BACHELOR OF COMPUTER APPLICATIONS

SEMESTER *5*

DCA3104

PYTHON PROGRAMMING

Unit 4

Operators

Table of Contents

1. INTRODUCTION

Python program needs these basic elements so that one can define the objects that are needed to create a program. However, a program cannot only be created with the help of variables. These variables need to perform some sort of function so that their data can be manipulated. Only through manipulation, comparison, and identification of variables can we proceed with Python and code programs that perform a certain function.

All programming languages are essentially computer languages. And the basic function of a computer language is to compute data. The data entered should be available for performing basic operations such as addition, subtraction, division, multiplication, and other elementary mathematical operations.

To perform these operations, Python uses various symbols. These symbols hold specifmeanings and each has a special function. The symbols indicate to the interpreter what kind of operation is to be performed on the variables where the symbols are used. These symbols are known as operators. Operators are used to performing specific mathematical or logical manipulations.

Operators take one or more values or expressions and provide another value. The value provided is obtained after operating the previous values as indicated by the operator used. You can group various operators together in a single expression and the interpretation will apply them one after another, in a specific order. Various operators can take a different number of values. Some operators used in Python will need only one value to perform their function. Some may need two or more values to work. One should take care that they define the required number of variables while using an operator so that they do not get an error in their program.

In this unit, we will learn the various operators that are applicable in Python and how some may differ from the ones that you know from mathematics. With the help of these operators, you will be able to create basic programs that yield the desired value.

1.1 Learning Objectives

After studying the chapter, you will be able to:

- *Describe various arithmetic operators used in Python and their uses.*
- *Understand the concept and working of comparison, assignment, and bitwise operators.*
- *Detail the working of logical operators and the values they can yield.*
- *Describe how membership and identity operators work.*
- *Understand the concept of operator precedence in a Python program*

2. ARITHMETIC OPERATORS

Python allows the use of various arithmetic operators that are used to perform mathematical operations such as addition, subtraction, division, multiplication, etc. These operators are essentially used to perform calculations using the Python programming language. The operators can be used on numeric data types such as integers, long integers, floating-point real numbers, complex numbers, and others that can be defined in Python. All these arithmetic operators are explained in detail below.

2.1. Addition

The addition operator, signified by the symbol '+', is used to add two operands. The syntax through which it works is shown below. It needs at least two values or operands at its either side for it to work.

Syntax:

```
x + y
```

Example:

```
a = 5
b = 8
c = a + b
print (c )

#The output of the program will be 13.
```

STUDY NOTE

The + operator can also be used to add two strings.

2.2. Subtraction

The subtraction operator, represented by '-' also works similar to the addition operator. It finds the difference between the two operands that are written at its either side.

Syntax:

```
x - y
```

Example:

```
a = 5
b = 8
c = a - b
print (c )

#The output of the program will be -3.
```

2.3. Division

The division operator, also known as float division operator to differentiate it from floor division, is used to divide the first operand from the second. In Python, it is not necessary to define two floating-point real numbers to get the answer in decimal. You can get the answer is float even if the integers divided are not perfectly divisible. Thus, in Python, the decimal is not truncated even if the values used are defined as integers.

Syntax:

```
x / y
```

Example:

```
a = 5
b = 20
c = b / a
print (c )
#The output of the program will be 4.
```

2.4. Multiplication

The multiplication operator multiples the two operands that are placed on either side. You can multiply two different data types using this operator.

Syntax:

```
x * y
```

Example:

```
a = 5
b = 8
c = a * b
print (c )
#The output of the program will be 40.
```

2.5. Remainder

The modulus operator, represented by '%' is used to find the remainder of a division. It divides the first operand by the second and returns the value of the remainder obtained after the division.

Syntax:

```
x % y
```

Example:

```
a = 40
b = 9
c = a % b
print (c )
#The output of the program will be 4.
```

2.6. Exponent

The exponent operator, also known as the power operator, is used to yield the value when a number is raised to a power. The second operand defined here will be the power.

