PYTHON QUESTION BANK

Q.1 What is Python ?
Answer: Python is a high-level, interpreted programming language known for its simplicity and readability. It was created by Guido van Rossum and first released in 1991. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Q.2 List what can you do with python programming language
Answer: Python is an incredibly versatile programming language, and you can do a wide range of tasks with it. Here's a list of some of the things you can do with Python:
a. Web Development
b. Data Science and Machine Learning
c. Game Development
d. Desktop GUI Applications
e. Mobile App Development
f. DevOps and Cloud Computing
g. Cyber Security
h. GIS (GEOGRAPHIC INFORMATION SYSTEM)
i. Network Programming
j. A.I (Artificial Intelligence)

Q.3 Write basic syntax of python programming language.
Answer: Use `if`, `elif`, and `else` for conditional execution.

```
 if age > 18:
   print("Adult")
elif age > 12:
   print("Teenager")
else:
   print("Child")
```

Q.4 What is numeric data type in python?
Answer: In Python, the numeric data type is used to store numeric values. There are mainly three numeric data types:

1. **Integers (int)**:
   - They are whole numbers, positive or negative, without decimals, of unlimited length.
   - Example: -5, 0, 10, 100
2. **Floating-point numbers (float)**:
   - They are numbers with a decimal point or use an exponential (e) to define the number.
   - Example: -5.0, 0.0, 10.5, 1.6e2 (which is equivalent to 160.0)
3. **Complex numbers (complex)**:
   - They are written with a "j" as the imaginary part.
   - Example: 3 + 4j, 5.2 - 2.3j

Q.5 List the different arithmetic operator in python.

Answer: In Python, you can perform arithmetic operations using various arithmetic operators. Here's a list of the different arithmetic operators in Python:

1. Addition (+)

- Adds two operands.

Example

result = 5 + 3 print(result) # Output: 8

2. Subtraction (-)

- Subtracts the right-hand operand from the left-hand operand.

Example

result = 5 - 3 print(result) # Output: 2

3. Multiplication (*)

- Multiplies two operands.

Example

result = 5 * 3 print(result) # Output: 15

4. Division (/)

- Divides the left-hand operand by the right-hand operand (always returns a float).

Example

result = 10 / 3 print(result) # Output: 3.3333333333333335

5. Floor Division (//)

- Divides the left-hand operand by the right-hand operand and returns the floor value (drops the decimal part).

Example

result = 10 // 3 print(result) # Output: 3

6. Modulus (%)

- Returns the remainder when the left-hand operand is divided by the right-hand operand.

Example

result = 10 % 3 print(result) # Output: 1

7. Exponentiation (**)

- Raises the left-hand operand to the power of the right-hand operand.

Example

result = 2 ** 3 print(result) # Output: 8

Q.6 Explain python in details with features, advantages and disadvantages.

Answer: Certainly! Let's dive deeper into Python by exploring its features, advantages, and disadvantages.

Features of Python

1. Easy-to-Read Syntax

- Python's syntax is clear and easy to understand, making it accessible for beginners and experts alike.

2. Interpreted Language

- Python is an interpreted language, meaning that code is executed line by line, which can make debugging easier.

3. Dynamic Typing

- Python uses dynamic typing, allowing for more flexibility in variable types.

4. Extensive Standard Library

- Python has a large standard library that provides a wide range of modules and functions for tasks like file I/O, networking, and web development.

5. Cross-Platform

- Python is cross-platform, meaning it can run on various operating systems like Windows, macOS, and Linux.

6. Object-Oriented

- Python supports object-oriented programming (OOP), which allows for the creation and manipulation of objects.

7. Interoperability
   - Python can be integrated with other languages like C, C++, and Java, which makes it versatile for a variety of applications.
8. High-Level Language
   - Python is a high-level language, which means it abstracts most of the complex details from the user, making it easier to write and maintain code.
9. Community and Ecosystem
   - Python has a large and active community of developers and users, which contributes to the development of libraries, frameworks, and tools.
10. Scalable and Extensible
    - Python supports modular and functional programming, making it scalable and extensible for large projects.

