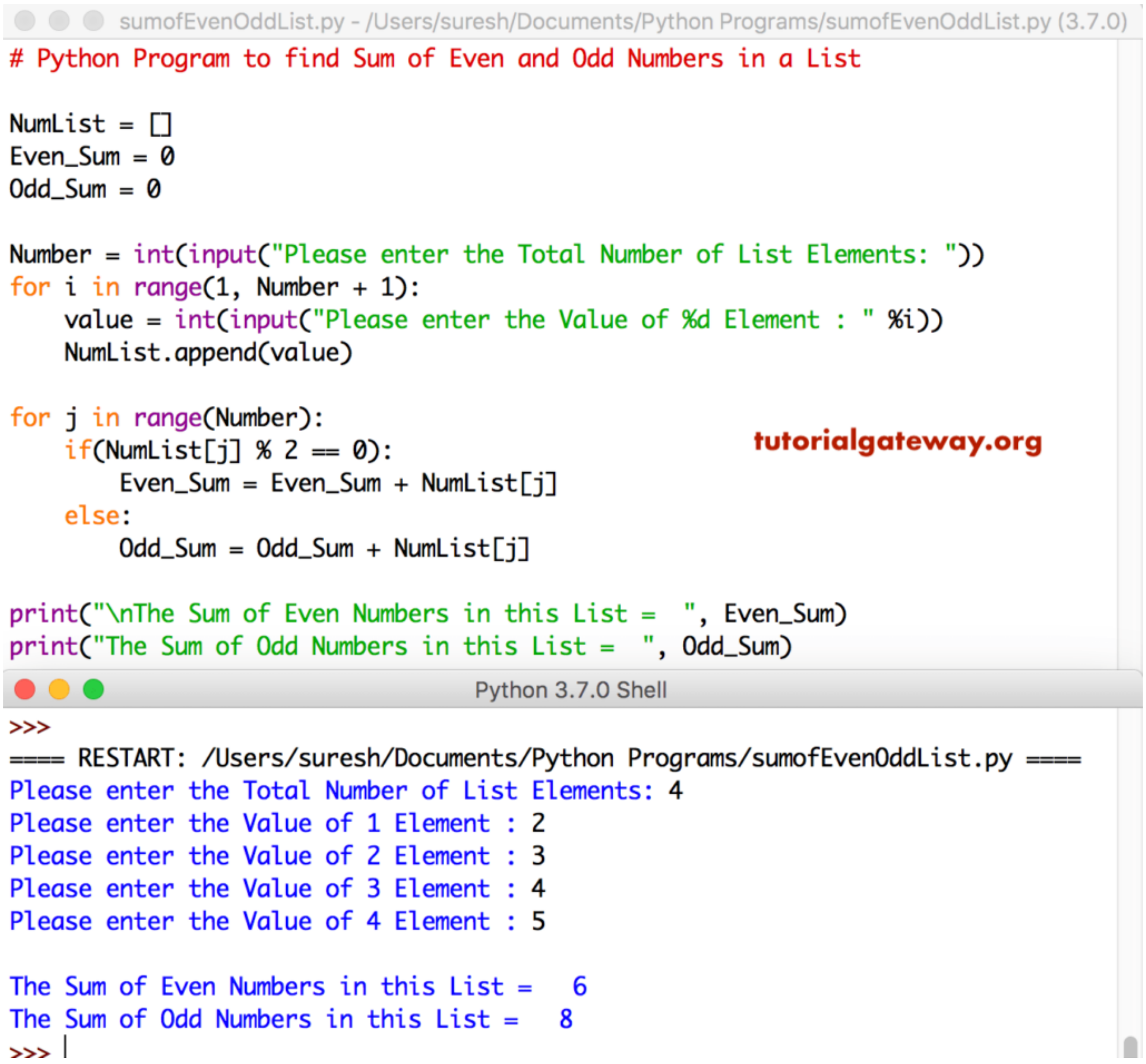
4.Write a program to repeatedly check for the largest number until the user enters “done”

```
largest =
None
smallest = None

while True:
    try:
        num = raw_input("Enter a number: ")
        if num == 'done':
            break;
        n = int(num)
        largest = num if largest < num or largest == None else
largest
        smallest = num if smallest > num or smallest == None else
smallest
    except:
        print "Invalid input"

print "Maximum number is ", largest
print "Minimum number is ", smallest
```

o/p:

#Enter 7,2,bob,10,4 u will get desired output

5. **Write a program to find the sum of all Odd and Even numbers up to a number specified by the user.**



sumofEvenOddList.py - /Users/suresh/Documents/Python Programs/sumofEvenOddList.py (3.7.0)

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

NumList = []
Even_Sum = 0
Odd_Sum = 0

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

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

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

Python 3.7.0 Shell

```
>>>
==== RESTART: /Users/suresh/Documents/Python Programs/sumofEvenOddList.py ====
Please enter the Total Number of List Elements: 4
Please enter the Value of 1 Element : 2
Please enter the Value of 2 Element : 3
Please enter the Value of 3 Element : 4
Please enter the Value of 4 Element : 5

The Sum of Even Numbers in this List =   6
The Sum of Odd Numbers in this List =   8
>>> |
```

6. **Explain the need for continue and break statements. Write a program to check whether a number is prime or not. Prompt the user for input**

```
num = int(input("enter a number: "))


for i in range(2, num):

    if num % i  == 0:
        print("not prime number")
        break
else:
    print("prime number")
```

7. Describe the syntax for the following functions and explain with an example. a) abs() b) max() c) divmod() d) pow() e) len()

- `abs()` for absolute value
- `divmod()` to find a quotient and remainder simultaneously

- `pow()` to raise a number to a certain power
- The `max()` function returns the largest of the input values.

- Its syntax is as follows:

- `max(iterable[, default=obj, key=func]) -> value`

- The `len()` function returns the number of items in an object.

- When the object is a string, the `len()` function returns the number of characters in the string.

8. Write Pythonic code to solve the quadratic equation ax2 + bx + c = 0 by getting input for coefficients from the user.**

```
# Solve the quadratic equation ax**2 + bx + c = 0

# import complex math module
import cmath

a = 1
b = 5
c = 6

# calculate the discriminant
d = (b**2) - (4*a*c)

# find two solutions
sol1 = (-b-cmath.sqrt(d))/(2*a)
sol2 = (-b+cmath.sqrt(d))/(2*a)

print('The solution are {0} and {1}'.format(sol1,sol2))
```
Run Code

Output

```
Enter a: 1
Enter b: 5
Enter c: 6
The solutions are (-3+0j) and (-2+0j)
```

9.Find the area and perimeter of a circle using functions. Prompt the user for input.

```
def areaperi(rad) :

   area = 3.14 * (rad**2)
```

```
pr = 2*3.14*rad

return area,pr

radius = int(input("Enter the radius of circle :"))

area,perimeter = areaperi(radius)
```

10. Write a Python program using functions to find the value of nPr and nCr without using inbuilt factorial() function

```
import math;
print("Enter 'x' for exit.");
nval = input("Enter value of n: ");
if nval == 'x':
    exit();
else:
    rval = input("Enter value of r: ");
    n = int(nval);
    r = int(rval);
    npr = math.factorial(n)/math.factorial(n-r);
    ncr = npr/math.factorial(r);
    print("ncR =",ncr);
    print("nPr =",npr);
```