STUDY NOTE

The operator for floor division was introduced in version Python 2.2 and onwards

Syntax:

```
x ** y
```

This indicates that the interpreter will compute the value of xy.

Example:

```
a = 5
b = 2
c = a ** b
print (c )
#The output of the program will be 25.
```

2.7. Floor Division

There are times when you need a division to be in the form of an integer. Python has the feature of providing a floating result even if the operands used in the division are integers. However, for some operations, an integer result is needed. Thus, floor division can be used in such cases.

Syntax:

```
x // y
```

Example:

```
a = 7
b = 54
c = b // a
print (c )
#The output of the program will be 7.
```

SELF-ASSESSMENT QUESTIONS - 1

1. What is the correct operator to write xy in Python?
   A. x^y
   B. x**y
   C. x+86
   D. x^^y
2. _____ operator is used for floor division.
3. Floor division gives _____ as result.
4. _____ is the sign used for reminder operator in Python.
5. Mathematical operations can be performed on strings. (True or False)

3. COMPARISON OPERATORS

There are situations when you need to compare two quantities and take a decision as per the result. For example, when you need to find whether a person is eligible to cast a vote or not, you will need to compare their ages to the standard, that is, 18 years. If their age is more than 18, then they can cast a vote. However, if the opposite is true, then the person will not be allowed. Such decisions need to be made in various scenarios and situations.

To get a result in these cases, you can use comparison operators. Comparison operators, or, also known as relational operators, are used to compare two values. The yield a value in the terms of True or False. That means, while comparing two properties, the relation between them can either be true or false. The following operators are used in Python to establish a relation between given operands.

Greater Than Operator

The operator '>' will give true if the first operand is greater than the second operand and false if otherwise.

STUDY NOTE

Operands are the values or data on which an operator is operated.

Syntax:

```
x > y
```

Example:

```
a = 5
b = 8
print (a > b)
#The output of the program will be False.
```

Lesser Than Operator

The operator used to show that the first operator is lesser than the second operator.

Syntax:

```
x < y
```

Example:

```
a = 5
b = 8
print (a < b)
#The output of the program will be True.
```

Equal to

This operator with the symbol "==" is used to compare two operands and find that they are both equal. Note that the equal to "==" operator is different from the assigning operator "=".

Syntax:

```
x == y
```

Example:

```
a = 5
b = 8
c = a + b
print (c == a)
#The output of the program will be False.
```

Not Equal to

The operator not equal to, represented by "!=" is used to find whether two operands are equal or not.

Syntax:

```
x != y
```

Example:

```
a = 5
b = 8
c = a + b
print (c != a)
```

```
#The output of the program will be True.
```

Greater than or equal to:

The operator yields the result true when the first operand is either greater than or equal to the second operand.

Syntax:

```
x >= y
```

Example:

```
a = 5
b = 8
print (a >= b)
#The output of the program will be False.
```

Lesser than or equal to

The operator has the symbol "<=" and is used to yield a value by comparing two values and finding whether the first operand is less than or equal to the second operand.

Syntax:

```
x <= y
```

Example:

```
a = 5
b = 8
print (a <= b)
#The output of the program will be True.
```

SELF-ASSESSMENT QUESTIONS - 2

6. Which operator means less than or equal to?
   A. >=
   B. >
   C. <
   D. <=

7. The output of the following program will be
   x = 5
   print (x>5)
   A. True
   B. False
   C. 5
   D. None of the above

8. The output of the following program will be
   x = 9
   print (x > 3 and x < 10)
   A. True
   B. False
   C. 9
   D. None of the above

9. The operator for "not equal to" in Python is _____.

10. Relational operators can yield values in the form of True or False. (True or False)

4. ASSIGNMENT OPERATORS

Assignment operators are used to assigning the desired value to a variable. The most commonly used assignment operator is one that we have been using in most of the programs till now, that is "=". The operator directly stores the value in the data address assigned. In addition to this, there are various other assignment operators in Python.

These are shorthand operators that are used to shorten an expression. These are explained in the table below.

