Python Syntax and Basic Constructs

1. Variables and Data Types

Variables in Python

Variables are containers for storing data values. Python is dynamically typed, meaning you don't need to declare the variable type explicitly.

Variable Naming Rules:

- Must start with a letter (a-z, A-Z) or underscore (_)
- Can contain letters, numbers, and underscores
- Case-sensitive (age, Age, and AGE are different variables)
- Cannot use Python keywords (if, while, for, etc.)
- Use descriptive names (student_name is better than sn)

Examples:

python

```
# Valid variable names
student_name = "John"
_age = 25
total_marks = 450
student1 = "Alice"

# Invalid variable names
# 1student = "Bob"  # Cannot start with a number
# student-name = "Tom"  # Cannot use hyphens
# for = 10  # Cannot use keywords
```

Data Types

Python has several built-in data types:

1. Numeric Types:

a) Integer (int): Whole numbers, positive or negative

python

```
age = 25
temperature = -10
population = 1000000
```

b) Float (float): Decimal numbers

python

```
price = 99.99
pi = 3.14159
temperature = 36.6
```

2. Text Type:

String (str): Sequence of characters enclosed in quotes

python

```
name = "Alice"
message = 'Hello, World!'
multiline = """This is a
multiline string"""
```

3. Boolean Type:

Boolean (bool): Represents True or False

python

```
is_student = True
is_passed = False
```

2. Type Casting

Type casting is the process of converting one data type to another.

Implicit Type Casting

Python automatically converts one data type to another when needed:

python

```
x = 10      # integer
y = 5.5     # float
z = x + y   # z will be 15.5 (float)
print(type(z))  # <class 'float'>
```

Explicit Type Casting

Manual conversion using built-in functions:

1. Converting to Integer:

python

```
# Float to int (decimal part is removed)
x = int(5.9)  # x = 5

# String to int
y = int("100")  # y = 100

# Boolean to int
z = int(True)  # z = 1
```

2. Converting to Float:

python

```
# Int to float
x = float(10)  # x = 10.0

# String to float
y = float("3.14")  # y = 3.14

# Boolean to float
```

```
z = float(False)  # z = 0.0
```

3. Converting to String:

python

```
# Int to string
x = str(100)  # x = "100"


# Float to string
y = str(3.14)  # y = "3.14"


# List to string
z = str([1, 2, 3])  # z = "[1, 2, 3]"
```

4. Converting to Boolean:

python

```
# Any non-zero number is True
x = bool(10)  # True
y = bool(0)   # False


# Empty string is False
a = bool("")  # False
b = bool("Hello")  # True


# Empty list is False
c = bool([])  # False
d = bool([1, 2])  # True
```

4. Input and Output Functions

Output Function: print()

The print() function displays output to the console.

Basic Usage:

python

```
print("Hello, World!")
print(100)
print(3.14)
```

Multiple Arguments:

python

```
name = "Alice"
age = 25
print("Name:", name, "Age:", age)
# Output: Name: Alice Age: 25
```

Input Function: input()

The input() function reads data from the user as a string.

Basic Usage:

python

```
name = input("Enter your name: ")
print("Hello,", name)
```

Reading Different Types:

Since input() always returns a string, you need to convert it:

python

```
# Reading integer
age = int(input("Enter your age: "))
print("You are", age, "years old")

# Reading float
```

```
price = float(input("Enter price: "))

print("Price is:", price)
```

5. Operators in Python

Operators are special symbols that perform operations on variables and values.

Arithmetic Operators

Perform mathematical operations:

```
python

a = 10

b = 3


print(a + b)   # Addition: 13

print(a - b)   # Subtraction: 7

print(a * b)   # Multiplication: 30

print(a / b)   # Division: 3.333...

print(a // b)  # Floor Division: 3

print(a % b)   # Modulus (remainder): 1

print(a ** b)  # Exponentiation: 1000
```

Comparison (Relational) Operators

Compare two values and return True or False:

```
python

x = 10

y = 5


print(x == y)  # Equal to: False

print(x != y)  # Not equal to: True

print(x > y)   # Greater than: True
print(x < y)   # Less than: False
print(x >= y)  # Greater than or equal to: True
print(x <= y)  # Less than or equal to: False
```

Logical Operators

Combine conditional statements:

python

```
x = 10
y = 5
z = 15

# AND - Both conditions must be True
print(x > y and x < z)  # True

# OR - At least one condition must be True
print(x < y or x < z)   # True

# NOT - Reverses the result
print(not(x > y))       # False
```

Assignment Operators

Assign values to variables:

python

```
x = 10          # Simple assignment

x += 5          # x = x + 5 (now x = 15)
x -= 3         # x = x - 3 (now x = 12)
x *= 2          # x = x * 2 (now x = 24)
```

```
x /= 4        # x = x / 4 (now x = 6.0)

x //= 2       # x = x // 2 (now x = 3.0)

x %= 2        # x = x % 2 (now x = 1.0)

x **= 3       # x = x ** 3 (now x = 1.0)
```

Membership Operators

Test if a value is present in a sequence:

python

```
fruits = ["apple", "banana", "cherry"]


print("apple" in fruits)      # True

print("mango" not in fruits)  # True


text = "Hello World"

print("Hello" in text)        # True

print("hello" in text)        # False (case-sensitive)
```

7. Identity Operators

Compare the memory locations of objects:

python

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

y = [1, 2, 3]

z = x


print(x is z)      # True (same object)

print(x is y)      # False (different objects)

print(x == y)      # True (same values)

print(x is not y)  # True
```