Questions

Q1. Why do we call Python as a general purpose and high-level programming language?

Q2. Why is Python called a dynamically typed language?

Q3. List some pros and cons of Python programming language?

Q4. In what all domains can we use Python?

Q5. What are variable and how can we declare them?

Q6. How can we take an input from the user in Python?

Q7. What is the default datatype of the value that has been taken as an input using input() function?

Q8. What is type casting?

Q9. Can we take more than one input from the user using single input() function? If yes, how? If no, why?

Q10. What are keywords?

Q11. Can we use keywords as a variable? Support your answer with reason.

Q12. What is indentation? What's the use of indentaion in Python?

Q13. How can we throw some output in Python?

Q14. What are operators in Python?

Q15. What is difference between / and // operators?

Q16. Write a code that gives following as an output.

```
```

iNeuroniNeuroniNeuroniNeuron

```

Q17. Write a code to take a number as an input from the user and check if the number is odd or even.

Q18. What are boolean operator?

Q19. What will the output of the following?

```

1 or 0

0 and 0

True and False and True

1 or 0 or 0

```

Q20. What are conditional statements in Python?

Q21. What is use of 'if', 'elif' and 'else' keywords?

Q22. Write a code to take the age of person as an input and if age >= 18 display "I can vote". If age is < 18 display "I can't vote".

Q23. Write a code that displays the sum of all the even numbers from the given list.

```

numbers = [12, 75, 150, 180, 145, 525, 50]

```

Q24. Write a code to take 3 numbers as an input from the user and display the greatest no as output.

Q25. Write a program to display only those numbers from a list that satisfy the following conditions
```

- The number must be divisible by five

- If the number is greater than 150, then skip it and move to the next number

- If the number is greater than 500, then stop the loop

```
numbers = [12, 75, 150, 180, 145, 525, 50]
```

Answers

Q1. Python is called a general-purpose programming language because it is designed to be used for a wide range of applications. It provides support for various programming paradigms, such as procedural, object-oriented, and functional programming. Python is not limited to any specific domain or application and can be used for developing web applications, scientific computing, data analysis, artificial intelligence, automation, scripting, and more.

Python is considered a high-level programming language because it provides abstractions that hide low-level details of computer hardware and memory management. It has a syntax that is closer to natural language, making it easier for humans to read and write code. High-level languages like Python abstract away complex tasks, allowing developers to focus on problem-solving rather than dealing with low-level implementation details.

Q2. Python is called a dynamically typed language because the type of a variable is determined at runtime. In Python, you don't need to explicitly declare the type of a variable when you define it. The type of a variable can change dynamically during the execution of a program based on the value assigned to it. For example, you can assign an integer value to a variable initially and then later assign a string value to the same variable without any type declarations or conversions.

Q3. Pros of Python:

- Readability: Python has a clean and readable syntax that emphasizes code readability and reduces the cost of program maintenance.

- Large Standard Library: Python comes with a comprehensive standard library that provides a wide range of modules and functions for various tasks, minimizing the need for external dependencies.

- Portability: Python programs can run on different operating systems without any or minimal modifications.

- Easy Integration: Python can easily integrate with other languages like C/C++, Java, and .NET, allowing developers to leverage existing codebases and libraries.

- Rapid Development: Python's simplicity and productivity features, such as dynamic typing and automatic memory management, enable faster development cycles.

- Vast Community and Ecosystem: Python has a large and active community of developers who contribute to open-source libraries and frameworks, providing extensive support and resources.

Cons of Python:

- Performance: Python is an interpreted language, which can make it slower compared to compiled languages for certain tasks. However, optimizations and third-party libraries can mitigate this issue.

- Global Interpreter Lock (GIL): Python's Global Interpreter Lock allows only one thread to execute Python bytecode at a time, which can limit the performance of CPU-bound multithreaded programs. However, Python provides alternatives for concurrency, such as multiprocessing and asynchronous programming, to overcome this limitation.

- Mobile and Game Development: Python is not widely used for mobile app development or high-performance game development compared to languages like Java, Swift, or C++.

Q4. Python can be used in various domains, including:

- Web Development: Python frameworks like Django and Flask are popular for building web applications and APIs.

- Data Analysis and Scientific Computing: Python provides libraries like NumPy, Pandas, and SciPy that are widely used in data analysis, machine learning, and scientific computing.

