Day 2 - Syntax and Rules in Python coding

Premanand S

Assistant Professor
School of Electronics Engineering
Vellore Institute of Technology
Chennai Campus

premanand.s@vit.ac.in

October 21, 2024

Overview of Python Syntax

Python Syntax: A set of rules that define how Python programs are written and interpreted.

- Case sensitive
- Indentation-based structure
- Fewer syntax rules compared to other languages
- Python code is designed to be readable and intuitive

Key Syntax Rules

- **Case Sensitivity**: Python distinguishes between ‘Variable‘ and ‘variable‘.
- **Indentation**: Proper indentation is required for blocks of code.
- **Comments**: Use ‘‘ for single-line and ‘‘‘‘‘ or ‘‘‘‘‘‘‘‘‘ for multi-line comments.
- **Statements**: Instructions that Python can execute, e.g., variable assignment.
- **Line Continuation**: Use \or parentheses to split lines.
- **White Space**: Indentation is crucial for code structure.

Python Identifiers

Identifiers: Names used to identify variables, functions, classes, etc.

- Can contain letters, digits, and underscores, but must start with a letter or an underscore.
- Case-sensitive (‘myVar‘ and ‘myvar‘ are different).
- Cannot use Python keywords (We discussed in Day 1) as identifiers.
- Examples: ‘my_variable‘, ‘_privateVar‘, ‘Class123‘

Indentation Example

Indentation: Python uses indentation to define blocks of code.

Example (Python Code)

```
if True:
    print("This is correctly indented.")
    if False:
        print("This will not print.")
    print("Still inside the outer block.")
print("This is outside the block.")
```

Comments in Python

- **Single-line Comment**: Use the # symbol.
- **Multi-line Comment**: Use triple quotes, ''' or """.

Example (Single-line Comment)

```
# This is a single-line comment
```

Example (Multi-line Comment)

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

Line Continuation Example

Python allows you to split a long line of code using ‘òr parentheses.

Example (Python Code)

```
total = 1 + 2 + 3 + \
        4 + 5
```

Example (Using Parentheses)

```
total = (1 + 2 + 3 +
         4 + 5)
```

Input and Output in Python

Input: Use the ‘input()‘ function to take input from the user.

Output: Use the ‘print()‘ function to display output.

Example (Input and Output Example)

```
name = input("Enter your name: ")
print(f"Hello, {name}!")
```

Variables and Data Types

Variables: Store values in memory. Python does not require you to declare the type.

- Example: Age = 38, name = "premanand"‘, ‘is_student = False‘

Data Types: Python automatically infers the data type.

- `int`, `float`, `complex`, `str`, `bool`, `list`, `dict`, `set`, `tuple`

Control Structures

Conditional Statements: `if`, `elif`, `else`

Example (If-else Statement)

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

Loops in Python

For Loop: Iterates over a sequence (like a list or a range of numbers).

Example (For Loop Example)

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

While Loop

While Loop: Repeats as long as a condition is `True`.

Example (While Loop Example)

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

Function Definitions

Defining Functions: Use the `def` keyword to define a function.

Example (Function Example)

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

greet("Premanand")
```

Common Syntax Errors

- **IndentationError**: Improper indentation.
- **SyntaxError**: Incorrect syntax (e.g., missing colons).
- **NameError**: Undefined variable or object.
- **TypeError**: Incorrect data types used in operations.

More Advanced Syntax Rules

- **List Comprehensions**: A concise way to create lists.
- **Lambda Functions**: Small anonymous functions.
- **Generators**: Use `yield` to return values lazily.
- **Decorators**: A way to modify the behavior of functions or methods.

Error Handling

Try-Except Block: Used to handle exceptions.

Example (Try-Except Example)

```
try:
    x = 1 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
```

Important Symbols in Python

Python Programming Symbols:

- \ - Used as an escape character and line continuation symbol.
- # - Indicates a comment in Python.
- ( ) - Parentheses are used for function calls, grouping expressions, and more.
- [ ] - Square brackets are used for list indexing, slicing, and comprehension.
- { } - Curly braces are used in sets, dictionaries, and format strings.
- : - Used to define function bodies, loops, and conditional statements.
- = - Assignment operator, assigns values to variables.
- == - Comparison operator, checks if two values are equal.
- > / < - Comparison operators for greater than / less than.
- +=, −=, *=, /= - Compound assignment operators.

Note: These symbols are crucial for both beginner and advanced Python learners. As you progress, you will encounter them frequently in various contexts.

mail me: er.anandprem@gmail.com / premanand.s@vit.ac.in
ring me: +91 73586 79961
follow me: Linkedin
author at Analytics Vidhya: premanand17
instagram: premsanand

Learning gives Creativity,
Creativity leads to Thinking,
Thinking provides Knowledge,
and
Knowledge makes you Great
- Dr APJ Abdul Kalam