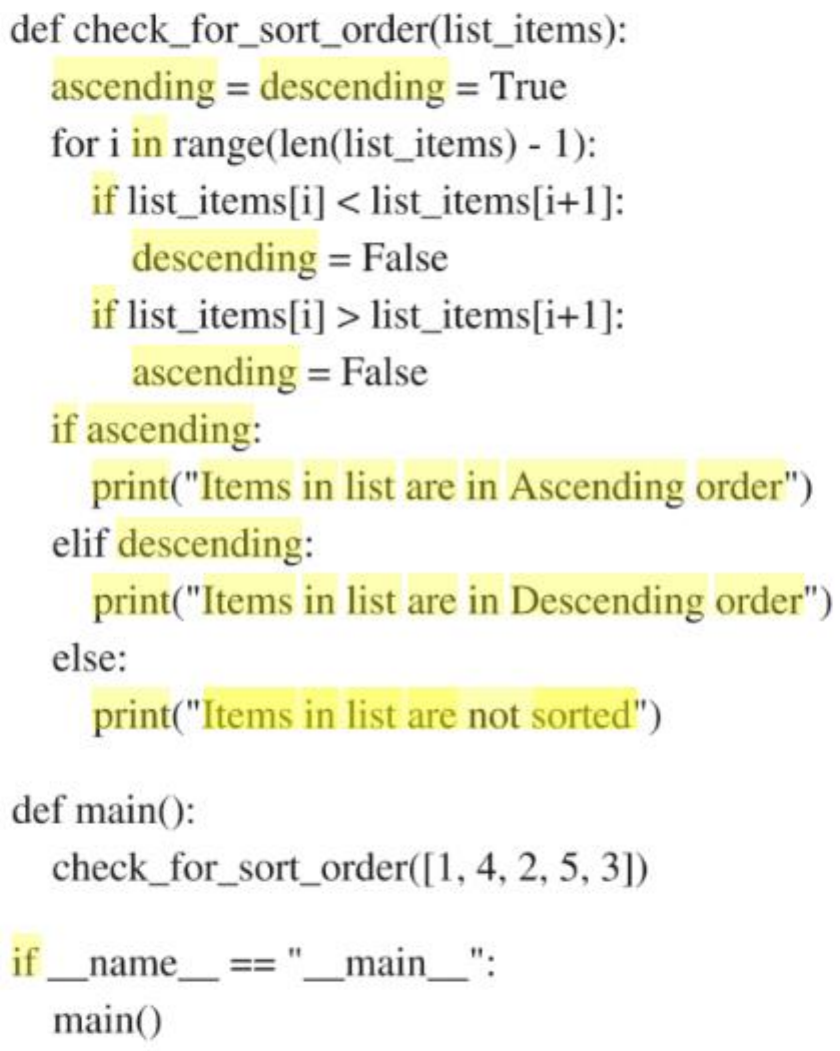
11.Write a program to print the sum of the following series 1 + 1/2 + 1/3 +. …. + 1/n

```
# Python program to find the sum of series

def sum(n):
    i = 1
    s = 0.0
    for i in range(1, n+1):
        s = s + 1/i;
    return s;

# Driver Code
n = 5
print("Sum is", round(sum(n), 6))
```

12. Write a function which receives a variable number of strings as arguments. Find unique characters in each string

```
def is_unique_1(string):

    for i in range(0, len(string)):              # O(n) time
        for j in range(i+1, len(string)):        # O(n) time
            if (string[i] ==  string[j]):
                print "duplicates found."; return

    print "all unique."; return
```

13. Check if the items in the list are sorted in ascending or descending order and print suitable messages accordingly. Otherwise, print "Items in list are not sorted"



```
def check_for_sort_order(list_items):
    ascending = descending = True
    for i in range(len(list_items) - 1):
        if list_items[i] < list_items[i+1]:
            descending = False
        if list_items[i] > list_items[i+1]:
            ascending = False
    if ascending:
        print("Items in list are in Ascending order")
    elif descending:
        print("Items in list are in Descending order")
    else:
        print("Items in list are not sorted")

def main():
    check_for_sort_order([1, 4, 2, 5, 3])

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

14. Write Pythonic code to multiply two matrices using nested loops and also perform transpose of the resultant matrix.

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

       [4,5,6],

       [7,8,9]]


Y = [[10,11,12],

      [13,14,15],

      [16,17,18]]
```

```
result = [[0,0,0],

               [0,0,0],

              [0,0,0]]


# iterate through rows of X

for i in range(len(X)):

   for j in range(len(Y[0])):

       for k in range(len(Y)):

           result[i][j] += X[i][k] * Y[k][j]

for r in result:

   print(r)
```

Output

```
The result:
[84, 90, 96]
[201, 216, 231]
[318, 342, 366]
```

15. Write Python program to sort words in a sentence in decreasing order of their length. Display the sorted words along with their length

16. Write Pythonic code to create a function called most_frequent that takes a string and prints the letters in decreasing order of frequency. Use dictionaries.

```
def most_frequent(string):
    d = dict()
    for key in string:
        if key not in d:
            d[key] = 1
        else:
            d[key] += 1
    return d

print most_frequent('aabbbc')
```

Returning:

{'a': 2, 'c': 1, 'b': 3}

UNIT-3

MCQ

1. To read the entire remaining contents of the file as a string from a file object infile, we use ____________

a) infile.read

b) infile.read()

c) infile.readline()

d) infile.readlines()

Answer: b Explanation: read function is used to read all the lines in a file.

2. Which are the two built-in functions to read a line of text from standard input, which by default comes from the keyboard?

a) Raw_input & Input

b) Input & Scan

c) Scan & Scanner

d) Scanner View

Answer: a

Explanation: Python provides two built-in functions to read a line of text from standard input, which by default comes from the keyboard. These functions are: raw_input and input

3. Which one of the following is not attributes of file?

a) closed

b) softspace

c) rename

d) mode

Answer: c

Explanation: rename is not the attribute of file rest all are files attributes

4. What is the use of seek() method in files?

a) sets the file's current position at the offset

b) sets the file's previous position at the offset

c) sets the file's current position within the file

d) none of the mentioned

Answer: a

Explanation: Sets the file's current position at the offset. The method seek() sets the file's current position at the offset.

5.What is the use of truncate() method in file?

a) truncates the file size

b) deletes the content of the file

c) deletes the file size

d) none of the mentioned

Answer: a

Explanation: The method truncate() truncates the file size. Following is the syntax for truncate() method: fileObject.truncate([size])

6. What is the pickling?

a) It is used for object serialization

b) It is used for object deserialization

c) None of the mentioned

d) All of the mentioned

Answer: a

7. What does the function re.match do?

a) matches a pattern at the start of the string

b) matches a pattern at any position in the string

c) such a function does not exist

d) none of the mentioned

Answer: a Explanation: It will look for the pattern at the beginning and return None if it isn't found.

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

sentence = 'horses are fast'

regex = re.compile('(?P\w+) (?P\w+) (?P\w+)') matched = re.search(regex, sentence) print(matched.group(2))

a) {'animal': 'horses', 'verb': 'are', 'adjective': 'fast'}

b) ('horses', 'are', 'fast')

c) 'horses are fast'

d) 'are'

9. The character Dot (that is, ‘.’) in the default mode, matches any character other than _____________

a) caret

b) ampersand

c) percentage symbol

d) newline

Answer: d

Explanation: The character Dot (that is, ‘,’) in the default mode, matches any character other than newline. If DOTALL flag is used, then it matches any character other than newline.

10. The expression a{5} will match _____________ characters with the previous regular expression.
a) 5 or less

b) exactly 5

c) 5 or more

d) exactly 4

Answer: b

Explanation: The character {m} is used to match exactly m characters to the previous regular expression. Hence the expression a{5} will match exactly 5 characters and not less than that.

11. ________ matches the start of the string. ________ matches the end of the string

. a) ‘^’, ‘$’

b) ‘$’, ‘^’

c) ‘$’, ‘?’

d) ‘?’, ‘^’ Answer: a Explanation: ‘^’ (carat) matches the start of the string. ‘$’ (dollar sign) matches the end of the string.

12.What will be the output of the following Python function? re.findall("hello world", "hello", 1) a) [“hello”] b) [ ] c) hello d) hello world View Answer Answer: b Explanation: The function findall returns the word matched if and only if both the pattern and the string match completely, that is, they are exactly the same.