- Artificial Intelligence and Machine Learning: Python has extensive support for machine learning frameworks such as TensorFlow, Keras, and PyTorch, making it a popular choice for AI development.

- Automation and Scripting: Python's simplicity and readability make it ideal for writing scripts and automating repetitive tasks.

- DevOps and System Administration: Python is often used for writing scripts to automate system administration tasks and for managing infrastructure using tools like Ansible and SaltStack.

- Game Development: Python has game development libraries like Pygame and Panda3D for creating 2D and 3D games.

- Internet of Things (IoT): Python's simplicity and availability of libraries make it suitable for IoT prototyping and development.

- Desktop GUI Applications: Python provides frameworks like Tkinter,

PyQt, and wxPython for building desktop GUI applications.

Q5. A variable is a named reference to a value stored in computer memory. In Python, you can declare a variable by assigning a value to it using the assignment operator (=). Here's an example of variable declaration:

```
x = 10
```

In this example, `x` is the variable name, and `10` is the value assigned to the variable. Python does not require explicit variable type declarations.

Q6. In Python, you can take input from the user using the `input()` function. The `input()` function reads a line of text entered by the user and returns it as a string. Here's an example:

```
name = input("Enter your name: ")
```

In this example, the user is prompted to enter their name, and the input is stored in the `name` variable.

Q7. The default datatype of the value returned by the `input()` function in Python is a string. Regardless of what the user enters, the `input()` function always returns a string.

Q8. Type casting, also known as type conversion, is the process of converting a value from one data type to another. In Python, you can perform type casting using built-in functions like `int()`, `float()`, `str()`, etc. For example, you can convert a string to an integer using the `int()` function:

```
num_str = "10"
num_int = int(num_str)
```

In this example, the variable `num_str` contains the string "10", and `num_int` contains the integer 10 after type casting.

Q9. No, you cannot take more than one input from the user using a single `input()` function call. The `input()` function reads a single line of text as input. If you need to take multiple inputs, you can use multiple `input()` statements or split the input line into separate values using string manipulation or the `split()` function.

Q10. Keywords, also known as reserved words, are the words that have special meanings in a programming language. These words are predefined and cannot be used as variable names or other identifiers. In Python, examples of keywords include `if`, `else`, `for`, `while`, `def`, `class`, `import`, `and`, `or`, etc.

Q11. No, you cannot use keywords as variable names in Python. Keywords have predefined meanings in the language and are reserved for specific purposes. If you try to use a keyword as a variable name, you will get a syntax error.

For example, the following code will raise a syntax error:

```
if = 10
```

In this case, `if` is a keyword used for conditional statements, and it cannot be used as a variable name.

Q12. In Python, indentation refers to the spaces or tabs placed at the beginning of a line of code to indicate its grouping or indentation level. Python uses indentation to define the structure and hierarchy of code blocks, such as loops, conditional statements, and function definitions.

Indentation is essential in Python because it replaces the traditional use of curly braces or keywords like `begin` and `end` in other programming languages. It ensures that the code is visually organized and improves readability. Incorrect indentation can lead to syntax errors or change the logical meaning of the code.

Q13. In Python, you can throw output or display information using the `print()` function. The `print()` function takes one or more arguments and prints them to the standard output (usually the console). Here's an example:

```
print("Hello, World!")
```

This code will output the string "Hello, World!" to the console.

Q14. Operators in

Python are symbols or special characters that perform operations on operands (variables or values). Python supports various types of operators, including arithmetic, assignment, comparison, logical, bitwise, and more.

Examples of operators in Python:

- Arithmetic operators: `+`, `-`, `*`, `/`, `%`, `//`, `**`
- Assignment operators: `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `//=`, `**=`
- Comparison operators: `==`, `!=`, `<`, `>`, `<=`, `>=`
- Logical operators: `and`, `or`, `not`
- Bitwise operators: `&`, `|`, `^`, `~`, `<<`, `>>`

Q15. In Python, the `/` operator performs floating-point division, while the `//` operator performs integer division or floor division.

For example:

- `10 / 3` returns `3.3333333333333335`

- `10 // 3` returns `3`

The `/` operator always returns a floating-point result, even if the operands are integers. The `//` operator, on the other hand, discards the fractional part of the result and returns the integer quotient.