Advantages of Python

1. Readability
   - Python's syntax and readability make it easier to understand and write code, reducing the cost of program maintenance and development.
2. Large Standard Library
   - Python's extensive standard library provides a rich set of modules and functions, reducing the need for code to perform common tasks.
3. Diverse Programming Paradigms
   - Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming, providing flexibility in coding.
4. Community Support
   - Python has a large and active community, which contributes to the development of libraries, frameworks, and tools, and provides support through forums and communities.

Disadvantages of Python

1. Performance
   - Python is an interpreted language, which can result in slower performance compared to compiled languages like C and C++.
2. Global Interpreter Lock (GIL)
   - In multi-threaded applications, the Global Interpreter Lock can be a limitation as it allows only one thread to execute at a time, which can impact performance.
3. Design Restrictions
   - Python's dynamic typing can lead to runtime errors, as the interpreter does not check the type of variables until runtime.
4. Mobile Development
   - Python is not as commonly used for mobile app development as languages like Java and Swift, although frameworks like Kivy and BeeWare are available.

Q.7 Explain features of python programming in details

Answer: Features of Python

1. Easy-to-Read Syntax
   - Python's syntax is clear and easy to understand, making it accessible for beginners and experts alike.
2. Interpreted Language
   - Python is an interpreted language, meaning that code is executed line by line, which can make debugging easier.
3. Dynamic Typing
   - Python uses dynamic typing, allowing for more flexibility in variable types.
4. Extensive Standard Library

- Python has a large standard library that provides a wide range of modules and functions for tasks like file I/O, networking, and web development.

5. Cross-Platform
   - Python is cross-platform, meaning it can run on various operating systems like Windows, macOS, and Linux.
6. Object-Oriented
   - Python supports object-oriented programming (OOP), which allows for the creation and manipulation of objects.
7. Interoperability
   - Python can be integrated with other languages like C, C++, and Java, which makes it versatile for a variety of applications.
8. High-Level Language
   - Python is a high-level language, which means it abstracts most of the complex details from the user, making it easier to write and maintain code.
9. Community and Ecosystem
   - Python has a large and active community of developers and users, which contributes to the development of libraries, frameworks, and tools.
10. Scalable and Extensible
    - Python supports modular and functional programming, making it scalable and extensible for large projects.

Q.8 Explain the basic data types in python language
Answer:
In Python, data types specify the type of value that a variable can hold. Python has several built-in data types that are used to define variables, including:

1. Integer (int)
   - **Definition**: Integers are whole numbers, positive or negative, without decimals, of unlimited length.
2. Floating-Point Number (float)
   - **Definition**: Floating-point numbers are numbers with a decimal point or use an exponential (e) to define the number.
3. Complex Number (complex)
   - **Definition**: Complex numbers are written with a "j" as the imaginary part.
4. String (str)
   - **Definition**: Strings are sequences of characters enclosed within single (' '), double (" "), or triple (''' ''' or """ """) quotes.
5. Boolean (bool)
   - **Definition**: Booleans represent one of two values: True or False.
6. List
   - Definition: **Lists are ordered collections of items. They can contain items of different data types and are mutable (modifiable).**

Q.9 Write a python program to calculate area of triangle.
Answer: Program:

```
#Take input from the user.
Base= int(input("Enter the base of the triangle: "))
Height= int(input("Enter the height of the triangle: "))
#Using the formula of area of triangle
Area=0.5*Base*Height
#Printing the output of the area of triangle
print("The area of the triangle is: ",Area)
```

Ouput:

```
Enter the base of the triangle:50
Enter the height of the triangle: 10
The area of the triangle is:  250.0
```

Q.10 Write a python program to swap two numbers without using third variable.
Answer: Program:

```
num1= int(input("Enter first number:"))
num2= int(input("Enter second number:"))

print("******Before Swapping******")
print("First number",num1)
print("Second number",num2)

#Swapping without using a third variables
num1=num1+num2
num2=num1-num2
num1=num1-num2

print("******After Swapping******")
print("First number:",num1)
print("Second number:",num2)
```