13. . What will be the output of the following Python code? re.sub('morning', 'evening', 'good morning')

a)‘good evening’
b)‘good’
c)‘morning’
d)‘evening’
View Answer
Answer:a
Explanation: The code shown above first searches for the pattern ‘morning’ in the string

‘good morning’ and then replaces this pattern with ‘evening’. Hence the output of this code is: ‘good evening’.

14. What will be the output of the following Python code? re.split('[a-c]', '0a3B6', re.I)

a) Error

b) [‘a’, ‘B’]

c) [‘0’, ‘3B6’]

d) [‘a’]

Answer: c Explanation: The function re.split() splits the string on the basis of the pattern given in the parenthesis. Since we have used the flag e.I (that is, re.IGNORECASE), the output is: [‘0’, ‘3B6’].

15. 4. Which of the following pattern matching modifiers permits whitespace and comments inside the regular expression?

a) re.L

b) re.S

c) re.U d) re.X

Answer: d Explanation: The modifier re.X allows whitespace and comments inside the regular expressions.

16. Which of the following special characters matches a pattern only at the end of the string? a) \B b) \X c) \Z d) \A

Ans- c) \Z

17. Which of the following special characters represents a comment (that is, the contents of the parenthesis are simply ignores)? a) (?:…) b) (?=…) c) (?!…) d) (?#…)

Ans- d) (?#...)

18. _____ represents an entity in the real world with its identity and behaviour. a) A method b) An object c) A class d) An operation

Ans- b)An Object

19. What is setattr() used for? a) To access the attribute of the object b) To set an attribute c) To check if an attribute exists or not d) To delete an attribute

Ans- b) To set an attribute

20. The assignment of more than one function to a particular operator is _______ a) Operator over-assignment b) Operator overriding c) Operator overloading d) Operator instance

Ans- c)Operator overloading

21. What are the methods which begin and end with two underscore characters called? a) Special methods b) In-built methods c) User-defined methods d) Additional methods

Ans- a) Special Methods

22. What is hasattr(obj,name) used for? a) To access the attribute of the object b) To delete an attribute c) To check if an attribute exists or not d) To set an attribute

Ans- c)To check if an attribute exists or not

23. Which of these is not a fundamental features of OOP? a) Encapsulation b) Inheritance c) Instantiation d) Polymorphism

Ans- c) Instantiation

24. Methods of a class that provide access to private members of the class are called as ______ and ______ a) getters/setters b) __repr__/__str__ c) user-defined functions/in-built functions d) __init__/__del__

Ans- a)getters/setters

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

Ans- d)more than zero

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

Ans- c)when no execution occurs

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

Ans-a)yes, like except TypeError, SyntaxError [,…]

28. What happens when '1' == 1 is executed? a) we get a True b) we get a False c) an TypeError occurs d) a ValueError occurs

Ans- b)we get a false

29. What will be the output of the following Python code?g = (i for i in range(5))type(g)a) class <'loop'> b) class <'iteration'>c) class <'range'> d) class <'generator'>

Ans- d)class <'generator'>

30. What will be the output of the following Python code? lst = [1, 2, 3] lst[3] a) NameError b) ValueError c) IndexError d) TypeError

Ans- c) IndexError

Short Answer

1.Explain split(), sub(), subn() methods of “re” module in Python.

Ans- The re module in python refers to the module Regular Expressions (RE). It specifies a set of strings or patterns that matches it. Metacharacters are used to understand the analogy of RE.

Function split()

This function splits the string according to the occurrences of a character or a pattern. When it finds that pattern, it returns the remaining characters from the string as part of the resulting list. The split method should be imported before using it in the program.

Sub()

Syntax: re.split (pattern, string, maxsplit=0, flags=0)

re.sub (pattern, repl, string, count=0, flags=0)

This function stands for the substring in which a certain regular expression pattern is searched in the given string (3rd parameter)

Subn()

re.subn (pattern, repl, string, count=0, flags=0)

This function is similar to sub() in all ways except the way in which it provides the output. It returns a tuple with count of total of all the replacements as well as the new string.

2. What are Python packages?

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

3.How can files be deleted in Python?

In **Python**, you **can** use the os. remove() method **to** remove **files**, and the os. rmdir() method **to delete** an empty folder. If you want **to delete** a folder with all of its **files**

4. Does Python have OOps concepts?

Major **OOP** (**object-oriented programming**) **concepts** in **Python** include Class, Object, Method, Inheritance, Polymorphism, Data Abstraction, and Encapsulation

5. Explain Inheritance in Python with an example.

Inheritance is a powerful feature in object oriented programming. It refers to defining a new class with little or no modification to an existing class. The new class is called derived (or child) class and the one from which it **inherits** is called the base (or parent) class.

6. How are classes created in Python?

Like function definitions begin with the def keyword in **Python**, **class** definitions begin with a **class** keyword. ... As soon as we define a **class**, a new **class** object is **created** with the same name. This **class** object allows us to access the different attributes as well as to instantiate new objects of that **class**.

7.Does python support multiple inheritance?

8. What is Polymorphism in Python?

In **Python**, **Polymorphism** allows us to define methods in the child class with the same name as defined in their parent class. As we know, a child class inherits all the methods from the parent class.

9. Define encapsulation in Python?

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of wrapping data and the methods that work on data within one unit. This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data. To prevent accidental change, an object's variable can only be changed by an object's method. Those type of variables are known as **private variable**.

10. How do you do data abstraction in Python?

Abstraction in Java and Python is a programming methodology in which details of the programming codes are hidden away from the user, and only the essential things are displayed to the user. Abstraction is concerned with ideas rather than events. It's like a user running a program (Web Browser) without seeing the background codes. Abstraction is achieved in either Abstract classes or interface in Java and Python. NetBeans and Eclipse IDE implements abstraction for Java while Django implements abstraction for Python.

11. Does python make use of access specifiers?

Python does not have **access modifiers**. If you want to **access** an instance (or class) variable from outside the instance or class, you are always allowed to **do** so. The single underscore prefix for a member variable or method **is** a commonly used convention to denote a private method.

12. What does an object() do?

The `object()` function returns an empty object.

You cannot add new properties or methods to this object.

This object is the base for all classes, it holds the built-in properties and methods which are default for all classes.