TABLE 1: ASSIGNMENT OPERATORS

Statement	Assignment Operator	Example
a = a+1	+=	a+=1
a = a-1	-=	a-=1
a = a* n	*=	a*=n
a = a/n	/=	a/=n
a = a % b	%=	a %= b
a = a //n	//=	a //= n
a = a ** n	**=	a **= n
a = a & n	&=	a & = n
a = a | n	|=	a |= n

Activity I

Draw a flowchart and algorithm of a program that works performs various arithmetic operators that are applicable in Python.

SELF-ASSESSMENT QUESTIONS - 3

11. The shorthand operator used for the expression a = a + 1 is _____.

12. _____ is the assignment operator most commonly used in Python or any programming language.

13. The operator used to find the product of a number with the number itself or its multiples is _____.

14. _____ operator performs AND on the operands and assigns the value to the first operand written on the left.

15. Find the output of the following program.

```
a = 97
b = 7
a %= b
print (a)
```

A. 9
B. 13
C. 6
D. None of the above.

5. BITWISE OPERATORS

In Python, bitwise operators are used to performing calculations and manipulations bitwise. The integers are converted into bits or in binary form and the required functions are performed bit by bit, from left to right. These operators only work on integers and cannot be performed on floats and another form of data types.

The table below details the description of each bitwise operator used in Python along with its syntax.

TABLE 2: BITWISE OPERATORS

Name of the Operator	Symbol	Description
Bitwise AND operator	&	Returns 1 if both of the bit is 1
Bitwise OR operator	|	Returns 1 is either of the bit is 1
Bitwise NOT operator	~	Returns 1 if the bit is 0, that is, returns the compliment of the bit
Bitwise XOR operator	^	Returns 1 if one of the bits is 1 and the other is 0
Bitwise right shift	>>	Shifts the bits of the number to the right and enters 0 in the space left empty. It gives the result similar to dividing a number with some power of two.
Bitwise left shift	<<	Shifts the bits of the number to the left and enters 0 in the space left empty. It gives the result as if multiplying the number with some power of two.

SELF-ASSESSMENT QUESTIONS - 4

16. _____ is the symbol of the bitwise left shift operator.
17. _____is the symbol for the bitwise XOR operator.
18. In Bitwise operator, the integer if first converted in _____ form.
19. _____ provides the result similar to multiplying the number with a power of two.
20. The operator that provides the compliment of the number provided is _____.

6. LOGICAL OPERATORS

Logical operators work similarly to Boolean operators. In Python, there are three logical operators, and, or, and not. Logical AND, represented by "and" yields a true value if both the operands are true. Logical OR, written as "or" is true when one of the operands is true.

The logical NOT, or, as written in Python "not" give false if the operand is true and vice versa. The operands written on either side can be relational expressions. Such an arrangement of expressions is called logical expressions. A logical expression can yield only True or False values.

Syntax:

```
a and b
a or b
not a
```

Example:

```
a = True
b = False
print (a and b)
print (a or b)
print (not a)

#The output of the example code given above will look like this:
False
True
False
```

STUDY NOTE

Truth tables are created to signify the result of a Boolean expression.

STUDY NOTE

is operator will be false for two lists with same values because they are mutated.

SELF-ASSESSMENT QUESTIONS - 5

21. The output of the following program will be

```
a = True
b = False
print (not a)
```

A. True
B. False
C. a
D. b

22. The output of the code given below will be

```
a = 5
b = 7
x = a + b
y = a – b
c = x > y
d = x < y
print (c and d)
```

A. True
B. False
C. C.14
D. None of the above

23. _____ operator yields true when both the operands are true.
24. _____ operator gives the opposite of the operated operand.
25. NAND is a logical operator used in Python. (True or False)

7. MEMBERSHIP OPERATORS

There are times when you need to test whether a value is present in the given list or other sequence data type. For this, Python provides membership operators. There are two membership operators under membership.

in operator: The operator yields true when the given value is found in the sequence.

Syntax:

```
x in y
```

not in operator: The operator is used when the given value is not present in the tested sequence.

Syntax:

```
x not in y
```

Example:

```
a = 3
y = {3, 4, 'a', 'b', 5}
print (a in y)
#The output of the code will be True.
```

There is another special type of operator called the dot operator. The symbol ".", is used to access the instance variables and methods of class objects.

Syntax:

```
student.age
```

Let's look at some examples:

Accessing Object Attributes

You can use the dot operator to access the attributes or properties of an object. For instance, if you have an object of a class, you can use the dot operator to access the class's attributes.

```
class MyClass:
  def __init__(self):
```

```
        self.my_attribute = "Hello, world!"
my_object = MyClass()
print(my_object.my_attribute)  # Output: Hello, world!
```

In this example, my_object.my_attribute uses the dot operator to access the my_attribute attribute of the my_object object.

Calling Methods

The dot operator can also be used to call methods of an object.

```
class MyClass:
    def my_method(self):
        return "Hello, world!"
my_object = MyClass()
print(my_object.my_method())  # Output: Hello, world!
```

Here, my_object.my_method() uses the dot operator to call the my_method method of the my_object object.

Accessing Modules

Python's dot operator is also used to access functions, classes, or variables defined in a module.

```
import math
print(math.pi)  # Output: 3.141592653589793
print(math.sqrt(16))  # Output: 4.0
```

In this example, math.pi and math.sqrt use the dot operator to access the pi constant and the sqrt function in the math module.

SELF-ASSESSMENT QUESTIONS – 6

26. When a value is present in a sequence, then the "not in operator" will provide _____ result.
27. What will be the output of the code that is given below?

    x = "I am a Python Code"

    print ('o' in x)

    A. True
    B. False
    C. Python
    D. None of the above
28. Membership operators can be operated on dictionary data types. (True or False)
29. Membership operators can provide only _____ and _____ as output.
30. _____ is the symbol of the dot operator.

8. IDENTITY OPERATORS

Identity operators check whether the identity of two operators is the same or not. They do this by checking the address where the data is stored in the memory. There are two such identity operators. These are explained below.

is operator: It yields the value as true when the operands are identical and have the same memory address.

Syntax:

```
x is b
```

is not operator: It provides the result as true when the operands are not identical and have different memory locations.

Syntax:

```
x is not b
```

Example:

```
a = 6
b = 6
print (a is b)
print (a is not b)
#The output of this will be as follows: True False
```

SELF-ASSESSMENT QUESTIONS – 7

31. What will be the output of the following program?

```
a = [1, 2, 3]
b = [1, 2, 3]
print (a is b)
```

   A. True
   B. False
   C. [1, 2, 3]
   D. None of the above

32. is not operator cannot operate on strings. (True or False)
33. Identity of two lists can never be same. (True or False)

9. OPERATOR PRECEDENCE

The combination of values, variables, functions, and operators is called an expression. The interpreter in Python can test and interpret whether an expression is valid or not. When an expression includes more than one operator, then the interpreter operators them as per a precedence level.

An arithmetic expression in Python when written without any parenthesis will be calculated from left to right. All the arithmetic operators into different levels as per their order of precedence. It is through the order of precedence of an operator that its priority is decided. The precedence of operators in Python is listed in the table below.

TABLE 3: ORDER OF PRECEDENCE

Operator	Name
()	Parentheses
******	Exponent
+x, -x, ~x	Unary plus, unary minus, and Bitwise NOT
***, /, //, %**	Multiplication, division, floor division, modulus
+, -	Addition and subtraction
<<, >>	Bitwise left and right shift operators
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
==, !=, <, >, >=, <=, is, is not, in, in not	Comparison operators, identity operators, membership operators
not	Logical NOT
and	Logical AND
or	Logical OR

Note that some operators have equal precedence. In such cases, they are evaluated from left to right. This is called operator associativity.

