1. Explain the key features of Python that make it a popular choice for programming.

Python is a popular programming language due to below mentioned key features:

1. **Simple and Easy to Learn**: Python has a simple syntax similar to English, making it easy to learn and use.
2. **Interpreted Language**: Python is an interpreted language, meaning code is executed line by line, which makes debugging easier.
3. **Dynamically Typed**: Python does not require the declaration of variable types. The interpreter assigns the type automatically.
4. **Object-Oriented**: Python supports object-oriented programming, allowing for the creation of classes and objects.
5. **Extensive Libraries**: Python has a vast standard library that supports many modules and functions for various tasks.
6. **Platform-Independent**: Python code can run on various operating systems like Windows, macOS, and Linux without requiring changes.

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 a specific meaning and cannot be used for naming variables, functions, or other identifiers. They are part of Python's syntax. Some examples include:

- **if, else, elif:** Used for conditional statements.
- **for, while:** Used for loops.
- **def:** Used to define a function.
- **return:** Exits a function and returns a value.
- **class:** Used to define a class.

Example:

```
if x > 0:
   print("Positive")
else:
   print("Negative")
```

3. Compare and contrast mutable and immutable objects in Python with examples.

In Python, objects can be mutable or immutable:

Mutable Objects: These can be changed after their creation. Examples include lists, dictionaries, and sets.

Example:

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

Immutable Objects: These cannot be changed after their creation. Examples include strings, tuples, and integers.

Example:

```
my_tuple = (1, 2, 3)
# my_tuple[0] = 10  # This will raise a TypeError
print(my_tuple)  # Output: (1, 2, 3)
```

Mutable objects are useful when you need to change data, while immutable objects ensure the consistency and integrity of data.

4. Discuss the different types of operators in Python and provide examples of how they are used.

Python supports various types of operators, including:

1. **Arithmetic Operators:** Used for mathematical operations like +, -, *, /
   Example: x + y
2. **Comparison Operators:** Used for comparing values like ==, !=, >, <.
   Example: x > y
3. **Logical Operators:** Used for logical operations like and, or, not.
   Example: x > 0 and y < 10
4. **Assignment Operators:** Used for assigning values like =, +=, -=.
   Example: x += 1
5. **Bitwise Operators:** Used for binary operations like &, |, ^.
   Example: x & y
6. **Identity Operators:** Used to check if two variables are the same object like is, is not.
   Example: x is y
7. **Membership Operators:** Used to check if a value is present in a sequence like in, not in.

Example: x in my_list

5. Explain the concept of type casting in Python with examples.

Type casting in Python is the process of converting one data type to another. There are two types:

1. **Implicit Type Casting**: Done automatically by Python.

Example:

```
x = 5
y = 2.0
result = x + y  # x is implicitly converted to float
print(result)  # Output: 7.0
```

2. **Explicit Type Casting:** Done manually by the programmer.

Example:

```
x = "10"
y = int(x)  # Explicitly converting string to integer
print(y)  # Output: 10
```

Type casting is useful when there is a need to perform operations on different data types.

6. How do conditional statements work in Python? Illustrate with examples.

Conditional statements are used to execute code based on certain conditions. The main types are:

- if: Executes a block of code if the condition is true.
- elif: Executes a block of code if the previous conditions were false and the current condition is true.
- else: Executes a block of code if none of the conditions are true.

Example:

```
x = 10
if x > 0:
   print("Positive")
elif x == 0:
```

```
    print("Zero")
else:
    print("Negative")
```

7. Describe the different types of loops in Python and their use cases with examples.

Python supports two types of loops:

1. **for loop:** Used to iterate over a sequence (like a list, tuple, or string).
   Example:

```
for i in range(5):
    print(i)
```

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

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

Loops are useful for iterating over sequences, repeating actions, and automating repetitive tasks.