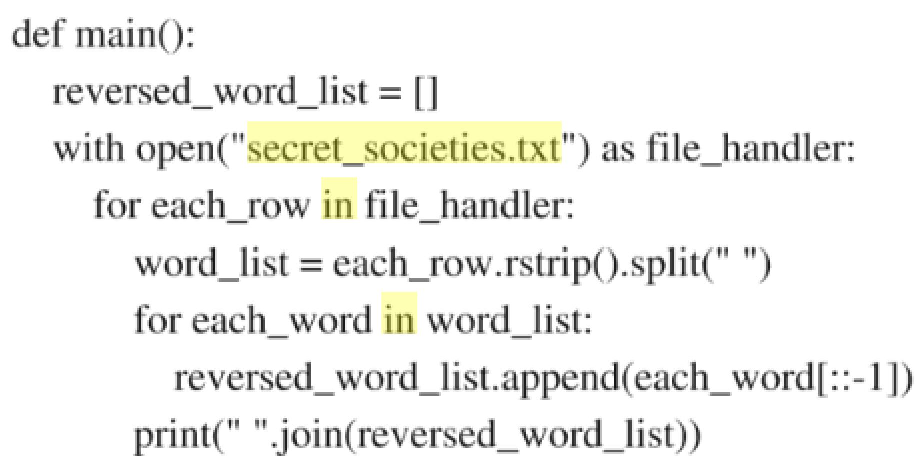
Syntax

```
object()
```

LONG ANSWER

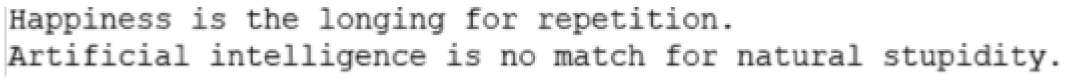
CHAPTER-3 {LONG}

1.Write Python Program to Reverse Each Word in "secret_societies.txt" file

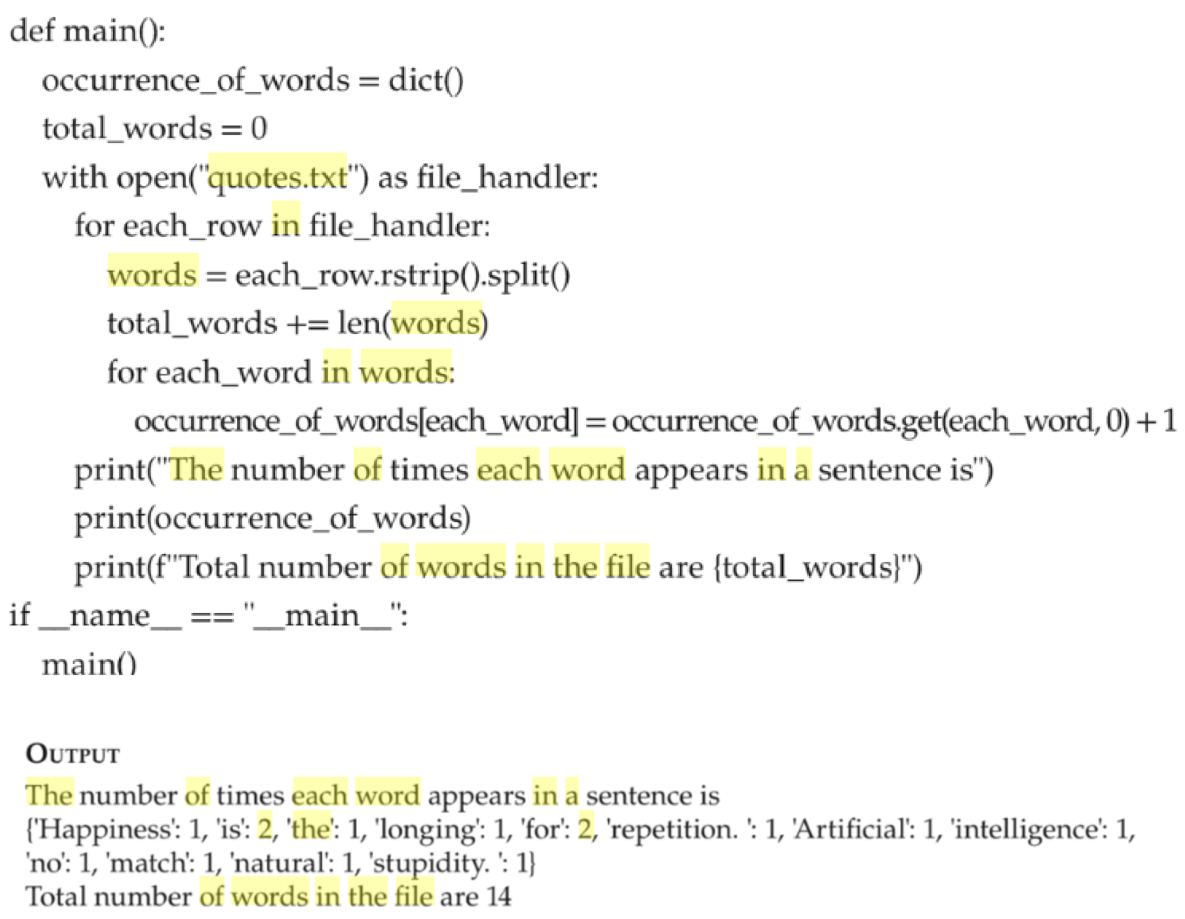


```
def main():
    reversed_word_list = []
    with open("secret_societies.txt") as file_handler:
        for each_row in file_handler:
            word_list = each_row.rstrip().split(" ")
            for each_word in word_list:
                reversed_word_list.append(each_word[::-1])
            print(" ".join(reversed_word_list))
```

2. Write Python Program to Count the Occurrences of Each Word and Also Count the Number of Words in a "quotes.txt" File.



```
Happiness is the longing for repetition.
Artificial intelligence is no match for natural stupidity.
```



```
def main():
    occurrence_of_words = dict()
    total_words = 0
    with open("quotes.txt") as file_handler:
        for each_row in file_handler:
            words = each_row.rstrip().split()
            total_words += len(words)
            for each_word in words:
                occurrence_of_words[each_word] = occurrence_of_words.get(each_word, 0) + 1
        print("The number of times each word appears in a sentence is")
        print(occurrence_of_words)
        print(f"Total number of words in the file are {total_words}")
if __name__ == "__main__":
    main()
```

OUTPUT

```
The number of times each word appears in a sentence is
{'Happiness': 1, 'is': 2, 'the': 1, 'longing': 1, 'for': 2, 'repetition. ': 1, 'Artificial': 1, 'intelligence': 1,
'no': 1, 'match': 1, 'natural': 1, 'stupidity. ': 1}
Total number of words in the file are 14
```

3. Write Python Program to Find the Longest Word in a File. Get the File Name from User.

A quick brown fox jumps over the lazy dog

```
def longest_words(filename):

    with open(filename, 'r') as infile:

        words = infile.read().split()

    max_len = len(max(words, key=len))

    return [word for word in words if len(word) == max_len]



print(longest_words('about.txt'))
```

The output of the above program is:-

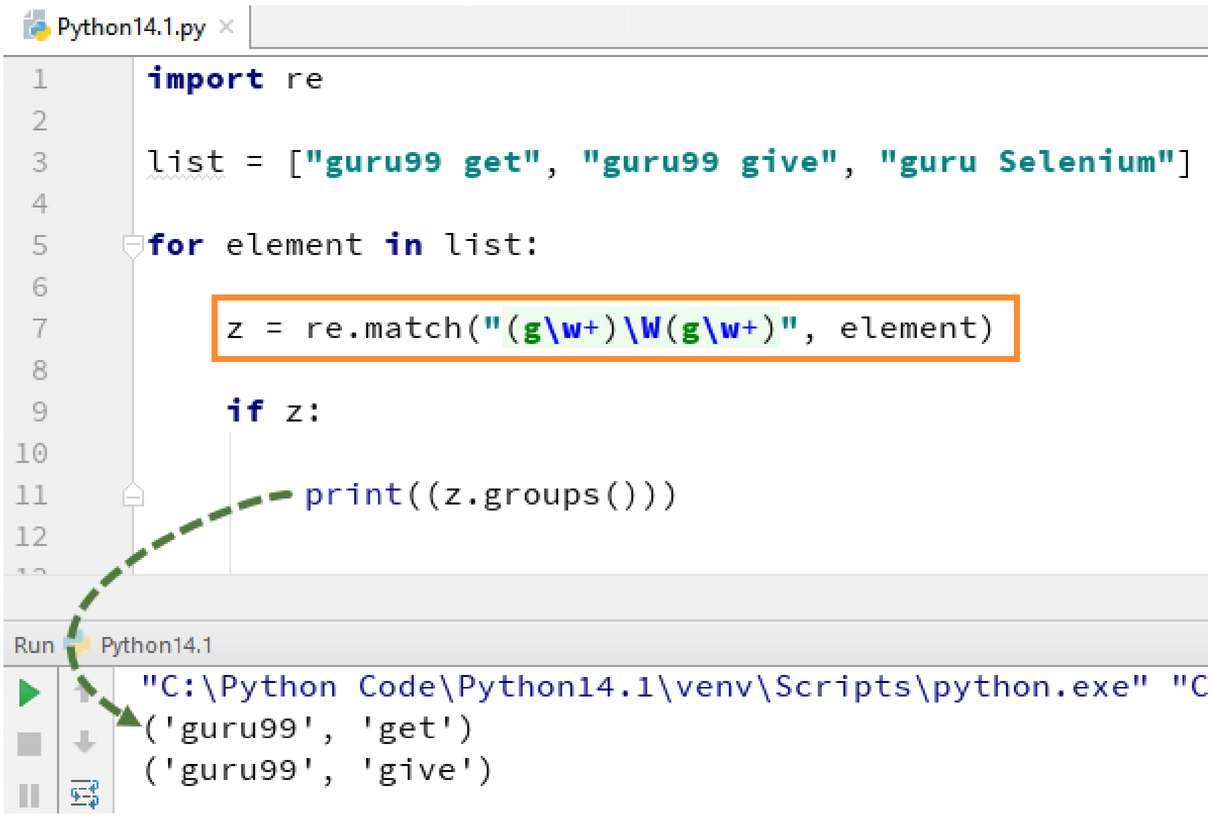
```
['quick', 'brown', 'jumps']
```

4. Discuss the following methods supported by compiled regular expression objects. a) search() b) match() c) findall()

re.match()

