Python Keywords, Identifiers, Comments, Indentation and Statements

September 1, 2024

Q1- Explain the significance of Python keywords and provide examples of five keywords

[10]:
```
#'if' 'elif' 'else'
x = 15
if x > 10:
    print("x is greater than 15")
elif x == 10:
    print("x is equal to 10")
else:
    print("x is less than 10")

#for
fruits = ["Apple","Mango","Banana","Orange"]
for fruit in fruits:
    print(fruit)

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

#break
for i in range(10):
    if i == 5:
        break
    print(i)

#pass
x = 10
if x > 5:
    pass
else:
    print("x is less than or equal to 5")
```

```
x is greater than 15
Apple
```

```
Mango
Banana
Orange
0
1
2
3
4
0
1
2
3
4
```

Q2- Describe the rules for defining identifiers in Python and provide an example.

Ans. In Python, identifiers are names used to identify variables, functions, classes, modules, or other objects. Here are the rules for defining identifiers in Python:

1. Valid Characters: Identifiers can include letters (both uppercase and lowercase), digits (0-9), and underscore (_).
2. Must Start with Letter or Underscore: An identifier must start with a letter (a-z, A-Z) or an underscore (_). It cannot start with a digit.
3. Case-sensitive: Python is case-sensitive, so uppercase and lowercase letters are considered different.
4. Cannot be a Keyword: Identifiers cannot be a reserved word or keyword used in Python. For example, you cannot name a variable "if" or "def" because these are keywords used for control structures.
5. No Special Characters: Special characters like !, @, #, $, %, etc., are not allowed in identifiers.

Here's an example of valid and invalid identifiers:

[1]:
```python
# Valid Identifiers
my_variable = 42
myVariable = "Hello"
_my_variable = True
MY_VARIABLE = [1, 2, 3]
this_is_a_long_identifier = "Long identifiers are fine"

# Invalid Identifiers
2nd_variable = 10  # Cannot start with a digit
$special = "Invalid"  # Cannot have special characters
for = 5  # 'for' is a keyword, so it cannot be used as an identifier
my-variable = 3.14  # Hyphens are not allowed
```

```
  Cell In[1], line 9
    2nd_variable = 10  # Cannot start with a digit
    ^
```

```
SyntaxError: invalid decimal literal
```

Q3- What are comments in Python, and why are they usefull? Provide an example.

Ans. Comments in Python are lines of text in your code that are not executed when the program runs. They are used to make the code more readable and understandable for other developers (including yourself in the future). Comments are essential for documenting code, explaining its purpose, providing context, or temporarily disabling parts of the code during testing or debugging.

Python supports two types of comments:

1. Single-line comments: These start with a hash character # and continue to the end of the line. They are typically used for short explanations or comments on a single line.
2. Multi-line comments: Python doesn't have a built-in syntax for multi-line comments like some other languages do (such as '/* */' in C or Java). However, developers often use triple quotes " " " " " " or ''' ''' to create multi-line string literals, which can serve as multi-line comments because they are not assigned to any variable and are ignored by the interpreter.

Here's an example:

[2]:
```
# This is a single-line comment
print("Hello, World!")  # This comment explains the next line of code

"""
This is a multi-line comment.
It spans across multiple lines.
These lines are not executed by the Python interpreter.
"""
print("This line will be executed")
```

```
Hello, World!
This line will be executed
```

Q4- Why is proper indentation important in Python?

Ans. Proper indentation in Python is important because it:

- Defines the structure and scope of the code.
- Ensures readability and clarity.
- Is required by Python's syntax.
- Helps avoid syntax errors.
- Makes the code easier to understand and maintain.
- Establishes consistency for collaboration.
- Determines code blocks like loops and functions.

Q5- What happens if indentation is incorrect in Python?

Ans. If indentation is incorrect in Python:

- It results in syntax errors.
- The code will not run.
- Python interpreter will raise an 'IndentationError'.
- Incorrectly indented code blocks will not be recognized.

Q6- Differentiate Between expression and statement in Python with examples.

Ans. Expression:

```
Definition: Produces a value.

Purpose: Computation of values.
```

[5]:
```
# Examples of expressions
x = 10
y = 5
result = x + y  # This is an expression that produces a value
```

Statement:

```
Definition: Performs an action or makes something happen.

Purpose: Action or control flow.
```

[6]:
```
# Examples of statements
x = 10  # This is a variable assignment statement
print("Hello, World!")  # This is a print statement
if x > 0:  # This is an if statement
    print("x is positive")
```

```
Hello, World!
x is positive
```

Differences:

```
Result: Expressions produce values, while statements perform actions.

Usage: Expressions are used within statements, but statements cannot be used within expressions

Syntax: Expressions can be part of statements, but statements cannot be part of expressions.
```

[ ]: