PART-1

Introduction : The Programming Cycle for Python, Python IDE, Interacting with Python Programs.

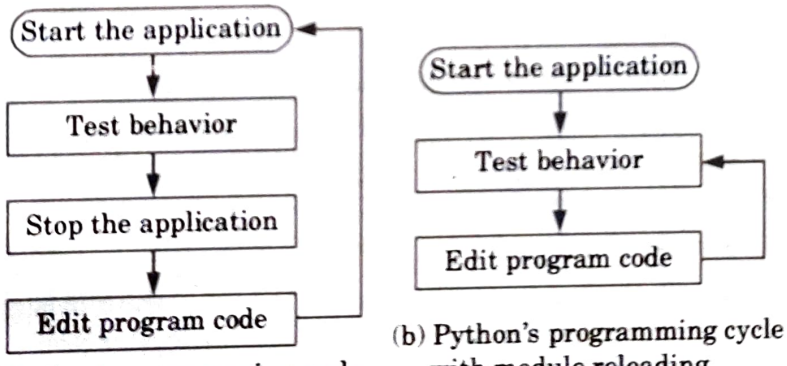
Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.1. Explain the programming cycle for Python.

Answer

1. Python's programming cycle is dramatically shorter than that of traditional programming cycle.
2. In Python, there are no compile or link steps.
3. Python programs simply import modules at runtime and use the objects they contain. Because of this, Python programs run immediately after changes are made.
4. In cases where dynamic module reloading can be used, it is even possible to change and reload parts of a running program without stopping it at all.
5. Fig. 1.1.1 shows Python's impact on the programming cycle.



(a) Python's programming cycle

(b) Python's programming cycle with module reloading

Fig. 1.1.1.

6. Since Python is interpreted, there is a rapid turnaround after program changes. And because Python's parser is embedded in Python-based systems, it is easy to modify programs at runtime.

Que 1.2. What is IDE ? Discuss some Python IDE.

Answer

1. IDE is a software package that consists of several tools for developing and testing the software.
2. An IDE helps the developer by automating the process.
3. IDEs integrate many tools that are designed for SDLC.
4. IDEs were introduced to diminish the coding and typing errors.
5. Some of the Python IDEs are :

a. PyCharm : PyCharm assists the developers to be more productive and provides smart suggestions. It saves time by taking care of routine tasks, hence increases productivity.

Features of PyCharm :

i. It has smart code navigation, good code editor, a function for quick refactoring.

ii. The integrated activities with PyCharm are profiling, testing, debugging, remote development, and deployments.

iii. PyCharm supports Python web development frameworks, Angular JS, JavaScript, CSS, HTML and live editing functions.

b. Spyder : Spyder is widely used for data science works. It is mostly used to create a secure and scientific environment for Python. Spyder Python uses PyQt (Python plug-in) which a developer can add as an extension.

Features of Spyder :

i. It has good syntax highlighting and auto code completion features.

ii. Spyder Python explores and edits variables directly from GUI.

iii. It performs very well in multi-language editor.

c. PyDev : It is an external plug-in for Eclipse and is very popular as Python interpreter.

Features of PyDev :

i. PyDev has strong parameters like refactoring, debugging, type hinting, code analysis, and code coverage function.

ii. PyDev supports tokens browser, PyLint integration, interactive console, remote debugger, Unittest integration, etc.

d. IDLE : IDLE is a basic IDE mainly used by beginner level developer.

i. IDLE Python is a cross-platform IDE, hence it increases the flexibility for users.

ii. It is developed only in Python in collaboration with Tkinter GUI toolkit.

iii. The feature of multi-window text editor in IDLE has some great functions like smart indentation, call tips, Python colorizing, and undo option.

iv. It also comes with a strong debugger along with continuous breakpoints, local spaces, and global view.

v. It supports browsers, editable configurations, and dialog boxes.

e. **Visual studio :** It enables development for various platforms and has its own marketplace for extensions.