SELF-ASSESSMENT QUESTIONS – 8

34. Which is the order of precedence in Python?

    a) Parentheses

    b) Exponential

    c) Multiplication

    d) Division

    e) Addition

    f) Subtraction

    A. a, b, c, d, e, f

    B. b, a, c, d, e, f

    C. b, a, d, c, e, f

    D. a, b, c, d, f, e

35. Which of the following have the same precedence level?

    A. Addition and subtraction

    B. Multiplication, division, and addition

    C. Multiplication, division, addition, and subtraction

    D. Addition and multiplication

36. Which of the following has the highest precedence?

    A. Exponential

    B. Addition

    C. Multiplication

    D. Parentheses

37. Operators with same precedence are evaluated _____ to _____.

38. _____ and division operators have the same precedence level.

10. SUMMARY

- Operators in Python are used to perform operations on the values and variables.
- There are standard symbols for each operator used to perform logical and arithmetic operations.
- Arithmetic operations are used to perform mathematical operations such as addition, subtraction, division, and multiplication.
- Floor division gives an integer answer no matter the data type of the operands.
- Comparison operators compare the given two operands to give true and false as results.
- Logical operators are used to performing Boolean operations such as AND, NOT, and OR.
- Bitwise operators are used to performing various actions on bits. Integers are converted into binary form before being operated by bitwise operators.
- Assignment variables are used to assign values to the variables. Most commonly, "=" is used in Python.
- Assignment variables can help shorten the code by providing a shorthand of expressions.
- Identity operators are used to checking whether two variables belong to the same memory location.
- Membership operators are used to testing whether a variable is present in the given sequence data type or not.
- The dot operator is used to access a data member present in a class.
- The operators used in Python are operated based on their priority level in order of precedence. It is known as PEDMAS in short.
- PEDMAS means Parentheses, Exponent, Division, Multiplication, Addition, and Subtraction.
- When operators are used without parentheses in an expression, then they are evaluated from left to right if they have the same precedence level. This is called the associativity of operators.

11. GLOSSARY

- **Operand:** The part of the computer instruction that defines the data which is to be manipulated or operated on.
- **Operators:** The special symbols in Python that carry out arithmetic, logical, or relational computation.
- **Logical expression:** It is a statement that can be either true or false.
- **Truth table:** It is a table that shows the truth-value of a logical expression.
- **Conditional operator:** Also known as a ternary operator, the conditional operator is used to evaluate a condition applied to one or more Boolean expressions.
- **Increment operator(+=)** It is a unary operator that adds one from the operand.
- **Decrement operator(-=):** The unary operator that subtracts one from the operand.
- **Dot operator:** It is a member operator that is used to access the member of a class.
- **Operator precedence:** The order of operations that dictates which procedures need to be performed first.
- **Associativity:** It determines how the operators of the same precedence level will be processed in the absence of parentheses.
- **Identity operators:** Operator used to compare two or more objects that have the same memory location.

