**Python question and answers**

1. **What is a keyword in Python?**

* A keyword in Python is a reserved word that has a predefined meaning and purpose. These words cannot be used for anything other than their intended function in the language.

2. **List five keywords in Python and describe their use.**

* if: Used for conditional statements.
* else: Used in conjunction with if to define an alternative block of code if the condition is false.
* for: Used to create a for loop, which iterates over a sequence.
* while: Used to create a while loop, which repeats as long as a condition is true.
* def: Used to define a function.

3. **What is an identifier in Python?**

* An identifier is a name given to entities like variables, functions, classes, etc. It helps to identify these entities uniquely.

4. **What are the rules for naming identifiers in Python?**

* Must start with a letter (a-z, A-Z) or an underscore (\_).
* Can have letters, digits (0-9), and underscores after the first character.
* Cannot be a keyword.
* Case-sensitive (e.g., myVar and myvar are different).

5. **What is the difference between a keyword and an identifier?**

* A keyword is a reserved word with a specific meaning in the language and cannot be used for other purposes.
* An identifier is a user-defined name for variables, functions, etc., and must follow the naming rules.

6. **Explain the concept of a variable in Python.**

* A variable in Python is a named location used to store data. The value stored in a variable can be changed during program execution.

7. **How is a variable different from an identifier?**

* A variable is an identifier used to store data. An identifier is a broader term that refers to any name given to program elements like variables, functions, etc.

8. **What is an operator in Python? List the types of operators available.**

* An operator is a symbol that performs an operation on one or more operands. Types of operators include:
  + Arithmetic Operators (+, -, \*, /, etc.)
  + Comparison Operators (==, !=, >, <, etc.)
  + Logical Operators (and, or, not)
  + Assignment Operators (=, +=, -=, etc.)
  + Bitwise Operators (&, |, ^, ~, etc.)
  + Membership Operators (in, not in)
  + Identity Operators (is, is not)

9. **Explain the difference between the '==' operator and the '=' operator in Python.**

* == is the equality operator, used to compare two values for equality.
* = is the assignment operator, used to assign a value to a variable.

10. **Describe the precedence of operators in Python.**

* Operator precedence determines the order in which operations are evaluated. For example, multiplication has higher precedence than addition, so in the expression 2 + 3 \* 4, the multiplication is performed first.

11. **What does the 'or' operator do in Python?**

* The or operator is a logical operator that returns True if at least one of the operands is True. If both are False, it returns False.

**Practical questions**

**11. Declare and initialize three variables with different data types:**

# Variable declarations

int\_var = 10 # Integer

float\_var = 3.14 # Float

str\_var = "Hello" # String

print (int\_var, float\_var, str\_var)

**12.Demonstrate the use of arithmetic operators:**

# Arithmetic operations

a = 10

b = 5

print ("Addition:", a + b) # 15

print ("Subtraction:", a - b) # 5

print ("Multiplication:", a \* b) # 50

print ("Division:", a / b) # 2.0

print ("Modulus:", a % b) # 0

print ("Exponentiation:", a \*\* b) # 100000

print ("Floor Division:", a // b) # 2

**13. Demonstrate the use of comparison operators:**

# Comparison operations

x = 10

y = 20

print ("x == y:", x == y) # False

print ("x! = y:", x! = y) # True

print ("x > y:", x > y) # False

print ("x < y:", x < y) # True

print ("x >= y:", x >= y) # False

print ("x <= y:", x <= y) # True

**14. Demonstrate the use of logical operators:**

# Logical operations

a = True

b = False

print ("a and b:", a and b) # False

print ("a or b:", a or b) # True

print ("not a:", not a) # False

**15. Use the 'is' and 'is not' operators:**

# Identity operations

a = [1, 2, 3]

b = [1, 2, 3]

c = a

print ("a is b:", a is b) # False (different objects)

print ("a is c:", a is c) # True (same object)

print ("a is not b:", a is not b) # True (different objects)

print ("a is not c:", a is not c) # False (same object)

**16. Demonstrate the use of assignment operators:**

# Assignment operations

x = 5

x += 3 # x = x + 3

print ("x += 3:", x) # 8

x -= 2 # x = x - 2

print ("x -= 2:", x) # 6

x \*= 4 # x = x \* 4

print ("x \*= 4:", x) # 24

x /= 6 # x = x / 6

print ("x /= 6:", x) # 4.0

x %= 3 # x = x % 3

print ("x %= 3:", x) # 1.0

**17. Demonstrate the use of bitwise operators:**

# Bitwise operations

a = 10 # 1010 in binary

b = 4 # 0100 in binary

print ("a & b:", a & b) # 0 (bitwise AND)

print ("a | b:", a & b) # 0 (bitwise OR)

print ("a ^ b:", a ^ b) # 14 (bitwise XOR)

print ("~a:", ~a) # -11 (bitwise NOT)

print ("a << 1:", a << 1) # 20 (left shift)

print ("a >> 1:", a >> 1) # 5 (right shift)

**18. Demonstrate the use of membership operators:**

# Membership operations

my\_list = [1, 2, 3, 4, 5]

print ("2 in my\_list:", 2 in my\_list) # True

print ("6 not in my\_list:", 6 not in my\_list) # True

**19. Function to count the number of times a character appears in a string:**

def count\_char\_in\_string(input\_string, char):

return input\_string.count(char)

# Example usage

result = count\_char\_in\_string("hello world", "l")

print("Character 'l' appears:", result, "times")

**20. Swap the values of two variables without using a third variable:**

# Swapping without a third variable

a = 5

b = 10

a, b = b, a

print("After swapping: a =", a, "b =", b)

Top of Form

Bottom of Form