Features of visual studio :

i. It supports Python coding in visual studio, debugging, and other activities.

ii. It has both paid and free versions in the market with great features.

Que 1.3. Discuss interaction with Python program with example.

Answer

1. The Python program that we have installed will by default act as an interpreter.
2. An interpreter takes text commands and runs them as we enter text.
3. After Python opens, it will show some contextual information like this :

   Python 3.5.0 (default, dec 20 2019, 11 : 28 : 25)

   [GCC 5.2.0] on Linux

   Type "help", "copyright", "credits" or "license" for more information

   >>>
4. The prompt >>> defines that we are now in an interactive Python interpreter session, also called the Python shell.
5. Now enter some code for Python to run such as print("Hello World!") and press the Enter key.
6. The interpreter's response should appear on the next line like this :

   >>> print ("Hello World!")

   Hello World!
7. After showing the results, Python will bring you back to the interactive prompt, where we could enter another command or code.
8. Python program communicates its result to user using print statement.

Que 1.4. What is Python ? How Python is interpreted ? What are the tools that help to find bugs or perform static analysis ? What are Python decorators ? **AKTU 2019-20, Marks 10**

Answer

Python : Python is a high-level, interpreted, interactive and object-oriented scripting language. It is a highly readable language. Unlike other programming languages, Python provides an interactive mode similar to that of a calculator.

Interpretation of Python :

1. An interpreter is a kind of program that executes other programs.
2. When we write Python programs, it converts source code written by the developer into intermediate language which is again translated into the machine language that is executed.
3. The python code we write is compiled into python bytecode, which creates file with extension .pyc .
4. The bytecode compilation happened internally and almost completely hidden from developer.
5. Compilation is simply a translation step, and byte code is a lower-level, and platform-independent, representation of source code.
6. Each of the source statements is translated into a group of bytecode instructions. This bytecode translation is performed to speed execution. Bytecode can be run much quicker than the original source code statements.
7. The .pyc file, created in compilation step, is then executed by appropriate virtual machines.
8. The Virtual Machine iterates through bytecode instructions, one by one, to carry out their operations.
9. 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 actually runs the Python scripts.

10 It is the last step of Python interpreter.

Following tools are the static analysis tools that help to find bugs in python :

1. **Pychecker :** Pychecker is an open source tool for static analysis that detects the bugs from source code and warns about the style and complexity of the bug.
2. **Pylint :**
   a. Pylint is highly configurable and it acts like special programs to control warnings and errors, it is an extensive configuration file
   b. It is an open source tool for static code analysis and it looks for programming errors and is used for coding standard.
   c. It also integrates with Python IDEs such as Pycharm, Spyder, Eclipse, and Jupyter.

Python decorators :

1. Decorators are very powerful and useful tool in Python since it allows programmers to modify the behavior of function or class.

2. Decorators allow us to wrap another function in order to extend the behavior of wrapped function, without permanently modifying it.
3. In decorators, functions are taken as the argument into another function and then called inside the wrapper function.
4. **Syntax :**
   ```
   @gfg_decorator
   def hello_decorator():
       print("Gfg")
   ```
5. gfg_decorator is a callable function, will add some code on the top of some another callable function, hello_decorator function and return the wrapper function.

PART-2

Elements of Python, Type Conversion.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.5. What do you mean by comments in Python ?

Answer

Comments :

1. Python allows us to add comments in the code.
2. Comments are used by the programmer to explain the piece of code to be understood by other programmer in a simple language. Every programming language makes use of some character for commenting.
3. Python uses the hash character (#) for comments. Putting # before a text ensures that the text will not be parsed by the interpreter.
4. Comments do not affect the programming part and the Python interpreter does not display any error message for comments.

   For example : Commenting using hash mark (#)

   ```
   >>> 8 + 9                    # addition
   17                           # Output
   >>>
   ```

   In this example, 'addition' is written with a hash mark. Hence the interpreter understands it as a comment and does not display any more messages.