re.match() function will search the regular expression pattern and return the first occurrence. This method checks for a match only at the beginning of the string. So, if a match is found in the first line, it returns the match object. But if a match is found in some other line, it returns null.

For example, consider the following code. The expression "w+" and "\W" will match the words starting with letter 'g' and thereafter, anything which is not started with 'g' is not identified. To check match for each element in the list or string, we run the forloop.



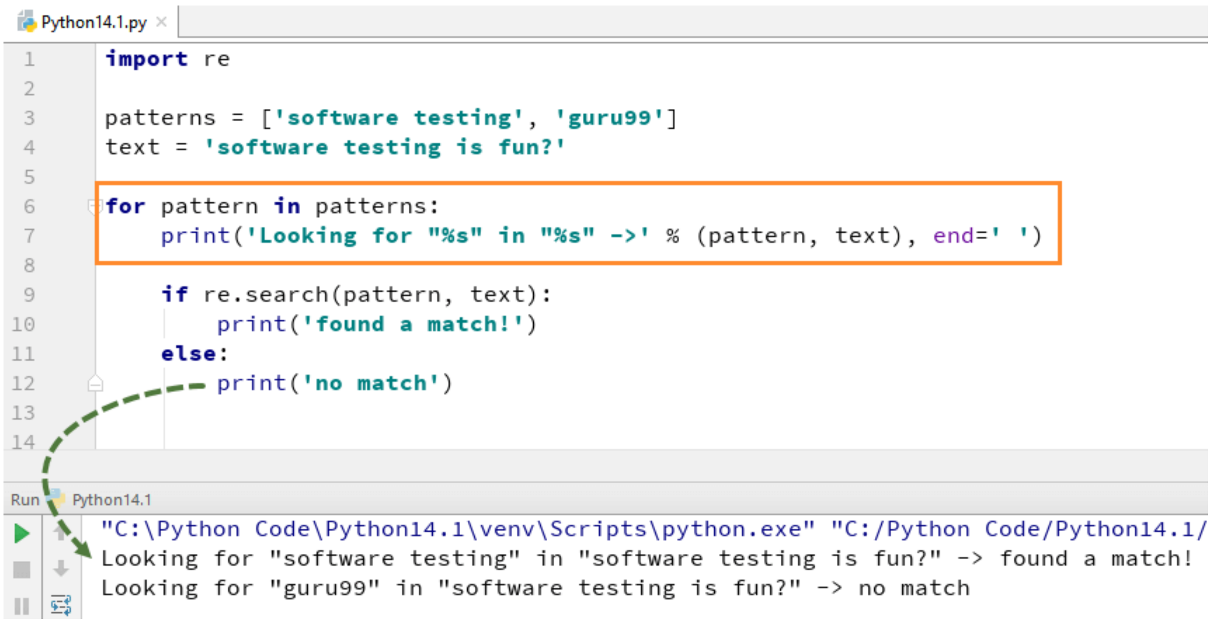
```python
import re

list = ["guru99 get", "guru99 give", "guru Selenium"]

for element in list:

    z = re.match("(g\w+)\W(g\w+)", element)

    if z:

        print((z.groups()))
```

```
"C:\Python Code\Python14.1\venv\Scripts\python.exe" "C
('guru99', 'get')
('guru99', 'give')
```

re.search(): Finding Pattern in Text

re.search() function will search the regular expression pattern and return the first occurrence. Unlike re.match, it will check all lines of the input string. It returns a match object when the pattern is found and “null” if the pattern is not found

In order to use search() function, you need to import re first and then execute the code. The search() function takes the "pattern" and "text" to scan from our main string



```
import re

patterns = ['software testing', 'guru99']
text = 'software testing is fun?'

for pattern in patterns:
    print('Looking for "%s" in "%s" ->' % (pattern, text), end=' ')

    if re.search(pattern, text):
        print('found a match!')
    else:
        print('no match')
```

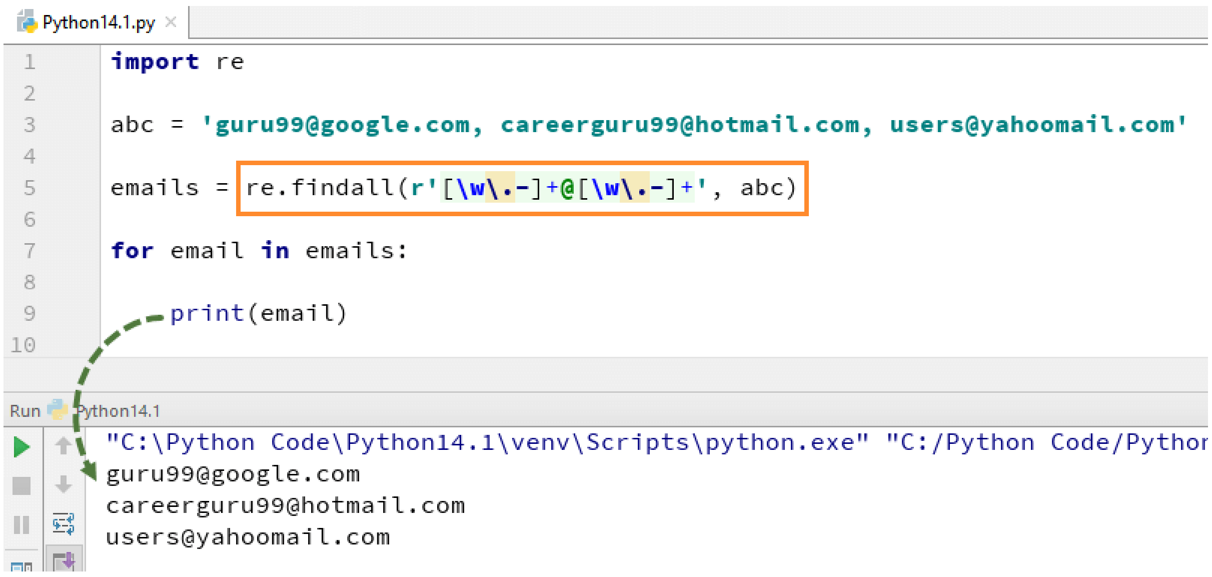
```
"C:\Python Code\Python14.1\venv\Scripts\python.exe" "C:/Python Code/Python14.1/
Looking for "software testing" in "software testing is fun?" -> found a match!
Looking for "guru99" in "software testing is fun?" -> no match
```

For example here we look for two literal strings "Software testing" "guru99", in a text string "Software Testing is fun". For "software testing" we found the match hence it returns the output as "found a match", while for word "guru99" we could not found in string hence it returns the output as "No match".

re.findall()

findall() module is used to search for “all” occurrences that match a given pattern. In contrast, search() module will only return the first occurrence that matches the specified pattern. findall() will iterate over all the lines of the file and will return all non-overlapping matches of pattern in a single step.