Q16. Here's a code that gives the desired output:

```python
word = "Neuron"
output = "i" + word[1:] + word
print(output)
```

This code takes the string "Neuron" and concatenates it with "i" and a sliced portion of the word starting from the second character (`word[1:]`). The resulting output is "iNeuroniNeuroniNeuroniNeuron".

Q17. Here's a code to take a number as input from the user and check if it's odd or even:

```python
number = int(input("Enter a number: "))
if number % 2 == 0:
    print("The number is even.")
else:
    print("The number is odd.")
```

In this code, the input number is divided by 2 using the modulo operator (`%`). If the remainder is 0, it means the number is divisible by 2 and hence even. Otherwise, it's odd.

Q18. Boolean operators in Python are used to perform logical operations on Boolean values (`True` and `False`). The three main Boolean operators are:

- `and`: Returns `True` if both operands are `True`, otherwise returns `False`.

- `or`: Returns `True` if at least one of the operands is `True`, otherwise returns `False`.

- `not`: Returns the opposite Boolean value of the operand. If the operand is `True`, it returns `False`, and vice versa.

Q19. The output of the following expressions would be:

- `1 or 0` returns `1` because the `or` operator evaluates to the first `True` operand it encounters.

- `0 and 0` returns `0` because the `and` operator evaluates to the first `False` operand it encounters.

- `True and False and True` returns `False` because the `and` operator requires all operands to be `True` for the result to be `True`.

- `1 or 0 or 0` returns `1` because the `or` operator evaluates to the first `True` operand it encounters.

Q20. Conditional statements in Python are used to execute different blocks of code based on certain conditions. The main conditional statements in Python are `if`, `elif` (short for "else if"), and `else`.

Q21. The `if`, `elif`, and `else` keywords are used in conditional statements to control the flow of the program based on

specified conditions.

- `if` statement: It checks a condition and executes a block of code if the condition is true. If the condition is false, the code block is skipped.

- `elif` statement: It allows you to check additional conditions after the `if` statement. If the previous conditions are false, the `elif` statement checks its condition and executes the corresponding block of code if true.

- `else` statement: It is optional and comes after the `if` and `elif` statements. It is executed only if all previous conditions are false. It provides a default block of code to be executed when no other conditions are met.

Q22. Here's a code that takes the age of a person as input and displays whether they can vote or not:

```python
age = int(input("Enter your age: "))
if age >= 18:
    print("I can vote.")
else:
    print("I can't vote.")
```

If the age is greater than or equal to 18, it prints "I can vote." Otherwise, it prints "I can't vote."

Q23. Here's a code that calculates the sum of all the even numbers from the given list:

```python
numbers = [12, 75, 150, 180, 145, 525, 50]
sum_even = 0

for num in numbers:
    if num % 2 == 0:
        sum_even += num

print("Sum of even numbers:", sum_even)
```

In this code, a variable `sum_even` is initialized to 0. The `for` loop iterates over each number in the `numbers` list. If the number is even (divisible by 2), it adds it to the `sum_even` variable. Finally, it prints the sum of even numbers.

Q24. Here's a code that takes 3 numbers as input from the user and displays the greatest number as output:

```python
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
num3 = int(input("Enter the third number: "))

max_num = max(num1, num2, num3)
print("The greatest number is:", max_num)
```

In this code, three numbers are taken as input using the `input()` function and stored in variables `num1`, `num2`, and `num3`. The `max()` function is then used to find the maximum of the three numbers, and the result is stored in the `max_num` variable. Finally, the greatest number is displayed using the `print()` function.

Q25. Here's a code that displays only those numbers from a list that satisfy the given conditions:

```python
numbers = [12, 75, 150, 180, 145, 525, 50]
```

```
for num in numbers:
    if num % 5 == 0:
        if num > 150:
            continue
        elif num > 500:
            break
        print(num)
```

In this code, the `for` loop iterates over each number in the `numbers` list. It checks if the number is divisible by 5 using the modulo operator (`%`). If it is divisible by 5, it further checks the following conditions:

- If the number is greater than 150, it uses `continue` to skip the remaining statements and move to the next number.

- If the number is greater than 500, it uses `break` to exit the loop.

- If the number satisfies all the conditions, it is displayed using the `print()` function.