Que 1.6. Explain identifiers and keywords with example.

Answer

1. A Python identifier is the name given to a variable, function, class, module or other object.

2. An identifier can begin with an alphabet (A – Z or a – z), or an underscore (_) and can include any number of letters, digits, or underscores and spaces are not allowed.
3. Python will not accept @, $ and % as identifiers.
4. Python is a case-sensitive language. Thus, Hello and hello both are different identifiers. In python, a class name will always start with a capital letter.

Table 1.6.1. Examples of Identifiers

Valid	Invalid
MyName	My Name (Space is not allowed)
My_Name	3dfig (cannot start with a digit)
Your_Name	Your#Name (Only alphabetic character, Underscore (_) and numeric are allowed)

Reserved keywords : Python has a list of reserved words known as keywords. Every keyword has a specific purpose and use.

Some of the reserved keywords in Python :

and	del	from	None	True
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	false	in	pass	yield
continue	finally	is	raise	
def	for	lambda	return	

Que 1.7. **Define variable. Also discuss variable initialization.**

Answer

Variables :

1. A variable holds a value that may change.
2. The process of writing the variable name is called declaring the variable.
3. In Python, variables do not need to be declared explicitly in order to reserve memory spaces as in other programming languages like C, Java, etc.
4. When we initialize the variable in Python, Python Interpreter automatically does the declaration process.

Initializing a variable :

1. The general format of assignment statement is as follows :

Variable = Expression

2. The equal sign (=) is known as assignment operator.
3. An expression is any value, text or arithmetic expression where as variable is the name of variable.
4. The value of the expression will be stored in the variable.

Example of initializing a variable :

```
>>>year=2016
>>> name='Albert'
```

The two given statements reserve two memory spaces with variable names year and name. 2016 and Albert, are stored in these memory spaces.

Que 1.8. What do you mean by data types ? Explain numeric and string data type with example.

Answer

Data types :

i. The data stored in the memory can be of many types. For example, a person's name is stored as an alphabetic value and his address is stored as an alphanumeric value.

ii. Python has six basic data types which are as follows :

1. Numeric
2. String
3. List
4. Tuple
5. Dictionary
6. Boolean

Numeric :

1. Numeric data can be broadly divided into integers and real numbers (*i.e.*, fractional numbers). Integers can be positive or negative.
2. The real numbers or fractional numbers are called, floating point numbers in programming languages. Such floating point numbers contain a decimal and a fractional part.

For example :

```
>>> num1 = 2   # integer number
>>>num2 = 2.5   # real number (float)
>>>num1
2              # Output
>>>num2
2.5            # Output
>>>
```

String :

1. Single quotes or double quotes are used to represent strings.
2. A string in Python can be a series or a sequence of alphabets, numerals and special characters.

For example :

```
>>> sample_string = "Hello"   # store string value
>>> sample_string   # display string value
'Hello'             # Output
```

Que 1.9. **Discuss list and tuple data types in detail.**

Answer

List :

1. A list can contain the same type of items.
2. Alternatively, a list can also contain different types of items.
3. A list is an ordered and indexable sequence.
4. To declare a list in Python, we need to separate the items using commas and enclose them within square brackets ([ ]).
5. Operations such as concatenation, repetition and sub-list are done on list using plus (+), asterisk (*) and slicing (:) operator.

For example :

```
>>>first = [1, "two", 3.0, "four" ]   # 1st list
>>>second = ["five", 6]   # 2nd list
>>>first   # display 1st list
[1, 'two', 3.0, 'four']   # Output
```

Tuple :

1. A tuple is also used to store sequence of items.
2. Like a list, a tuple consists of items separated by commas.
3. Tuples are enclosed within parentheses rather than within square brackets.

