On the Way to Python: A Fun STEM Course for Future Coders

By : Larbi OUIYZME

What is STEM ?

STEM is an acronym that stands for Science, Technology, Engineering, and Mathematics. STEM courses are designed to teach and integrate these disciplines in an interdisciplinary manner, encouraging students to develop skills in these crucial areas. The introduction to Python in a STEM context aims to familiarize students with computer programming while emphasizing these fundamental disciplines. Additionally, this course will cover mathematical concepts, robot programming, as well as game development.

What is Python ?

Python is a fun and powerful programming language that we will explore together. Today, we will discover some basic concepts to get you started. Python allows you to give instructions to the computer to perform various tasks. It is used in many fields such as software development, data science, artificial intelligence, robotics, and even game creation.

Python Syntax

Syntax is how instructions are written in Python. The Python language is designed to be easy to read and write. In Python, variable naming follows certain rules and is case-sensitive.

Here are some principles to follow :

1. Variable Naming:
   - Variable names must start with a letter (a-z, A-Z) or an underscore _.
   - Subsequent characters in the variable name can be letters, numbers (0-9), or underscores.
   - Avoid using Python reserved words as variable names (e.g., if, else, for, etc.).

2. Case Sensitivity:
   - Python is case-sensitive, meaning that variables maVariable and MAvariable are considered two distinct variables.
   - Be consistent in the use of case to ensure code clarity and consistency.

Example of good naming practices :

```
# Meaningful and explicit names
user_name = "JohnDoe"
user_age = 25

# Using Snake Case convention for compound names
my_variable = 10
total_elements = 50

# Avoid unclear or abbreviated names
x = 5  # Avoid such names, except in very limited contexts
```

Python Comments

Comments are annotations in the source code that are not executed during program execution. In Python, comments are created using the # character.

Example:

```
# This is a single-line comment

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

Displaying Messages with Print

In Python, we can ask the computer to display messages on the screen. Let's use the print command to say something to our program.

```
print("Hello, friends!")
```

By executing this code, you will see the phrase "Hello, friends!" displayed on the screen.

Asking the User with Input

Now, let's talk about communication. How do we ask something from the user? Let's use input to get a response from the user.

```
name = input("What is your name? ")
print("Hello", name, "!")
```

Here, the program will ask for your name, and then it will display it with a welcoming "Hello."

Note:

In Python, when you use the input() function, it automatically treats the input as a string. To convert this input into an integer (int) or decimal number (float), you can use the int() and float() functions.

```
# Ask the user to enter a decimal number
radius = float(input("Enter a decimal number: "))
# Ask the user to enter an integer
candies = float(input("Enter an integer: "))
# Display the result
print("The decimal number is:", radius)
print("The integer number is:", candies)
```

Discovering Data Types with Type

Python can handle different data types, such as numbers or words. To find out what type of data is, we use the type function.

```
age = 10
name = "Alice"
print(type(age))  # Displays <class 'int'>
print(type(name))  # Displays <class 'str'>
```

By using type, we can understand the data type we are working with.

Measuring Length with Len

Sometimes, we want to know how many letters or digits are in a variable or elsewhere. len helps us measure length.

Here, the program will say how many letters are in the word "Python."

```
word = "Python"
length = len(word)
print("The word", word, "has", length, "letters.")
```

Variables

A variable is a space in the computer's memory where data can be stored. You can give any name to a variable. For example:

```
# Declare a variable
name = "Python"
```

Types of Variables in Python

In Python, there are several types of variables to store different kinds of data. Here are some of the most commonly used variable types:

Integer (int) : Stores whole numbers, for example, 5, -10, 100.

```
age = 10
```

Float (float) : Stores decimal numbers, for example, 3.14, -0.5.

```
pi = 3.14
```

String (str) : Stores text, for example, "Python," 'Hello.'

```
first_name = "Alice"
```

Boolean (bool) : Stores either True or False.

```
    is_true = True
```

Arithmetic Operations

Arithmetic operations are used to perform mathematical calculations. Here are some basic arithmetic operations:

Addition (+) : Adds two numbers.

```
sum_result = 5 + 3
```

Subtraction (-) : Subtracts one number from another.

```
difference = 7 - 2
```

Multiplication (*) : Multiplies two numbers.

```
product = 4 * 6
```

Division (/) : Divides one number by another.

```
quotient = 10 / 2
```

Integer Division (//) : Divides one number by another, and the result is an integer.

```
quotient = 10 // 3  # Result is 3
```

Modulo (%) : Gives the remainder of the division.