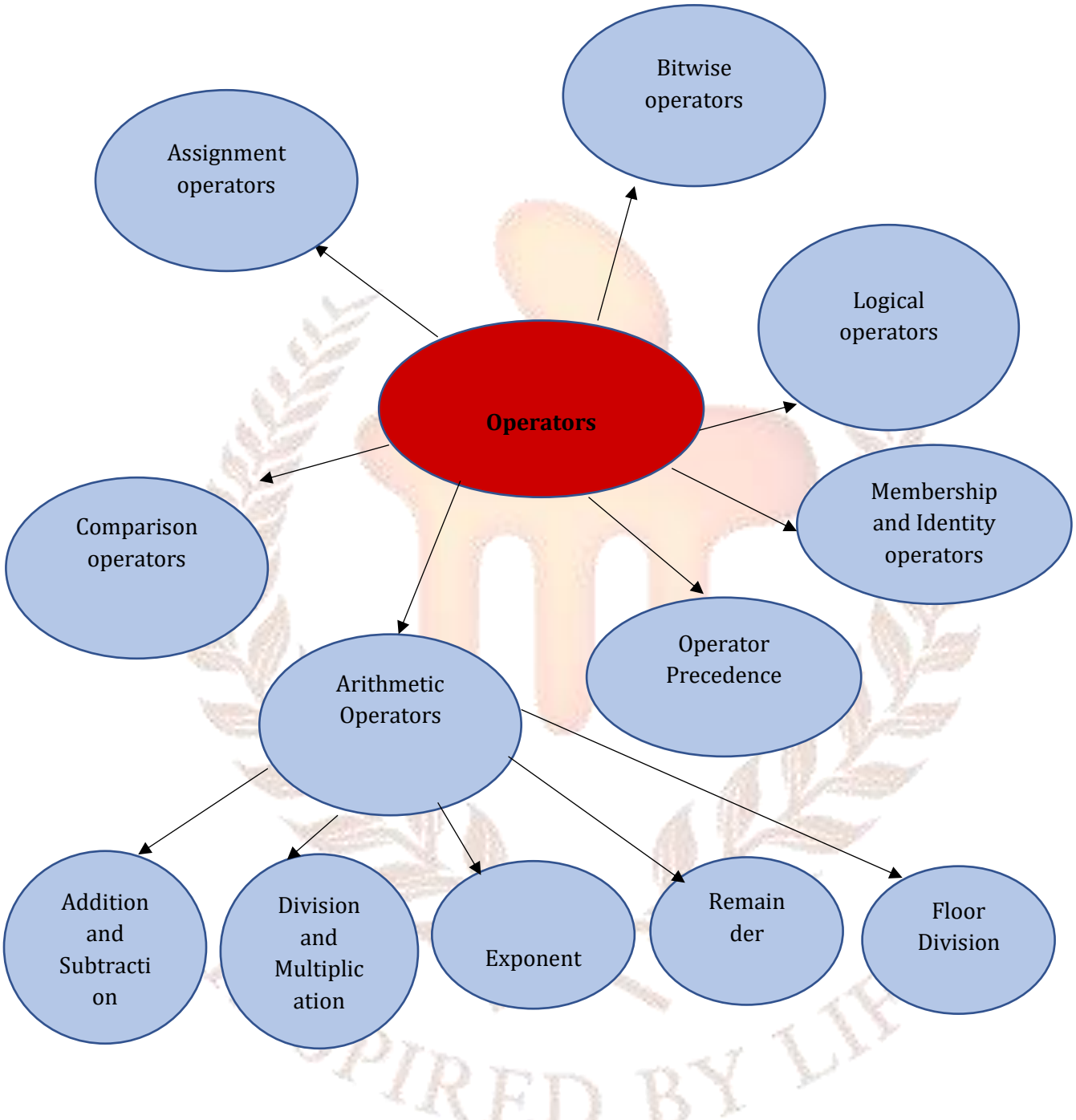

Fig 1: Conceptual Map

12. CASE STUDY

Results calculation using Python

Each professor can enter the marks obtained by a student on the website and the final result will be automatically calculated. The final result will include the total marks obtained by a student, their percentage, average score, rank, and grade.

The result should also be able to take in values when a student was unable to attend an exam. A team has been created to code a program that can perform all these functions while also creating a merit list in the order of highest to lowest marks.

The program should take in the values for each student from different professors and compute the percentage and other mathematical data.

The program should be capable of deciding whether a student has passed or not as per the percentage and grade obtained.

Imagine yourself as a part of the team that has been tasked to code the program. Based on this situation, answer the following questions.

***Source**: bestpy.com*

Questions

1. Which operators will be used to create the program and calculate all the required values? Discuss their types.
2. Discuss how the code can be simplified to ensure that all users can understand and use it.

13. TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS

Q1. Write a Python program that accepts the value of the radius of a circle and gives the area of the circle in output.

Q2. Write a Python program that accepts the value of an integer and computes the value of n + nn + nnn.

Q3. Write a Python program that calculates the number of days between two given dates.

Q4. Write a Python program that gives the volume of a sphere with the radius entered by the user.

Q5. Write a Python program that adds three numbers entered by the user and then returns a value that is three times their sum.

