# Python keywords Assignment

Python keywords are reserved words that have special meanings in the language's syntax and cannot be used for any other purpose, such as naming variables, functions, or classes. They define the structure and flow of a Python program and are essential for writing clear, correct, and efficient code. Each keyword serves a specific function and follows predefined rules.

Here are five examples of Python keywords along with their significance:

**1. if**

- **Significance**: The if keyword is used for conditional statements. It allows the execution of a block of code if a specified condition is true. If the condition is false, the block of code is skipped.
- **Example**:

python

Copy code

```
age = 18

if age >= 18:
    print("You are an adult.")
```

**2. for**

- **Significance**: The for keyword is used to create a loop that iterates over a sequence (like a list, tuple, or string). It repeatedly executes a block of code for each item in the sequence.
- **Example**:

python

Copy code

```
numbers = [1, 2, 3]

for num in numbers:
    print(num)
```

**3. def**

- **Significance**: The def keyword is used to define a function. It allows for the encapsulation of reusable code into a named function that can be called with arguments to perform a task.
- **Example**:

python

Copy code

```
def greet(name):
    print(f"Hello, {name}!")


greet("Alice")
```

**4. return**

- **Significance**: The return keyword is used inside a function to send a result back to the caller. It exits the function and optionally passes a value back.
- **Example**:

python

Copy code

```
def add(a, b):
    return a + b


result = add(3, 5)
print(result)  # Output: 8
```

**5. class**

- **Significance**: The class keyword is used to define a class, which is a blueprint for creating objects. A class encapsulates data (attributes) and functions (methods) that operate on the data.
- **Example**:

python

Copy code

```
class Dog:
    def __init__(self, name):
        self.name = name


    def bark(self):
```

```
    print(f"{self.name} says woof!")


my_dog = Dog("Buddy")
my_dog.bark()
```

These keywords form the foundation of Python's syntax and structure, and mastering them is essential for effective programming in Python.

2_Identifiers are names used to identify variables, functions, classes, modules, etc. Here are the rules for defining identifiers:

- Must start with a letter (a-z, A-Z) or an underscore (_).
- Can be followed by letters, digits (0-9), or underscores.
- Cannot be a Python keyword (e.g., if, else, for).
- Are case-sensitive (myvar is different from MyVar).

Here's an example of valid identifiers: my_variable, _count, item1.

3_ Comments in Python are lines of code that are not executed by the interpreter. They are used to explain what the code does and make it easier to understand.

There are two types of comments in Python:

- **Single-line comments:** Start with a # symbol and continue until the end of the line.
- **Multi-line comments:** Enclosed in triple quotes ("""Comment goes here""") or ('''Comment goes here''').

```
# This is a single-line comment


"""
This is a
multi-line comment
"""
```

4_ Why is proper indentation important in Python?

Indentation is crucial in Python because it defines the structure and grouping of code blocks. Unlike many other languages that use curly braces {}, Python relies on indentation to indicate which statements belong to a particular block.

Inconsistent indentation can lead to syntax errors or unexpected behavior.

5_) What happens if indentation is incorrect in Python?

If indentation is incorrect, the code will raise an Indentation Error. This means the interpreter cannot understand the structure of your code due to inconsistent or incorrect indentation.

To avoid this, ensure all statements within a block have the same indentation level, preferably using spaces (four spaces is a common convention) consistently.

6)) Differentiate between expression and statement in Python with examples.?

| expression | statement |
|---|---|
| A piece of code that evaluates to a value.<br>• Example: 1 + 2 (evaluates to 3), x * 5 (evaluates to the result of x multiplied by 5) | • An instruction that performs an action. It doesn't necessarily have a value.<br>o Example: print("Hello"), if x > 5: (conditional statement) |