Getting Started in Python

What is a Program?

A computer program is a set of instructions that causes a computer to perform some kind of action.
Like humans, computer use multiple languages to communicate – in this case programming languages. A programming language is a particular way to talk to a computer – a way to use instructions that both humans and the computer can understand.

Why Learn Programming?

- Helps improve your creativity, reasoning and problem solving
- Create something from nothing – give instructions to the computer to run that can perform a task or solve a problem
- Fun and challenging!

What is Python?

Python is a general purpose programming language created in the late 1980s, and named after **Monty Python (**a British TV comedy show), that's used by thousands of people to do things like web development, testing microchips at Intel, to powering Instagram, to building video games with the PyGame library.

Why Python?

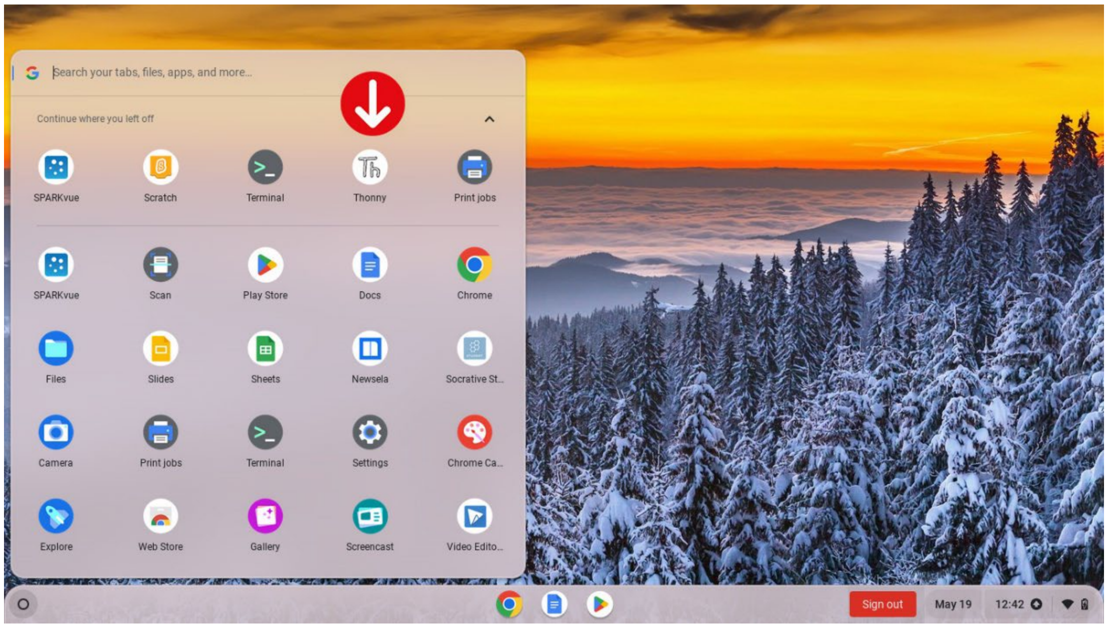
- Easy to learn and read
- Great for beginning programmers
- Interactive – can see results immediately
- Libraries and modules – reduces the amount of coding you have to do
- Great for graphics and animation, for example, creating games
- It's free!

In Chromebooks, the Python programming language runs on the Linux operating system.

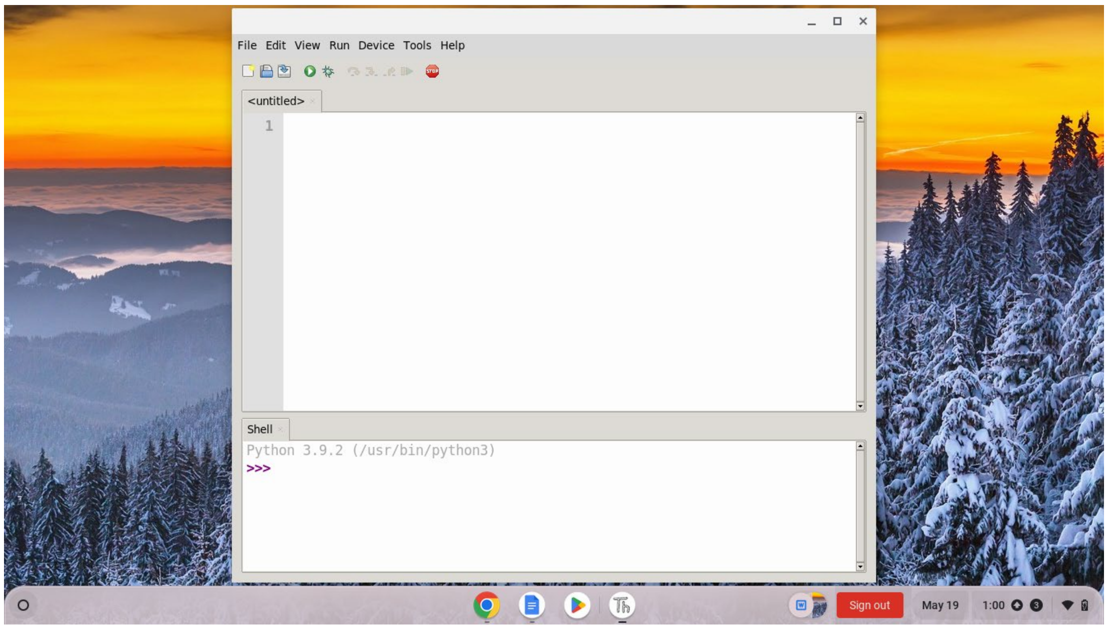
The Thonny Editor

Thonny is a free beginner-friendly Python Integrated Development Environment (IDE) editor. Think of Thonny as the workroom in which you will create amazing Python programs. Your workroom contains a toolbox containing many tools that will enable to you create and run Python programs.

To open the Thonny editor, go to applications and click on the "Th" (Thonny) icon.



When you open the Thonny application opens you should see a window with several icons across the top, and two white areas.



```
Python 3.9.2 (/usr/bin/python3)
>>>
```

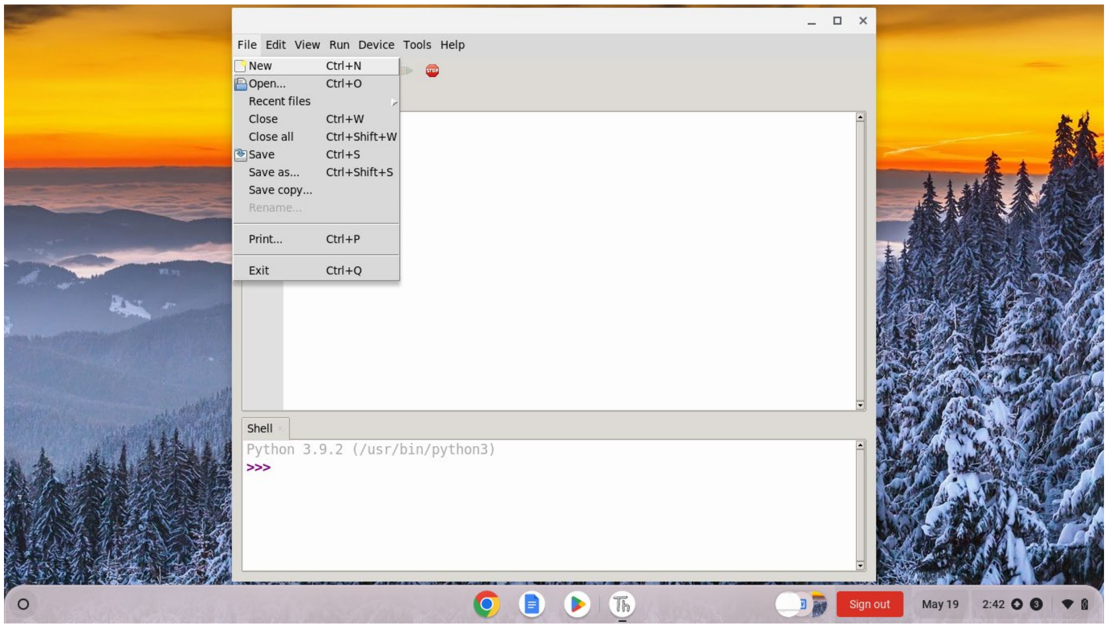
Across the top you'll see several icons. Let's explore what each of them does. You'll see an image of the icons below, with a letter above each one. We will use these letters to talk about each of the icons:

- The paper icon allows you to create a new file. Typically in Python you want to separate your programs into separate files. You'll use this button later in the tutorial to create your first program in Thonny!
- The open folder icon allows you to open a file that already exists on your computer. This might be useful if you come back to a program that you worked on previously.
- The floppy disk icon allows you to save your code. Press this early and often. You'll use this later to save your first Thonny Python program.
- The play icon allows you to run your code. Remember that the code you write is meant to be executed. Running your code means you're telling Python, "Do what I told you to do!" (In other words, read through my code and execute what I wrote.)