```
remainder = 15 % 4
```

Logical and Comparison Operations

Logical and comparison operations are used to evaluate logical conditions. Here are some of them:

Comparison Operations:

Equality (==) : Checks if two values are equal.
result = (5 == 5) # True

Inequality (!=) : Checks if two values are not equal.
result = (5 != 3) # True

For Loop

The for loop allows you to repeat a series of instructions a certain number of times. For example:

```
# Display numbers from 1 to 5
for i in range(1, 6):
   print(i)
```

While Loop

The while loop allows you to repeat instructions as long as a condition is true. For example:

```
# Display squares of numbers from 1 to 5
number = 1
while number <= 5:
   print(number ** 2)
   number += 1
```

If-Else Conditional Structure

The if conditional structure allows you to execute instructions only if a condition is true. The else structure allows you to execute alternative instructions if the condition is not true. For example:

```
# Check if a number is even or odd
number = int(input("Enter a number: "))
if number % 2 == 0:
   print("The number is even.")
else:
   print("The number is odd."
```

Python and Mathematics Activities

Calculate the area of a circle

```
# Program to calculate the area of a circle
# Ask the user to enter the radius of the circle
radius = float(input("Enter the radius of the circle: "))
# Calculate the area of the circle using the formula: pi * r^2
area = 3.14 * (radius ** 2)
# Display the result
print("The area of the circle is:", area)
```

Find the maximum of two numbers

```
# Program to find the maximum of two numbers
# Ask the user to enter two numbers
number1 = float(input("Enter the first number: "))
number2 = float(input("Enter the second number: "))
# Find the maximum of the two numbers
maximum = max(number1, number2)
# Display the result
print("The maximum of the two numbers is:", maximum)
```

Check if a number is even

```
# Program to check if a number is even
# Ask the user to enter a number
number = int(input("Enter a number: "))
# Check if the number is even using the modulo operator (%)
if number % 2 == 0:
   print("The number is even.")
else:
   print("The number is odd.")
```

Addition of two numbers

```
# Program to add two numbers
# Ask the user to enter two numbers
number1 = float(input("Enter the first number: "))
number2 = float(input("Enter the second number: "))
# Add the two numbers
result = number1 + number2
# Display the result
print("The sum of the two numbers is:", result)
```

Calculate the average of three numbers

```
# Program to calculate the average of three numbers
# Ask the user to enter three numbers
number1 = float(input("Enter the first number: "))
number2 = float(input("Enter the second number: "))
number3 = float(input("Enter the third number: "))
# Calculate the average of the three numbers
average = (number1 + number2 + number3) / 3
# Display the result
print("The average of the three numbers is:", average)
```

Check if a number is positive, negative, or zero

```
# Program to check if a number is positive, negative, or zero
# Ask the user to enter a number
number = float(input("Enter a number: "))
# Check the sign of the number
if number > 0:
   print("The number is positive.")
elif number < 0:
   print("The number is negative.")
else:
   print("The number is zero.")
```

Condensed form of writing for performing an arithmetic operation and updating the variable in a single step

The operators +=, -=, *=, and /= offer a concise and efficient way to perform an arithmetic operation and update a variable in a single step. These operators are commonly used in programming to simplify code and improve readability.

Here's a brief introduction with examples for each of these operators:

 # += (Addition and assignment) : The += operator adds the value to the right of the operator to the variable on the left, then assigns the result to that variable.

```
x = 5
x += 3  # Equivalent to x = x + 3
print(x)  # Displays 8
```

 # -= (Subtraction and assignment) : The -= operator subtracts the value to the right from the variable on the left, then assigns the result to the variable.

```
y = 10
y -= 4  # Equivalent to y = y - 4
print(y)  # Displays 6
```

 # *= (Multiplication and assignment) : The *= operator multiplies the variable on the left by the value on the right, then assigns the product to the variable.

```
z = 3
z *= 2  # Equivalent to z = z * 2
print(z)  # Displays 6
```

 # /= (Division and assignment) : The /= operator divides the variable on the left by the value on the right, then assigns the quotient to the variable.

```
w = 8
w /= 4  # Equivalent to w = w / 4
print(w)  # Displays 2.0
```

Calculate the factorial of a number

```
# Program to calculate the factorial of a number
# Ask the user to enter a positive integer
n = int(input("Enter a positive integer: "))
# Initialize the result to 1
result = 1
# Calculate the factorial using a loop
for i in range(1, n + 1):
    result *= i
# Display the result
print(f"The factorial of {n} is:", result)
```