Output:

```
Enter first number:45
Enter second number:18
******Before Swapping******
First number 45
Second number 18
******After Swapping******
First number: 18
Second number: 45
```

Q.11 Write a syntax of nested if-else statement
Answer:

```
if condition1:
    # Code to execute if condition1 is True

    if condition2:
        # Code to execute if both condition1 and condition2 are True
    else:
        # Code to execute if condition1 is True and condition2 is False
else:
    # Code to execute if condition1 is False
```

Q.12 Write syntax for loop and while loop.
Answer:
Syntax for loop :

```
# for loop syntax
for <variable> in <iterable>:
    <statements>

Example:
```

```
# Print each item in a list
my_list = [1, 2, 3, 4, 5]
for item in my_list:
    print(item)
```

Syntax while loop:

```
# while loop syntax
while <condition>:
    <statements>

Example:
# Print numbers from 1 to 10
i = 1
while i <= 10:
    print(i)
    i += 1
```

Q.13 What do you mean by nesting loop

Answer: Nesting loops refers to the process of placing one loop inside another loop. This allows you to perform more complex iterations and can be particularly useful when you need to iterate over multiple sequences or when you need to create patterns or perform calculations involving multiple iterations.

Nested For Loop Syntax:

```
for variable1 in iterable1:
  # Code block for first loop

  for variable2 in iterable2:
    # Code block for second loop
```

Q.14 Define range function in python programming language.

Answer: The range() function in Python is a built-in function that returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

The syntax for the range() function is:

range(start, stop, step)

- start: (optional): The start value of the sequence. Defaults to 0.
- stop: (required): The end value of the sequence. The sequence will stop before this value.
- step: (optional): The increment value of the sequence. Defaults to 1.

The range() function is a powerful tool that can be used to generate sequences of numbers for a variety of purposes. It is commonly used in loops to iterate over a sequence of values.

Q.14 Write a python program to check entered number is even or odd

Answer: Program;

```
# Input the number
num = int(input("Enter a number: "))

# Check if the number is even or odd
if num % 2 == 0:
   print(f"{num} is an even number.")
else:
   print(f"{num} is an odd number.")
```

Output:
```
Enter a number: 25
25 is an odd number.
```

Q.15 Write a python program to find factorial of entered number

Answer: Program:
```
# Input the number
num = int(input("Enter a number: "))

# Initialize factorial to 1
factorial = 1

# Check if the number is negative, positive, or zero
if num < 0:
    print("Factorial does not exist for negative numbers.")
elif num == 0:
    print("The factorial of 0 is 1.")
else:
    for i in range(1, num + 1):
        factorial *= i

    print(f"The factorial of {num} is {factorial}.")
```

Output:
```
Enter a number: 5
The factorial of 5 is 120.
```

Q.16 What do you mean by continue, break and pass statement in details with example

Answer:

In Python, **continue**, **break**, and **pass** are control flow statements that alter the flow of the loop execution or provide a way to handle empty blocks of code. Let's look at each of them in detail:

1. continue Statement:

The **continue** statement is used to skip the current iteration of a loop and continue with the next iteration.

Syntax:
```
for variable in iterable:
    if condition:
        continue
    # Code to execute
```

2. break Statement:

The **break** statement is used to exit the loop prematurely, without completing all iterations.

Syntax:
```
for variable in iterable:
if condition:
break
# Code to execute
```

3. pass Statement:

The **pass** statement is a null operation, meaning nothing happens when it executes. It is used as a placeholder in empty blocks of code or to create a minimal class or function.

Syntax:

```
if condition:
pass
```

- **continue**: Skips the current iteration and continues with the next iteration of the loop.
- **break**: Exits the loop prematurely, without completing all iterations.
- **pass**: A null operation, used as a placeholder in empty blocks of code or to create minimal classes or functions.

These control flow statements are fundamental in controlling the flow of execution in loops and conditionals in Python.