For example, here we have a list of e-mail addresses, and we want all the e-mail addresses to be fetched out from the list, we use the re.findall method. It will find all the e-mail addresses from the list.



```
import re

abc = 'guru99@google.com, careerguru99@hotmail.com, users@yahoomail.com'

emails = re.findall(r'[\w\.-]+@[\w\.-]+', abc)

for email in emails:

    print(email)
```

```
"C:\Python Code\Python14.1\venv\Scripts\python.exe" "C:/Python Code/Pythor
guru99@google.com
careerguru99@hotmail.com
users@yahoomail.com
```

5. Consider a line "From stephen.marquard\@uct.ac.za Sat Jan 5 09:14:16 2008" in the file email.txt. Write Pythonic code to read the file and extract email address from the lines starting from the word "From". Use regular expressions to match email address.

```
import re
hand = open('email.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^From:.+@', line):
        print(line)
o/p:- From: stephen.marquard@uct.ac.za
```

6. Describe the need for catching exceptions using try and except statements

Ans-

example

```
try:
    a = 10
    b = 0
    print("Result of Division: " + str(a/b))
    print("No! This line will not be executed.")
except:
    print("You have divided a number by zero, which is not allowed.")


# outside the try-except blocks
```

```
print("Yo! This line will be executed.")
```

The `try` block

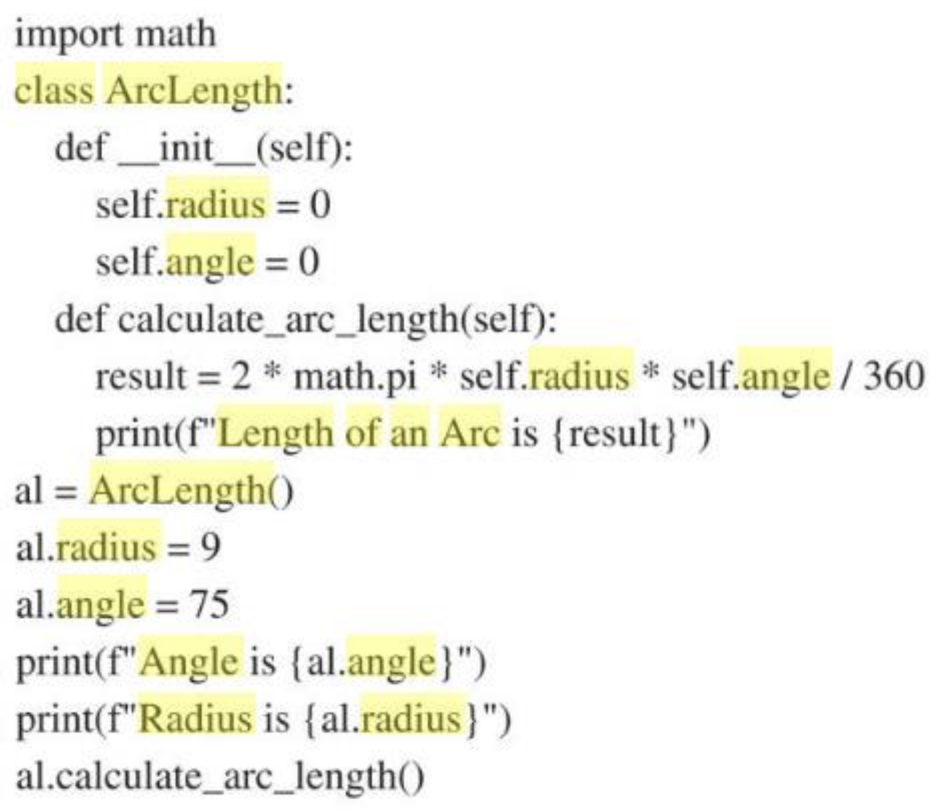
As you can see in the code example above, the `try` block is used to put the whole code that is to be executed in the program(which you think can lead to exception), if any exception occurs during execution of the code inside the `try` block, then it causes the execution of the code to be directed to the `except` block and the execution that was going on in the `try` block is interrupted. But, if no exception occurs, then the whole `try` block is executed and the `except` block is never executed.

The `except` block

The `try` block is generally followed by the `except` block which holds the exception cleanup code(*exception has occured, how to effectively handle the situation*) like some `print` statement to **print some message** or may be **trigger some event** or **store something in the database** etc.

In the **except block**, along with the keyword `except` we can also provide the **name of exception class** which is expected to occur. In case we do not provide any exception class name, it catches all the exceptions, otherwise it will only catch the exception of the type which is mentioned.

7.Write Python program to calculate the Arc Length of an Angle by assigning values to the radius and angle data attributes of the class ArcLength.



```
import math
class ArcLength:
    def __init__(self):
        self.radius = 0
        self.angle = 0
    def calculate_arc_length(self):
        result = 2 * math.pi * self.radius * self.angle / 360
        print(f"Length of an Arc is {result}")
al = ArcLength()
al.radius = 9
al.angle = 75
print(f"Angle is {al.angle}")
print(f"Radius is {al.radius}")
al.calculate_arc_length()
```

8. Write Python Program to simulate a Bank Account with support for depositMoney, withdrawMoney and showBalance Operations.

```
class BankAccount:
   def __init__(self, name):
      self.user_name = name
      self.balance = 0.0

def show_balance(self):
      print(f"{self.user_name} has a balance of {self.balance} dollars")
   def withdraw_money(self, amount):
if amount > self.balance:
         print("You don't have sufficient funds in your account")
      else:
self.balance -= amount
      print(f"{self.user_name} has withdrawn an amount of {self.balance} dollars")

   def deposit_money(self, amount):
      self.balance += amount
      print(f"{self.user_name} has deposited an amount of {self.balance} dollars")

def main():
   savings_account = BankAccount("Olivia")
   savings_account.deposit_money(1000)
   savings_account.show_balance()
savings_account.withdraw_money(500)
   savings_account.show_balance()

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

o/p:

```
Olivia has deposited an amount of 1000.0 dollars
Olivia has a balance of 1000.0 dollars
Olivia has withdrawn an amount of 500.0 dollars
Olivia has a balance of 500.0 dollars
```

9.Given three Points (x1, y1), (x2, y2) and (x3, y3), write a Python program to check if they are Collinear.

```
class Collinear:
  def __init__(self, x, y):
    self.x_coord = x
    self.y_coord = y

  def check_for_collinear(self, point_2_obj, point_3_obj):
            if    (point_3_obj.y_coord   -   point_2_obj.y_coord)*(point_2_obj.x_coord   -
self.x_coord)  ==  (point_2_obj.y_coord  -  self.y_coord)*(point_3_obj.x_coord  -
point_2_obj.x_coord):
      print("Points are Collinear")
    else:
      print("Points are not Collinear")

def main():
  point_1 = Collinear(1, 5)
  point_2 = Collinear(2, 5)
  point_3 = Collinear(4, 6)
  point_1.check_for_collinear(point_2, point_3)

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

o/p:

```
Points are Collinear
```

10.Discuss inheritance in Python programming language. Write a Python program to demonstrate the use of super() function.

Inheritance enables new classes to receive or inherit variables and methods of existing classes. Inheritance is a way to express a relationship between classes. If you want to build a new class, which is already similar to one that already exists, then instead of creating a new class from scratch you can reference the existing class and indicate what is different by overriding some of its behavior or by adding some new functionality. A class that is used as the basis for inheritance is called a *superclass* or *base class*. A class that inherits from a base class is called a *subclass* or *derived class*. The terms *parent class* and *child class* are also acceptable terms to use respectively. A derived class inherits variables and methods from its base class while adding additional variables and methods of its own. Inheritance easily enables reusing of existing code.

Class *BaseClass*, on the left, has one variable and one method. Class *DerivedClass*, on the right, is derived from *BaseClass* and contains an additional variable and an additional method (FIGURE 11.7).