For example :

```
>>>third = (7, "eight", 9, 10.0)
>>>third
(7, 'eight', 9, 10.0)   # Output
```

Que 1.10. **Explain dictionary and Boolean data type.**

Answer

Dictionary :

1. A Python dictionary is an unordered collection of key-value pairs.

2. When we have the large amount of data, the dictionary data type is used.
3. Keys and values can be of any type in a dictionary.
4. Items in dictionary are enclosed in the curly-braces {} and separated by the comma (,).
5. A colon (:) is used to separate key from value. A key inside the square bracket [ ] is used for accessing the dictionary items.

For example :

```
>>> dict1 = {1:"first line", "second" : 2}   # declare dictionary
>>> dict1[3] = "third line"   # add new item
>>> dict1 # display dictionary
    {1 : 'first line', 'second' : 2, 3: 'third line'}   # Output
```

Boolean :

1. In a programming language, mostly data is stored in the form of alphanumeric but sometimes we need to store the data in the form of 'Yes' or 'No'.
2. In terms of programming language, Yes is similar to True and No is similar to False.
3. This True and False data is known as Boolean data and the data types which stores this Boolean data are known as Boolean data types.

For example :

```
>>> a = True
>>> type (a)
<type 'bool'>
```

Que 1.11. What do you mean by type conversion ?

Answer

1. The process of converting one data type into another data type is known as type conversion.
2. There are mainly two types of type conversion methods in Python :
   - **a. Implicit type conversion :**
     - i. When the data type conversion takes place during compilation or during the run time, then it called an implicit data type conversion.
     - ii. Python handles the implicit data type conversion, so we do not have to explicitly convert the data type into another data type.

For example :

```
a = 5
b = 5.5
sum = a + b
print (sum)
print (type (sum)) # type() is used to display the datatype of a variable
```

Output :

```
10.5
<class 'float'>
```

iii. In the given example, we have taken two variables of integer and float data types and added them.

iv. Further, we have declared another variable named 'sum' and stored the result of the addition in it.

v. When we checked the data type of the sum variable, we can see that the data type of the sum variable has been automatically converted into the float data type by the Python compiler. This is called implicit type conversion.

b. Explicit type conversion:

i Explicit type conversion is also known as type casting.

ii. Explicit type conversion takes place when the programmer clearly and explicitly defines the variables in the program.

For example :

```
# adding string and integer data types using explicit type conversion
a = 100
b = "200"
result1 = a + b
b = int(b)
result2 = a + b
print (result2)
```

Output :

```
Traceback (most recent call last):
File "", line 1, in
TypeError : unsupported operand type (s) for +: 'int' and 'str'
300
```

iii. In the given example, the variable *a* is of the number data type and variable *b* is of the string data type.

iv. When we try to add these two integers and store the value in a variable named result1, a TypeError occurs. So, in order to perform this operation, we have to use explicit type casting.

v. We have converted the variable b into integer type and then added variable a and b. The sum is stored in the variable named result2, and when printed it displays 300 as output.

PART-3

Basics : Expressions, Assignment Statement.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.12. What is expression ?

Answer

1. An expression is a combination of symbols that evaluates to a value.
2. An expression is a combination of variables, operators, values sub-expressions and a reserve keyword.
3. Whenever we type an expression in the command line, the interpreter evaluates it and produces the result.
4. Expressions that evaluate to a numeric type are called arithmetic expressions.
5. A sub-expression is any expression that is part of a larger expression. Sub-expressions are denoted by the use of parentheses.

   For example : $4 + (3 * k)$

   An expression can also consist of a single literal or variable. Thus, 4, 3, and k are each expression.

   This expression has two sub-expressions, 4 and $(3 * k)$. Sub-expression $(3 * k)$ itself has two sub-expressions, 3 and k.

Que 1.13. What do you mean by assignment statement?

Answer

1. Assignment statements are used to create new variables, assign values, and change values.
2. **Syntax of assignment statement :**

   # LHS <=> RHS

   Variable = Expression

3. There are three types of assignment statements :
   i. Value-based expression on RHS
   ii. Current variable on RHS
   iii. Operation on RHS

Que 1.14. Discuss types of assignment statements with examples.

