1.Explain the key features of python that makes it a popular choice for programming.

Python is one of the most popular programming languages, known for its versatility and ease of use. Here are the key features that make Python so widely used:

1. Simplicity and Readability

- Python's syntax is clean, concise, and easy to understand, making it a great choice for beginners.
- The language emphasizes readability, which makes code easier to maintain and debug. Its structure avoids using complex symbols, and it often feels closer to pseudocode than traditional programming.

2. High-Level Language

- Python abstracts away much of the complexity involved with lower-level languages. Developers don't need to manage memory, or deal with machine-level operations, making development faster and more efficient.

3. Interpreted Language

- Python is interpreted, meaning that it executes code line by line, providing immediate feedback. This makes debugging and development cycles faster as there is no need to compile the code first.

4. Dynamically Typed

- Python is dynamically typed, meaning that variables do not need to be explicitly declared with a data type. This adds flexibility and reduces boilerplate code.

5. Extensive Libraries and Frameworks

- Python has an extensive standard library and a vast ecosystem of third-party packages and frameworks, covering everything from web development (Django, Flask) to data analysis (NumPy, pandas) and machine learning (TensorFlow, PyTorch).
- These libraries save time and effort by providing reusable, well-tested components.

6. Cross-Platform Compatibility

- Python is platform-independent, meaning code written on one operating system can easily run on another (with minimal adjustments). This flexibility makes it ideal for projects that need to work across multiple environments.

7. Community Support

- Python has one of the largest and most active programming communities. This means extensive documentation, tutorials, forums, and open-source projects are readily available. For any challenge, there's likely an existing solution or community to help.

8. Integration Capabilities

- Python easily integrates with other languages like C, C++, Java, and .NET, as well as various databases. This makes Python a great choice for scripting, automation, and embedding in larger systems.

9. Wide Range of Applications

- Python is versatile and can be used in a wide range of domains such as web development, automation, data science, artificial intelligence, game development, network programming, and more.

10. Open Source

- Python is free and open-source, which means anyone can use it, contribute to its development, and even modify it. This has led to rapid growth and innovation within the language and its ecosystem.

These features, combined with its flexibility and broad applicability, make Python a popular choice among both beginners and professional developers.

2. Describe the role of predefined keywords in python and provide examples of how they are used in a program.

Predefined keywords in Python are reserved words that have special meanings. These cannot be used as variable names, function names, or identifiers. They are integral to Python's syntax and structure. Python has 35 keywords (as of Python 3.9), such as if, else, for, while, def, return, and True.

Example:

```
# Using 'if', 'else', and 'elif'
 num = 5
 if num > 0:
    print("Positive")
elif num == 0:
    print("Zero")
else:
    print("Negative")

# Using 'for' keyword
for i in range(5):
  print(i)

# Using 'def' to define a function
def greet():
  print("Hello, World!")
```

3. compare and contrast mutable and immutable objects in python with examples.

Mutable vs Immutable Objects in Python

- Mutable objects: Objects whose value can be changed after creation. Examples include lists, dictionaries, and sets.
- Immutable objects: Objects whose value cannot be changed after creation. Examples include integers, strings, and tuples.

Example of Mutable Object (List):

```
my_list = [1, 2, 3]
my_list[0] = 10  # Changes the first element
print(my_list)  # Output: [10, 2, 3]
```

Example of Immutable Object (String):

```
my_string = "hello"
# Trying to change the first character results in an error
my_string[0] = 'H'  # Raises a TypeError
```

Comparison:

- Mutable objects: Can be altered in place without creating a new object.
- Immutable objects: Any modification results in a new object.

4. discuss the different types of operators in python and provide examples of how they are used.

Types of Operators in Python

Python has several types of operators:

- **Arithmetic operators**: +, -, *, /, // (floor division), % (modulus), ** (exponentiation)

  Example:

```
a, b = 10, 3
print(a + b)  # 13
```

```
print(a / b)  # 3.333...
print(a ** b)  # 1000 (10 raised to 3)
```

Comparison operators: ==, !=, >, <, >=, <=

Example:

```
print(a > b)  # True
print(a == b)  # False
```

Logical operators: and, or,not

Example:

```
print(a > 5 and b < 5)  # True
```

Assignment operators: =, +=, -=, *=, /=

Example:

```
a += 2  # a becomes 12
```

Bitwise operators: &, |, ^, ~, <<, >>

Example:

```
print(a & b)  # 2 (bitwise AND)
```

Membership operators: in, not in

Example:

```
my_list = [1, 2, 3]
print(2 in my_list)  # True
```

Identity operators: is, is not

Example:

```
Print(a is b) # False (compares object identity, not value)
```

5. explain the concept of type casting in python with example.

Type Casting in Python

Type casting refers to converting one data type to another. Python allows both implicit and explicit type casting.

- **Implicit type casting**: Python automatically converts one type to another when required.

```
a = 5
b = 2.5
result = a + b # Implicit conversion of 'a' to float
print(result) # 7.5
```

- **Explicit type casting**: Developers manually convert one data type to another.

```
num_str = "10"
num_int = int(num_str)  # Explicit conversion from string to integer
print(num_int + 5)  # 15
```

6. how do conditional statements work in python? Illustrate with examples.

Conditional Statements in Python

Conditional statements allow code execution based on conditions. Python supports if, elif, and else statements to control flow.

Example:

```
num = 10
if num > 0:
   print("Positive")
elif num == 0:
   print("Zero")
else:
   print("Negative")
```

This checks the condition num > 0 and executes the corresponding block if true. If not, it evaluates elif or else.

7. describe the different types of loops in python and their use cases with example.

Types of Loops in Python

Python has two main loop types:

- **for loop**: Iterates over a sequence (such as a list, string, range).

  Example:

```
for i in range(5):
   print(i)  # Output: 0 1 2 3 4
```

- **while loop**: Repeats as long as a condition is true.

  Example:

```
count = 0
while count < 5:
print(count)
 count += 1
```

Use Cases:

- Use a for loop when you know the number of iterations beforehand (e.g., iterating over a list).
- Use a while loop when the condition is evaluated dynamically (e.g., user input or a changing variable).