If you can describe the relationship between derived classes and base class using the phrase *is-a*, then that relationship is inheritance. For example, a lemon *is-a* citrus fruit, which *is-a* fruit. A shepherd *is-a* dog, which *is-a* animal. A guitar *is-a* steel-stringed instrument, which *is-a* musical instrument. If the *is-a* relationship does not exist between a derived class and base class, you should not use inheritance.

```
class Country:
      def __init__(self, country_name):
         self.country_name = country_name
      def country_details(self):
         print(f"Happiest Country in the world is {self.country_name}")

class HappiestCountry(Country):
      def __init__(self, country_name, continent):
         super().__init__(country_name)
         self.continent = continent
      def happy_country_details(self):
          print(f"Happiest Country in the world is {self.country_name} and is in {self.continent}
")

def main():
      finland = HappiestCountry("Finland", "Europe")
      finland.happy_country_details()
if __name__ == "__main__":
```

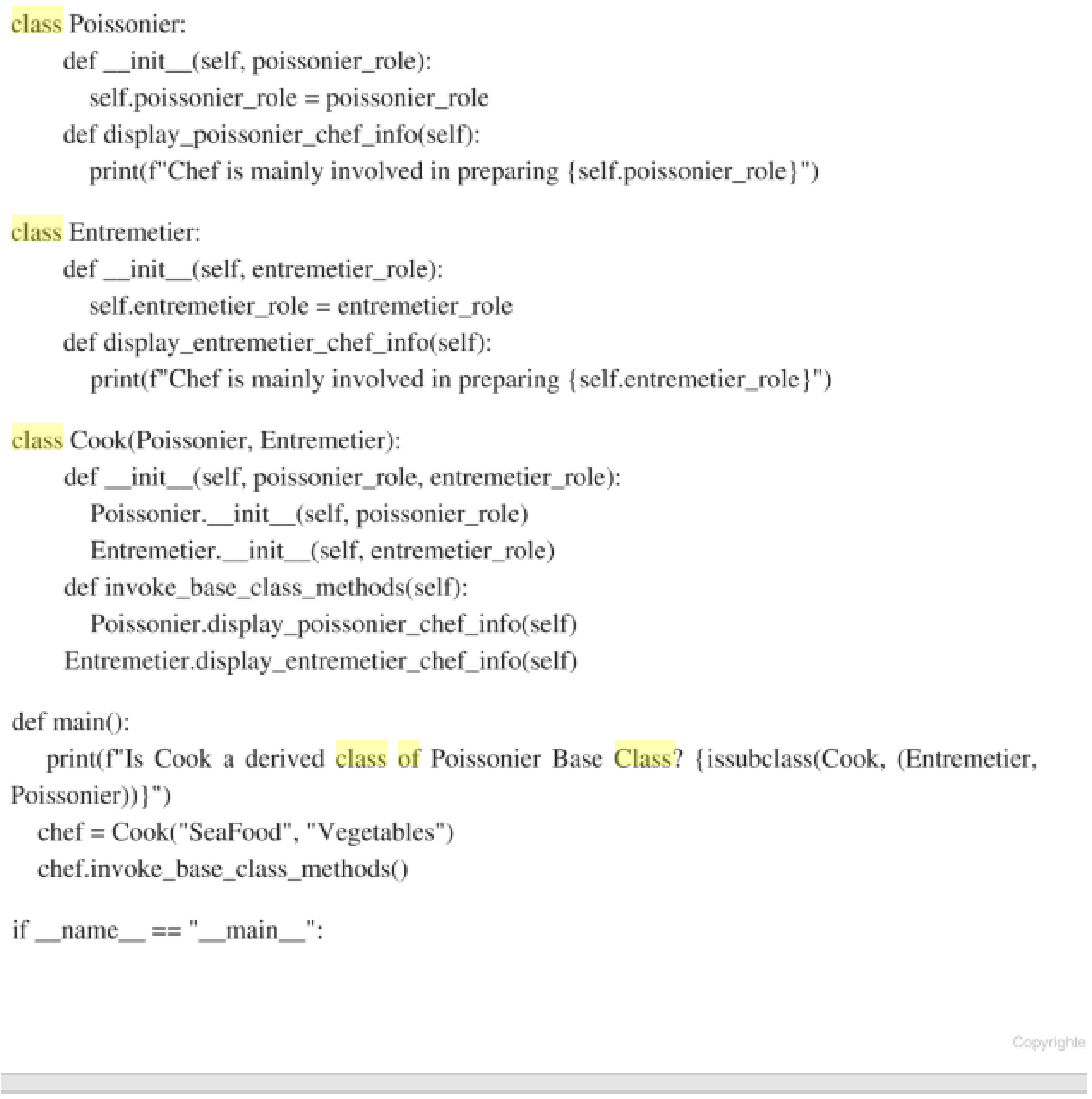
OUTPUT

```
Happiest Country in the world is Finland and is in Europe
```

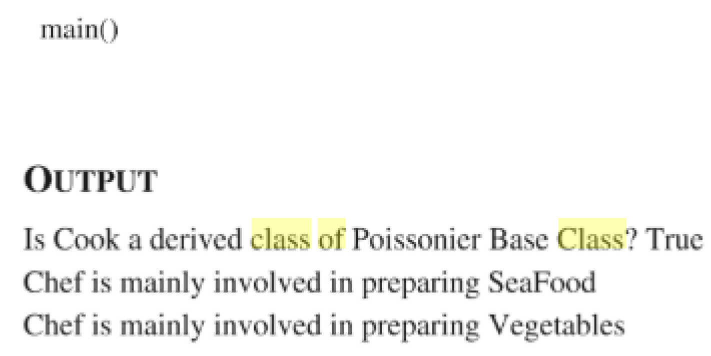
11.Program to demonstrate the Overriding of the Base Class method in the Derived Class.

```
class Book:
    def __init__(self, author, title):
        self.author = author
        self.title = title
    def book_info(self):
        print(f"{self.title} is authored by {self.author}")

class Fiction(Book):
    def __init__(self, author, title, publisher):
        super().__init__(author, title)
        self.publisher = publisher
    def book_info(self):
        print(f"{self.title} is authored by {self.author} and published by {self.publisher}")
    def invoke_base_class_method(self):
        super().book_info()
```

12.Write Python program to demonstrate Multiple Inheritance.



```
class Poissonier:
    def __init__(self, poissonier_role):
       self.poissonier_role = poissonier_role
    def display_poissonier_chef_info(self):
       print(f"Chef is mainly involved in preparing {self.poissonier_role}")

class Entremetier:
    def __init__(self, entremetier_role):
       self.entremetier_role = entremetier_role
    def display_entremetier_chef_info(self):
       print(f"Chef is mainly involved in preparing {self.entremetier_role}")

class Cook(Poissonier, Entremetier):
    def __init__(self, poissonier_role, entremetier_role):
       Poissonier.__init__(self, poissonier_role)
       Entremetier.__init__(self, entremetier_role)
    def invoke_base_class_methods(self):
       Poissonier.display_poissonier_chef_info(self)
    Entremetier.display_entremetier_chef_info(self)

def main():
    print(f"Is Cook a derived class of Poissonier Base Class? {issubclass(Cook, (Entremetier,
Poissonier))}")
   chef = Cook("SeaFood", "Vegetables")
   chef.invoke_base_class_methods()

if __name__ == "__main__":
```



```
   main()
```

OUTPUT

```
Is Cook a derived class of Poissonier Base Class? True
Chef is mainly involved in preparing SeaFood
Chef is mainly involved in preparing Vegetables
```