Answer

Types of assignment statement :

1. **Value-based expression on RHS :** In this type, Python allocates a new memory location to the newly assigned variable.
   For example :
   test1 = "Hello"
   id(test1)
   Output :
   2524751071304

2. **Current variable on RHS :** In this type, Python does not allocate a new memory location.
   For example :
   current_var = "It's Quantum Series"
   print(id(current_var))
   new_var = current_var
   print(id(new_var))
   Output :
   24751106240
   2524751106240

3. **Operation on RHS :** In this type, we have an operation on the RHS of the statement, which is the defining factor of the type of our statement.
   For example :
   test1 = 7 * 2 / 10
   type(test1)
   Output :
   float

PART-4

Arithmetic Operators.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.15. Define the term operator.

Answer

1. An operator is a symbol that represents an operation that may be performed on one or more operands.
2. Operators are constructs used to modify the values of operands.
3. Operators that take one operand are called unary operators.
4. Operators that take two operands are called binary operators.
5. Based on functionality operators are categories into following seven types :
   i. Arithmetic operators.
   ii. Assignment operators.
   iii. Bitwise operators.
   iv. Comparison operators.
   v. Identity operators.
   vi. Logical operators.
   vii. Membership operators.

Que 1.16. Discuss arithmetic and comparison operator with example.

Answer

Arithmetic operators : These operators are used to perform arithmetic operation such as addition, subtraction, multiplication and division.

Table 1.16.1. List of arithmetic operators.

Operator	Description	Example
+	Addition operator to add two operands.	10 + 20 = 30
–	Subtraction operator to subtract two operands.	10 – 20 = –10
×	Multiplication operator to multiply two operands.	10 × 20 = 200
/	Division operator to divide left hand by right hand operator.	5 / 2 = 2.5
**	Exponential operator to calculate power.	5 ** 2 = 25
%	Modulus operator to find remainder.	5 % 2 = 1
//	Floor division operator to find the quotient and remove the fractional part.	5 // 2 = 2

Comparison operators : These operators are used to compare values. It is also called relational operators. The result of these operator is always a Boolean value *i.e.*, true or false.

Table 1.16.2. List of comparison operators.

Operator	Description	Example
= =	Operator to check whether two operand are equal.	10 = = 20, false
! = or <>	Operator to check whether two operand are not equal.	10 ! = 20, true
>	Operator to check whether first operand is greater than second operand.	10 > 20, false
<	Operator to check whether first operand is smaller than second operand.	10 < 20, true
> =	Operator to check whether first operand is greater than or equal to second operand.	10 > = 20, false
< =	Operator to check whether first operand is smaller than or equal to second operand.	10 < = 20, true

Que 1.17. Explain assignment operator with example.

Answer

Assignment operators : This operator is used to store right side operands in the left side operand.

Table 1.17.1. List of assignment operators.

Operator	Description	Example
=	Store right side operand in left side operand.	$a = b + c$
+ =	Add right side operand to left side operand and store the result in left side operand	$a + = b$ or $a = a + b$
− =	Subtract right side operand to left side operand and store the result in left side operand	$a - = b$ or $a = a - b$
* =	Multiply right side operand with left side operand and store the result in left side operand	$a * = b$ or $a = a * b$
/ =	Divide left side operand by right side operand and store the result in left side operand	$a / = b$ or $a = a / b$
% =	Find the modulus and store the remainder in left side operand	$a \% = b$ or $a = a \% b$
* * =	Find the exponential and store the result in left side operand	$a * * = b$ or $a = a * * b$
/ / =	Find the floor division and store the result in left side operand	$a / / = b$ or $a = a / / b$

Que 1.18. **Define bitwise operator with example.**

Answer

Bitwise operators : These operators perform bit level operation on operands. Let us take two operand $x = 10$ and $y = 12$. In binary format this can be written as $x = 1010$ and $y = 1100$.