LONG ANSWER QUESTIONS

Q1. Write a Python program that finds the difference between the number input by the user and 90. If the number is greater than 90, then the program will yield a value that is double the absolute difference.

Q2. Write a Python program that takes in the value of the base and height of a triangle and calculates its area.

Q3. Write a Python program that calculates the lowest common multiple of the given two positive integers.

Q4. Write a Python program that solves the equation: (x + y)^2 * (x – y)

Q5. Write a Python program that calculates the amount received on a principal amount with the given rate of interest applicable over the given number of years.

14. ANSWERS

SELF ASSESMENT QUESTIONS:

1. B. x**y
2. //
3. Integer
4. %
5. True
6. D. <=
7. B. false
8. A. true
9. !=
10. True
11. +=
12. =
13. *=
14. &=
15. C. 6
16. <<
17. ^
18. Binary
19. Bitwise left shift
20. Bitwise NOR
21. B. false
22. A. True
23. AND
24. NOT
25. False
26. True
27. A. True
28. True

29. True and False
30. .
31. FALSE
32. FALSE
33. False
34. A. a, b, c, d, e, f
35. A. Addition and subtraction
36. D. Parentheses
37. left to right
38. Multiplication

TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS:

Answer 1:

```
from math import pi
r = float (input ("Input the radius of the circle : "))
print ("The area of the circle with radius " + str(r) + " is: " + str (pi * r**2))
```

Answer 2:

```
a = int (input ("Input an integer : "))
n1 = int (“%s" % a )
n2 = int (“%s%s" % (a,a) )
n3 = int (“%s%s%s" % (a,a,a) )
print (n1+n2+n3)
```

Answer 3:

```
from datetime import date
f_date = date (2014, 7, 2)
l_date = date (2014, 7, 11)
delta = l_date - f_date
print(delta.days)
```

Answer 4:

```
pi = 3.1415926535897931
r= float (input (“Enter the value of radius of the sphere: ”))
V= 4.0/3.0*pi* r**3
print ('The volume of the sphere is: ',V)
```

Answer 5:

```
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
num3 = int(input("Enter the third number: "))
sum_of_numbers = num1 + num2 + num3
result = 3 * sum_of_numbers
print("The result is:", result)
```

LONG ANSWER QUESTIONS:

Answer 1:

```
def difference(n):
    if n <= 90:
        return 90 - n
    else:
        return (n - 90) * 2
print (difference (22))
print (difference (14))
```

Answer 2:

```
b = int (input ("Input the base : "))
h = int (input ("Input the height : "))
area = b*h/2
print ("area = ", area)
```

Answer 3:

```
def lcm (x, y):
   if x > y:
```

```
    z = x
 else:
    z = y
 while (True):
    if ((z % x == 0) and (z % y == 0)):
       lcm = z
       break
    z += 1
 return lcm
print (lcm (4, 6))
print (lcm (15, 17))
```

Answer 4:

```
x, y = 4, 3
a = x + y
b = x - y
result = a * a * b
print ("({} + {}) ^ 2) = {}".format (x, y, result))
```

Answer 5:

```
amt = 10000
int = 3.5
years = 7
future_value = amt*((1+(0.01*int)) ** years)
print(round(future_value,2))
```

14. BOOKS AND E-REFERENCES

Books:

- Eric Matthes(2016), Python Crash Course: A Hands-On, Project-Based Introduction to Programming.
- John M. Zelle (2009), Python Programming: An Introduction to Computer Science (Preliminary Second Edition).
- Mark Lutz (2011), Python Programming: A Powerful Object-Oriented Programming (Fourth Edition).
- Sebastian Raschka (2017), Python Machine Learning - Machine Learning and Deep Learning with Python (Edition 2)

E-References:

- Python Programming Certification Training Course, last viewed on March 23, 2021< https://www.edureka.co/python-programming-certification-training >
- Python Tutorials and Sample Programs, last viewed on March 23, 2021 < https://www.w3schools.com/python/ >
- History of Python, last viewed on March 23, 2021 < https://en.wikipedia.org/wiki/History_of_Python >