13. Given the Coordinates (x, y) of a center of a Circle and its radius, write Python program to determine whether the Point lies inside the Circle, on the Circle or outside the Circle.

```
class Circle:
  def __init__(self, radius=0, circle_x=0, circle_y=0, point_x=0, point_y=0):
    self.radius = radius
    self.circle_x_coord = circle_x
    self.circle_y_coord = circle_y
    self.point_x_coord = point_x
    self.point_y_coord = point_y
    self.status = ""

def check_point_status(self):
       if (self.point_x_coord - self.circle_x_coord) ** 2 + (self.point_y_coord -
self.circle_y_coord) ** 2 < self.radius ** 2:
       self.status = f"Point with coordinates {(self.point_x_coord, self.point_y_coord)} is
inside the Circle"
      elif (self.point_x_coord - self.circle_x_coord) ** 2 + (self.point_y_coord -
self.circle_y_coord) ** 2 > self.radius ** 2:
       self.status = f"Point with coordinates {(self.point_x_coord, self.point_y_coord)} is
outside the Circle"
    else:
>      self.status = f"Point with coordinates {(self.point_x_coord, self.point_y_coord)} is on
the Circle"

return self

def main():
    point = Circle(10, 2, 3, 9, 9)
    returned_object = point.check_point_status()
    print(returned_object.status)
    print(f"Is point an instance of Circle Class? {isinstance(point, Circle)}")
      print(f"Is returned_object an instance of Circle Class? {isinstance(returned_object,
Circle)}")
    print(f"Identity of the location of a point object is {id(point)}")
  print(f"Identity of the location of the returned_object object is {id(returned_object)}")

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

OUTPUT

```
Opera ends at 13:16:20
```

14.Write Python Program to Demonstrate Multiple Inheritance with Method Overriding

```
class Pet:
    def __init__(self, breed):
        self.breed = breed
    def about(self):
        print(f"This is {self.breed} breed")

class Insurable:
    def __init__(self, amount):
        self.amount = amount
    def about(self):
        print(f"Its insured for an amount of {self.amount}")

class Cat(Pet, Insurable):
    def __init__(self, weight, breed, amount):
        self.weight = weight
        Pet.__init__(self, breed)
        Insurable.__init__(self, amount)
    def get_weight(self):
        print(f"{self.breed} Cat weighs around {self.weight} pounds")

def main():
    cat_obj = Cat(15, "Ragdoll", "$100")
    cat_obj.about()
    cat_obj.get_weight()
if __name__ == "__main__":
    main()
```

15. Write Pythonic code to overload “+”, “-” and “*” operators by providing the methods __add__, __sub__ and __mul__.

16.Write Pythonic code to create a function named move_rectangle() that takes an object of Rectangle class and two numbers named dx and dy. It should change the location of the Rectangle by adding dx to the x coordinate of corner and adding dy to the y coordinate of corner

Copying an object is often an alternative to aliasing. The `copy` module contains a function called `copy` that can duplicate any object:

```
>>> p1 = Point()
>>> p1.x = 3.0
>>> p1.y = 4.0
>>> import copy
>>> p2 = copy.copy(p1)
```

`p1` and `p2` contain the same data, but they are not the same Point.

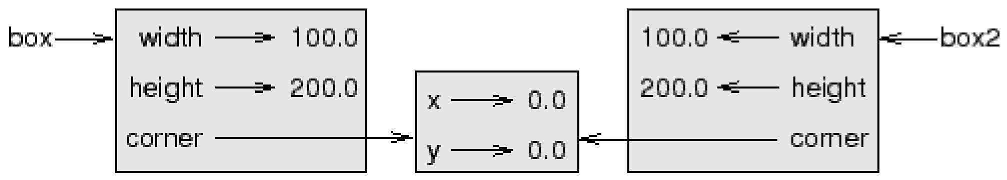
```
>>> print_point(p1)
(3.0, 4.0)
>>> print_point(p2)
(3.0, 4.0)
>>> p1 is p2
False
>>> p1 == p2
False
```

The `is` operator indicates that `p1` and `p2` are not the same object, which is what we expected. But you might have expected `==` to yield `True` because these points contain the same data. In that case, you will be disappointed to learn that for instances, the default behavior of the `==` operator is the same as the `is` operator; it checks object identity, not object equivalence. This behavior can be changed—we'll see how later.

If you use `copy.copy` to duplicate a Rectangle, you will find that it copies the Rectangle object but not the embedded Point.

```
>>> box2 = copy.copy(box)
>>> box2 is box
False
>>> box2.corner is box.corner
True
```

Here is what the object diagram looks like:



This operation is called a **shallow copy** because it copies the object and any references it contains, but not the embedded objects.

For most applications, this is not what you want. In this example, invoking `grow_rectangle` on one of the Rectangles would not affect the other, but invoking `move_rectangle` on either would affect both! This behavior is confusing and error-prone.

Fortunately, the `copy` module contains a method named `deepcopy` that copies not only the object but also the objects it refers to, and the objects *they* refer to, and so on. You will not be surprised to learn that this operation is called a **deep copy**.

```
>>> box3 = copy.deepcopy(box)
>>> box3 is box
False
>>> box3.corner is box.corner
False
```

`box3` and `box` are completely separate objects.

17.What is Exception? Explain Exception as control flow mechanism with suitable example.

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error.

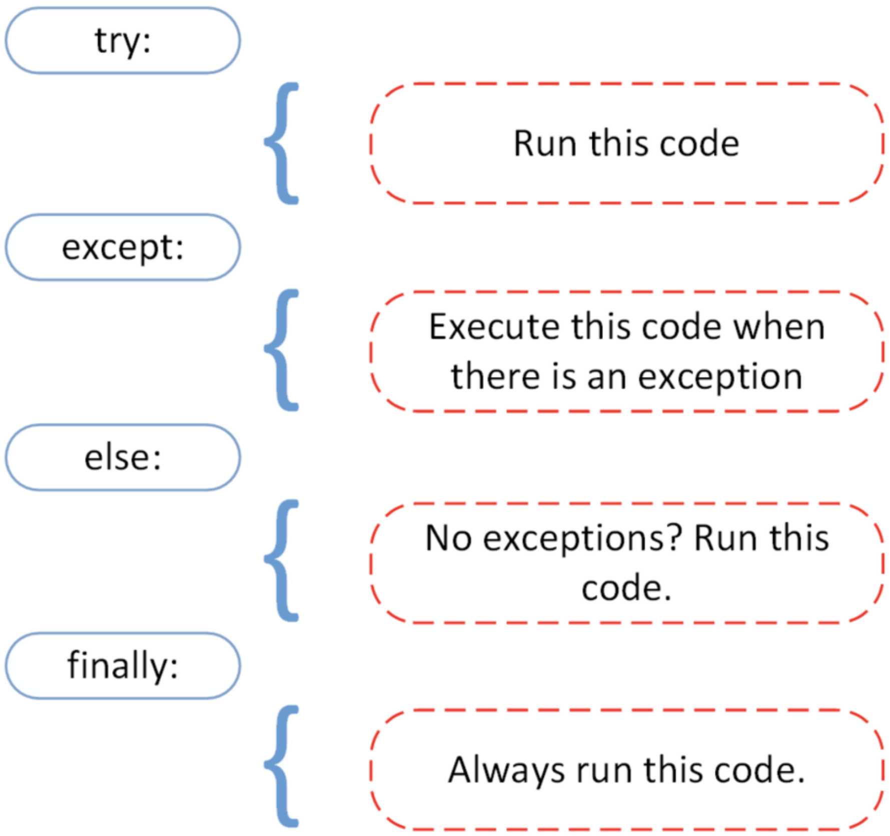
When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

Ex-

```
try:
   fh = open("testfile", "w")
   fh.write("This is my test file for exception handling!!")
except IOError:
   print "Error: can\'t find file or read data"
else:
   print "Written content in the file successfully"
   fh.close()
```

o/p-

```
Written content in the file successfully
```


try:
Run this code
except:
Execute this code when there is an exception
else:
No exceptions? Run this code.
finally:
Always run this code.