Table 1.18.1. List of bitwise operators.

Operator	Description	Example
& Bitwise AND	This operator performs AND operation between operands. Operator copies bit if it exists in both operands	x & y results 1000
| Bitwise OR	This operator performs OR operation between operands. Operator copies bit if it exists in either operand	x | y results 1 110
^ Bitwise XOR	This operator performs XOR operation between operands. Operator copies bit if it exists only in one operand.	x ^ y results 0110
~ bitwise inverse	This operator is a unary operator used to inverse the bits of operand.	~ x results 0101
<< left shift	This operator is used to shift the bits towards left	x << 2 results 101000
<<right shift	This operator is used to shift the bits towards right	x >> 2 results 0010

Que 1.19. **Discuss logical and identity operator with example.**

Answer

Logical operators : These operators are used to check two or more conditions. The resultant of this operator is always a Boolean value. Here x and y are two operands that store either true of false Boolean values.

Operator	Description	Example
and logical AND	This operator perform AND operation between operands. When both operands are true, the resultant become true.	x and y results false
or logical OR	This operator perform OR operation between operands. When any operand is true, the resultant become true.	x and y results true
not logical NOT	This operator is used to reverse the operand state.	not x result false

Identity operators : These operator are used to check whether both operands are same or not. Suppose x stores a value 20 and y stores a value 40. Then x is y returns false and x not is y return true.

Table 1.19.2. List of identity operators.

Operator	Description	Example
is	Return true, if the operands are same. Return false, if the operands are not same.	x is y, results false
not is	Return false, if the operands are same. Return true, if the operands are not same.	x not is y, results true

Que 1.20. **Explain membership operator with example.**

Answer

Membership operators :

1. These operators are used to check an item or an element that is part of a string, a list or a tuple.
2. A membership operator reduces the effort of searching an element in the list.
3. Suppose x stores a value 20 and y is the list containing items 10, 20, 30, and 40. Then x is a part of the list y because the value 20 is in the list y.

Table 1.20.1. List of Membership operators.

Operator	Description	Example
in	Return true, if item is in list or in sequence. Return false, if item is not in list or in sequence.	x in y, results true
not in	Return false, if item is in list or in sequence. Return true, if item is not in list or in sequence.	x not in y, results true

PART-5

Operator Precedence, Boolean Expression.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.21. What do you mean by operator precedence ?

Answer

1. When an expression has two or more operator, we need to identify the correct sequence to evaluate these operators. This is because result of the expression changes depending on the precedence.

   For example : Consider a mathematical expression :

   10 + 5 / 5

   When the given expression is evaluated left to right, the final answer becomes 3.

2. However, if the expression is evaluated right to left, the final answer becomes 11. This shows that changing the sequence in which the operators are evaluated in the given expression also changes the solution.

3. Precedence is the condition that specifies the importance of each operator relative to the other.

Table 1.21.1. Operator precedence from lower precedence to higher.

Operator	Description
NOT, OR AND	Logical operators
in , not in	Membership operator
is, not is	Identity operator
=, %=, /=, //=, –=, +=, *=, **==	Assignment operators.
<>, ==, !=	Equality comparison operator
<=, <, >, >=	Comparison operators
^, |	Bitwise XOR and OR operator
&	Bitwise AND operator
<<, >>	Bitwise left shift and right shift
+, –	Addition and subtraction
*, /, %, ??	Multiplication, Division, Modulus and floor division
**	Exponential operator

Que 1.22. **Explain operator associativity.**

Answer

Associativity :

1. Associativity decides the order in which the operators with same precedence are executed.
2. There are two types of associativity :
   a. **Left to right :** In left to right associativity, the operators of same precedence are executed from the left side first.
   b. **Right to left :** In right to left associativity, the operators of same precedence are executed from the right side first.
3. Most of the operators in Python have left to right associativity.
4. Left to right associative operators are multiplication, floor division, etc. and ** operator is right to left associative.
5. When two operators have the same precedence then operators are evaluated from left to right direction.

