# Chapter 1: Variables, Expressions, Arithmetic Operators, and Conditional Operators in Python

## 1.1 Introduction to Variables in Python

A variable in Python is a named location in memory used to store data. Unlike other programming languages, Python does not require explicit declaration of variables. You can assign a value to a variable using the = operator.

### Variable Naming Rules:

* Must start with a letter or underscore (\_)
* Cannot start with a number
* Can contain letters, numbers, and underscores
* Case-sensitive (Age and age are different)
* Cannot use reserved keywords (e.g., print, if, else, etc.)

### Example:

name = "John" # String variable

age = 25 # Integer variable

height = 5.9 # Float variable

is\_student = True # Boolean variable

### Hands-on Exercises:

1. Create variables to store your name, age, and favorite color.
2. Assign integer and floating-point values to two different variables and print them.
3. Swap two variables without using a third variable.
4. Create a variable to store the value of Pi (3.14159) and print it.
5. Assign a boolean value to a variable and print its type using type().
6. Type conversion Needs to be added in notes!

## 1.2 Expressions in Python

An expression is a combination of variables, values, and operators that Python evaluates to produce a result. Expressions can be used in assignments and computations.

### Example:

x = 10

y = 5

result = x + y # Expression

print(result) # Output: 15

### Hands-on Exercises:

1. Create an expression that multiplies two numbers and stores the result in a variable.
2. Combine a string variable with another string using concatenation.
3. Use an expression to calculate the area of a rectangle (length \* width).
4. Evaluate the expression (10 + 5) \* 2 - 3 and print the result.
5. Write an expression that converts temperature from Celsius to Fahrenheit using the formula F = C \* 9/5 + 32.

## 1.3 Arithmetic Operators in Python

Arithmetic operators perform mathematical operations on numeric data types.

### Python Arithmetic Operators:

| Operator | Description | Example |
| --- | --- | --- |
| + | Addition | x + y |
| - | Subtraction | x - y |
| \* | Multiplication | x \* y |
| / | Division (float) | x / y |
| // | Floor Division | x // y |
| % | Modulus (remainder) | x % y |
| \*\* | Exponentiation | x \*\* y |

### Example:

a = 10

b = 3

print(a + b) # Output: 13

print(a - b) # Output: 7

print(a \* b) # Output: 30

print(a / b) # Output: 3.3333

print(a // b) # Output: 3 (floor division)

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

print(a \*\* b) # Output: 1000 (10^3)

### Hands-on Exercises:

1. Compute the sum, difference, and product of two numbers.
2. Perform floor division between 25 and 4, then print the result.
3. Compute the remainder when 50 is divided by 7.
4. Use exponentiation to calculate 2^5.
5. Write an expression that calculates the average of three numbers.

## 1.4 Conditional Operators in Python

Conditional operators (also known as comparison operators) compare two values and return a boolean result (True or False).

### Python Conditional Operators:

| Operator | Description | Example |
| --- | --- | --- |
| == | Equal to | x == y |
| != | Not equal to | x != y |
| > | Greater than | x > y |
| < | Less than | x < y |
| >= | Greater than or equal to | x >= y |
| <= | Less than or equal to | x <= y |

### Example:

x = 10

y = 5

print(x > y) # Output: True

print(x < y) # Output: False

print(x == 10) # Output: True

print(x != 5) # Output: True

### Hands-on Exercises:

1. Compare two numbers using == and != and print the results.
2. Check if a number is greater than 100.
3. Compare the length of two strings.
4. Use >= to check if a person is eligible to vote (age >= 18).
5. Compare the values of three different variables using > and <.

## 1.5 Summary

* Variables store data of different types.
* Expressions combine variables, values, and operators to produce a result.
* Arithmetic operators perform mathematical operations.
* Conditional operators compare values and return True or False.

By practicing these concepts and exercises, you will strengthen your understanding of Python fundamentals.

# **If-Elif-Else, Loops, and Functions in Python**

## **1. If-Elif-Else Statements**

Conditional statements allow a program to make decisions based on different conditions. The if statement is used to execute a block of code if a condition is true. The elif (short for "else if") statement checks additional conditions if the previous ones were false. The else statement executes a block of code when none of the preceding conditions are met.

### **Example 1: Checking Positive or Negative Number**

num = int(input("Enter a number: "))

if num > 0:

print("Positive number")

elif num < 0:

print("Negative number")

else:

print("Zero")

### **Example 2: Voting Eligibility Checker**

age = int(input("Enter your age: "))

if age >= 18:

print("You are eligible to vote.")

else:

print("You are not eligible to vote.")

### **Example 3: Grade Calculation**

score = int(input("Enter your score: "))

if score >= 90:

print("Grade: A")

elif score >= 80:

print("Grade: B")

elif score >= 70:

print("Grade: C")

elif score >= 60:

print("Grade: D")

else:

print("Grade: F")

## **2. Loops**

Loops are used to execute a block of code multiple times. Python provides for and while loops. The for loop is generally used for iterating over sequences, while the while loop executes as long as a condition is true.

### **Example 1: Printing Numbers from 1 to 5 using For Loop**

for i in range(1, 6):

print(i)

### **Example 2: Sum of First N Numbers using While Loop**

n = int(input("Enter a number: "))

sum = 0

counter = 1

while counter <= n:

sum += counter

counter += 1

print("Sum of first", n, "numbers is:", sum)

### **Example 3: Printing Multiplication Table**

num = int(input("Enter a number: "))

for i in range(1, 11):

print(f"{num} x {i} = {num \* i}")

## **3. Functions**

Functions help in organizing code by grouping reusable blocks together. Functions in Python are defined using the def keyword and can accept parameters and return values.

### **Example 1: Factorial Function**

def factorial(n):

if n == 0 or n == 1:

return 1

return n \* factorial(n - 1)

print(factorial(5)) # Output: 120

### **Example 2: Check Prime Number**

def is\_prime(n):

if n < 2:

return False

for i in range(2, int(n \*\* 0.5) + 1):

if n % i == 0:

return False

return True

print(is\_prime(7)) # Output: True

### **Example 3: Reverse a String**

def reverse\_string(s):

return s[::-1]

print(reverse\_string("hello")) # Output: "olleh"

This chapter explains the fundamental concepts of conditional statements, loops, and functions in Python, followed by examples to reinforce learning. Practice these examples and modify them to gain a deeper understanding!