# Lecture 5: Variables in Python 2 (Numeric Variables)

1- **Preliminary Definition**

A numeric variable is a type of variable used to store numerical values such as integers or decimal fractions. In Python, there are multiple types for numeric representation, the most important are:

- **int** for integers (e.g., 5, -3, 1000)
- **float** for decimal numbers (e.g., 3.14, -0.5, 2.0)

**Example:**

```
age = 25       # Integer of type int
price = 19.99   # Decimal number of type float
```

**2- Why Do We Need Numeric Variables?**

Numbers are essential in programming because they are used for:

- Performing arithmetic and logical operations
- Representing ages, prices, quantities, ratios, and more
- Controlling loops and iterations
- Analyzing statistical or scientific data
- Building financial, educational, or game applications that rely on numbers

**3- How Are Numbers Written in Python?**

Numbers are written directly without quotation marks, and they can be:

- Integers: 10, -7, 0
- Decimals: 3.14, -0.01, 2.0

**Example:**

```
x = 10       # Integer
y = 3.5      # Decimal number
z = x + y    # Addition between int and float
```

**Note:** When adding an int and a float, the result will be of type float.

**4- Declaring a Numeric Variable**

Declaring variables in any programming language means informing the interpreter that you will use a certain variable and specifying its type (integer, decimal, string, etc.). This allows the program to allocate appropriate memory space to store the numeric value.

**Example of declaring numeric variables:**

```
count = 10       # Integer of type int
temperature = 36.6 # Decimal number of type float
```

Numbers can be:

- Positive or negative
- Integers or decimals
- Zero is also considered an integer

**Note:** Quotation marks are not used when declaring a numeric variable, because that would convert the value into a string rather than a number.

**5- Important Numeric Functions in Python**

In programming, functions are among the most important tools used by developers to organize code and simplify tasks. Numeric functions in Python are used to perform mathematical operations, conversions, or check numeric properties without rewriting instructions.

Since numbers are used in almost all programming applications, Python provides a rich set of built-in functions to handle them. These functions make it easier to: perform calculations, rounding, type conversions, property checks, and other essential tasks.

**A- Basic Mathematical Functions**

| Function | Purpose | Example |
|---|---|---|
| **abs(x)** | Absolute value of a number | print(abs(-7)) → 7 |
| **round(x)** | Round to the nearest integer | print(round(3.6)) → 4 |
| **pow(x, y)** | Compute (x^y) | print(pow(2, 3)) → 8 |
| **max(a, b, …)** | Find the largest value | print(max(4, 9, 2)) → 9 |
| **min(a, b, …)** | Find the smallest value | print(min(4, 9, 2)) → 2 |

**B- Functions from the math Library**

**Note**: In Python, libraries are collections of ready-made functions and code that help developers perform tasks without writing everything from scratch. To use a library, you must first import it.

```
import math
```

| Function | Purpose | Example |
|---|---|---|
| **math.sqrt(x)** | Square root of a number | print(math.sqrt(16)) → 4.0 |
| **math.floor(x)** | Round down to nearest integer | print(math.floor(3.9)) → 3 |
| **math.ceil(x)** | Round up to nearest integer | print(math.ceil(3.1)) → 4 |
| **math.log(x)** | Natural logarithm of a number | print(math.log(10)) → 2.302... |

**C- Type Conversion Functions**

| Function | Purpose | Example |
|---|---|---|
| **int(x)** | Convert to integer | print(int(3.99)) → 3 |
| **float(x)** | Convert to float | print(float(7)) → 7.0 |
| **complex(x)** | Convert to complex number | print(complex(5)) → (5+0j) |
| **str(x)** | Convert number to string | print(str(123)) → "123" |

**D- Functions for Checking Numeric Properties**

| Function | Purpose | Example |
|---|---|---|
| **isinstance(x, int)** | Check if x is of type int | print(isinstance(5, int)) → True |
| **isinstance(x, float)** | Check if x is of type float | print(isinstance(3.14, float)) → True |

**6- Inputting Numbers by the user in Python**

In programming, we often need to get input from the user. In Python, this is done using the input() function, but the input must be converted into a suitable numeric type such as int or float.

**Simple Example: Entering an Integer**

```
x = int(input("Enter an integer: "))
print("You entered:", x)
```

- The input() function receives what the user types as a string.
- The int() function converts the string into an integer.
- If the user enters a value that cannot be converted, an error will occur.

**Another Example: Entering a Float**

```
y = float(input("Enter a decimal number: "))
print("The entered decimal number is:", y)
```

- The float() function converts the string into a floating-point number (with a decimal point).

**Exercise**

Write a program that takes two numbers from the user (one integer and one float), then prints their sum.

**Solution: Program that adds an integer and a float from the user**

```
# Get an integer from the user
x = int(input("Enter an integer: "))

# Get a float from the user
y = float(input("Enter a decimal number: "))

# Calculate the sum
sum = x + y

# Print the result
print("The sum of the two numbers is:", sum)
```

**Notes**

- int() was used to convert the first input into an integer.
- float() was used to convert the second input into a float.
- The variable sum stores the result of the addition.
- The program prints the result using print().

**7. Basic Arithmetic Operations in Python**

| Operation | Symbol | Example | Result |
|---|---|---|---|
| **Addition** | + | 5 + 3 | 8 |
| **Subtraction** | - | 5 - 3 | 2 |
| **Multiplication** | * | 5 * 3 | 15 |
| **Division** | / | 5 / 2 | 2.5 |
| **Modulus (remainder)** | % | 5 % 2 | 1 |
| **Floor Division** | // | 5 // 2 | 2 |
| **Exponentiation** | ** | 2 ** 3 | 8 |

**Notes**

- Division / always returns a float.
- Floor division // returns an integer without the decimal part.
- Modulus % is very useful for determining whether numbers are even or odd.

**Applied Example**

```
a = 10
b = 3

print("Addition:", a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
print("Division:", a / b)
print("Floor Division:", a // b)
print("Modulus:", a % b)
print("Exponentiation:", a ** b)
```

**Exercise**

Write a program that takes two numbers from the user, then prints the results of the basic arithmetic operations between them.

**Model Solution: Arithmetic operations between two numbers**

```
# Get the first number from the user and convert it to float
a = float(input("Enter the first number: "))

# Get the second number from the user and convert it to float
b = float(input("Enter the second number: "))

# Perform the basic arithmetic operations
print("Addition:       ", a + b)
print("Subtraction:    ", a - b)
print("Multiplication:  ", a * b)
print("Division:       ", a / b)
print("Modulus:        ", a % b)
print("Floor Division:  ", a // b)
print("Exponentiation:  ", a ** b)
```

**Notes**

- `float()` is used to ensure the program accepts both integers and decimal numbers.
- The operations include all the basic symbols: +, -, *, /, %, //, **.
- The program displays the results in a clear, aligned format to make them easy to read.