Python Basic Syntax and Semantics

Python is known for its simplicity and readability. Here are the foundational elements of Python's syntax and semantics that will help you get started with writing Python code.

1. Comments

Comments are used to explain code and are ignored by the interpreter.

- **Single-line comment**:

  ```
  # This is a single-line comment
  ```
- **Multi-line comment**:

  ```
  """
  This is a multi-line comment.
  It can span multiple lines.
  """
  ```

2. Variables and Data Types

Variables store data values, and Python has various built-in data types.

- **Variable assignment**:

  ```
  x = 10  # Integer
  y = 3.14  # Float
  name = "Alice"  # String
  is_active = True  # Boolean
  ```
- **Type checking**:

  ```
  print(type(x))  # <class 'int'>
  print(type(y))  # <class 'float'>
  print(type(name))  # <class 'str'>
  print(type(is_active))  # <class 'bool'>
  ```

3. Operators

Python supports various operators for arithmetic, comparison, logical operations, etc.

- **Arithmetic operators**:

  ```
  a = 10
  b = 5

  print(a + b)  # Addition: 15
  print(a - b)  # Subtraction: 5
  print(a * b)  # Multiplication: 50
  print(a / b)  # Division: 2.0
  ```

```python
print(a % b)  # Modulus: 0
print(a ** b)  # Exponentiation: 100000
print(a // b)  # Floor Division: 2
```

- **Comparison operators**:

  ```python
  print(a == b)  # Equal to: False
  print(a != b)  # Not equal to: True
  print(a > b)  # Greater than: True
  print(a < b)  # Less than: False
  print(a >= b)  # Greater than or equal to: True
  print(a <= b)  # Less than or equal to: False
  ```

- **Logical operators**:

  ```python
  c = True
  d = False

  print(c and d)  # Logical AND: False
  print(c or d)  # Logical OR: True
  print(not c)  # Logical NOT: False
  ```

4. Control Structures

Control structures manage the flow of execution in a program.

- **Conditional statements**:

  ```python
  if a > b:
      print("a is greater than b")
  elif a == b:
      print("a is equal to b")
  else:
      print("a is less than b")
  ```

- **Loops**:

  - **For loop**:

    ```python
    for i in range(5):
        print(i)  # Prints numbers 0 to 4
    ```

  - **While loop**:

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

5. Functions

Functions are reusable blocks of code that perform a specific task.

- **Defining a function**:

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

greet("Alice")  # Output: Hello, Alice!
```

- **Returning values**:

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

  result = add(10, 5)
  print(result)  # Output: 15
  ```

6. Data Structures

Python provides several built-in data structures.

- **Lists**:

  ```
  fruits = ["apple", "banana", "cherry"]
  print(fruits[0])  # Accessing first element: apple
  fruits.append("date")  # Adding an element
  print(fruits)
  ```

- **Tuples**:

  ```
  point = (1, 2)
  print(point[0])  # Accessing first element: 1
  ```

- **Dictionaries**:

  ```
  person = {"name": "Alice", "age": 25}
  print(person["name"])  # Accessing value: Alice
  person["age"] = 26  # Updating value
  print(person)
  ```

- **Sets**:

  ```
  unique_numbers = {1, 2, 3, 3, 2}
  print(unique_numbers)  # Output: {1, 2, 3}
  ```

7. Exception Handling

Exception handling allows you to manage errors gracefully.

- **Try-except block**:

  ```
  try:
      result = 10 / 0
  except ZeroDivisionError:
      print("Cannot divide by zero")
  ```

8. Modules and Packages

Modules and packages help in organizing Python code.

- **Importing a module**:

  ```python
  import math

  print(math.sqrt(16))  # Output: 4.0
  ```

- **Creating a module**:

  - Create a file named `mymodule.py`:

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

  - Import and use the module:

    ```python
    import mymodule

    result = mymodule.add(10, 5)
    print(result)  # Output: 15
    ```

9. File I/O

Python allows for reading from and writing to files.

- **Reading a file**:

  ```python
  with open('file.txt', 'r') as file:
      content = file.read()
      print(content)
  ```

- **Writing to a file**:

  ```python
  with open('file.txt', 'w') as file:
      file.write("Hello, World!")
  ```

These basics provide a solid foundation for writing and understanding Python code. As you progress, you'll encounter more advanced concepts and libraries that extend Python's capabilities.