   For example :

```
>>> 3 * 4 / / 6
2  # Output
>>> 3 * (4 / / 6)
```

```
0  # Output
>>> 3 * * 4 * * 2  # 3 ^ 16
43046721  # Output
>>> (3 * * 4) * * 2  # 81 ^ 2
6561  # Output
```

Que 1.23. **What do you mean by Boolean expression ?**

OR

Write short notes with example : The programming cycle for Python, elements of Python, type conversion in Python, operator precedence, and Boolean expression. **AKTU 2019-20, Marks 10**

Answer

Programming cycle for Python : Refer Q. 1.1, Page 1–2T, Unit-1.
Elements of Python : Refer Q. 1.8, Page 1–8T, Refer Q. 1.9, Page 1–9T, and Refer Q. 1.10, Page 1–9T; Unit-1.
Type conversion in Python : Refer Q. 1.11, Page 1–10T, Unit-1.
Operator precedence : Refer Q. 1.21, Page 1–18T, Unit-1.
Boolean expression : A boolean expression may have only one of two values : True or False.
For example : In the given example comparison operator (==) is used which compares two operands and prints true if they are equal otherwise print false :

```
>>> 5 == 5
True  # Output
>>> 5 == 6
False  # Output
```

Que 1.24. **How memory is managed in Python? Explain PEP 8. Write a Python program to print even length words in a string.**

AKTU 2019-20, Marks 10

Answer

Memory management :

1. Memory management in Python involves a private heap containing all Python objects and data structures.
2. The management of this private heap is ensured internally by the Python memory manager.
3. The Python memory manager has different components which deal with various dynamic storage management aspects, like sharing, segmentation, preallocation or caching.
4. At the lowest level, a raw memory allocator ensures that there is enough room in the private heap for storing all Python-related data by interacting with the memory manager of the operating system.

5. On top of the raw memory allocator, several object-specific allocators operate on the same heap and implement distinct memory management policies adapted to the peculiarities of every object type.
6. For example, integer objects are managed differently within the heap than strings, tuples or dictionaries because integers imply different storage requirements and speed/space tradeoffs.
7. Python memory manager thus delegates some of the work to the object-specific allocators, but ensures that the latter operate within the bounds of the private heap.

PEP 8 :

1. A PEP is a design document providing information to the Python community, or describing a new feature for Python or its processes or environment.
2. The PEP should provide a concise technical specification of the feature.
3. PEP is actually an acronym that stands for Python Enhancement Proposal.
4. PEP 8 is Python's style guide. It is a set of rules for how to format the Python code to maximize its readability.
5. A PEP is a design document providing information to the Python community, or describing a new feature for Python or its processes or environment.

Program to print even length words in a string :

```
def printWords(s) :
# split the string
    s = s.split(' ')
    " iterate in words of string
    for word in s:
    # if length is even
    if len(word)%2==0:
        print(word)
# Driver Code
s = "i am muskan"
printWords(s)
```

Output :

```
am
muskan
```

Que 1.25. What will be the output after the following statements ?

```
x = 6
y = 3
print(x / y)
```

Answer

2.0

Que 1.26. **What will be the output after the following statements ?**

```
x = 8
y = 2
print(x // y)
```

Answer

4

Que 1.27. **What will be the output after the following statements ?**

```
x = 5
y = 4
print(x % y)
```

Answer

1

Que 1.28. **What will be the output after the following statements ?**

```
x = 3
y = 2
x += y
print(x)
```

Answer

5

Que 1.29. **What will be the output after the following statements ?**

```
x = 5
y = 7
x *= y
print(x)
```

Answer

35

Que 1.30. **What will be the output after the following statements ?**

```
x = 30
```

```
y = 7
x %= y
print(x)
```

Answer

2

Que 1.31. **What will be the output after the following statements ?**

```
x = 3
y = 7
print(x == y)
```

Answer

False

Que 1.32. **What will be the output after the following statements ?**

```
x = 8
y = 6
print(x != y)
```

Answer

True

Que 1.33. **What will be the output after the following statements ?**

```
x = 83
y = 57
print(x > y)
```

Answer

True

Que 1.34. **What will be the output after the following statements ?**

```
x = 72
y = 64
print(x < y)
```

Answer

False

Que 1.35. **What will be the output after the following statements ?**

```
x = True
```

```
y = False
print(x and y)
```

Answer

False

Que 1.36. **What will be the output after the following statements ?**

```
x = True
y = False
print(x or y)
```

Answer

True

Que 1.37. **What will be the output after the following statements ?**

```
x = True
y = False
print(not x)
```

Answer

False

Que 1.38. **What will be the output after the following statements ?**

```
x = True
y = False
print(not y)
```

Answer

True

Que 1.39. **What will be the output after the following statements ?**

```
x = 20
y = 40
z = y if (y > x) else x
print(z)
```

Answer

40

Que 1.40. **What will be the output after the following statements ?**

```
x = 50
```

```
y = 10
z = y if (y > x) else x
print(z)
```

Answer

50

Que 1.41. **What will be the output after the following statements ?**

```
x = 65
y = 53
z = y if (x % 2 == 0) else x
print(z)
```

Answer

65

Que 1.42. **What will be the output after the following statements ?**

```
x = 46
y = 98
z = y if (y % 2 == 0) else x
print(z)
```

Answer

98

Que 1.43. **What will be the output after the following statements ?**

```
x = 2 * 4 + 7
print(x)
```

Answer

15

Que 1.44. **What will be the output after the following statements ?**

```
x = 7 * (4 + 5)
print(x)
```

Answer

63

Que 1.45. **What will be the output after the following statements ?**

```
x = '24' + '16'
```

print(x)

Answer

2416

Que 1.46. **What will be the output after the following statements ?**

```
x = 15 + 35
print(x)
```

Answer

50

Que 1.47. **What will be the data type of *x* after the following statement if input entered is 18 ?**

```
x = input('Enter a number:')
```

Answer

String

Que 1.48. **What will be the data type of *y* after the following statements if input entered is 50 ?**

```
x = input('Enter a number:')
y = int(x)
```

Answer

Integer

Que 1.49. **What will be the data type of *y* after the following statements ?**

```
x = 71
y = float(x)
```

Answer

Float

Que 1.50. **What will be the data type of *y* after the following statements ?**

```
x = 48
y = str(x)
```

Answer

String

Que 1.51. **What will be the output after the following statements ?**

```
x = y = z = 8
print(y)
```

Answer

8

Que 1.52. **What will be the value of *x*, *y* and *z* after the following statement ?**

```
x = y = z = 300
```

Answer

All three will have the value of 300

Que 1.53. **What will be the value of *x*, *y* and *z* after the following statement ?**

```
x, y, z = 3, 4, 5
```

Answer

x will have the value of 3, *y* will have the value 4 and *z* will have the value of 5.

Que 1.54. **In the order of precedence, which of the operation will be completed last in the following statement ?**

```
3 * 6 + 5 – 4 / 2
```

Answer

Subtraction

Que 1.55. **What will be the order of precedence of operations in the following statement ?**

```
10 * 4 – 1 + 8 / 5
```

Answer

Multiplication, Division, Addition, Subtraction

Que 1.56. **What will be the data type of *x* after the following statement if input entered is 64 ?**

```
x = float(input('Enter a number:'))
```

Answer

Float

Que 1.57. **What will be the output after the following statements ?**

```
a = 27 / 3 % 2 * 4**2
print(a)
```

Answer

16.0

Que 1.58. **What will be the output after the following statements ?**

```
a = 3 / 3 + 4 * 7 - 3 * 3
print(a)
```

Answer

20.0