You are now ready to write and run your first Python program!

Your First Python Program

Select the “File” option on the top menu and choose “New.”

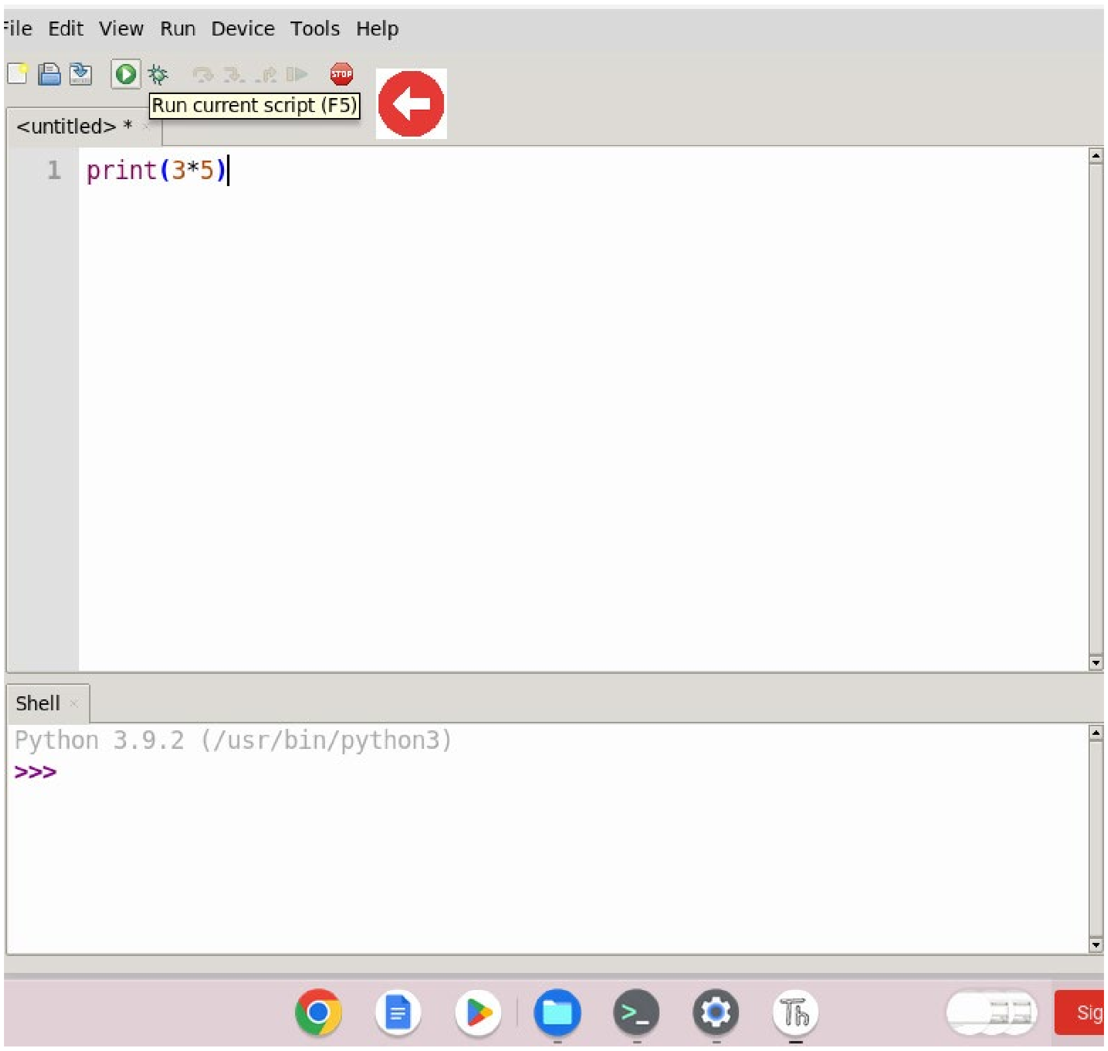


```
Python 3.9.2 (/usr/bin/python3)
>>>
```

Type the following code:

```
print (3 * 5)
```

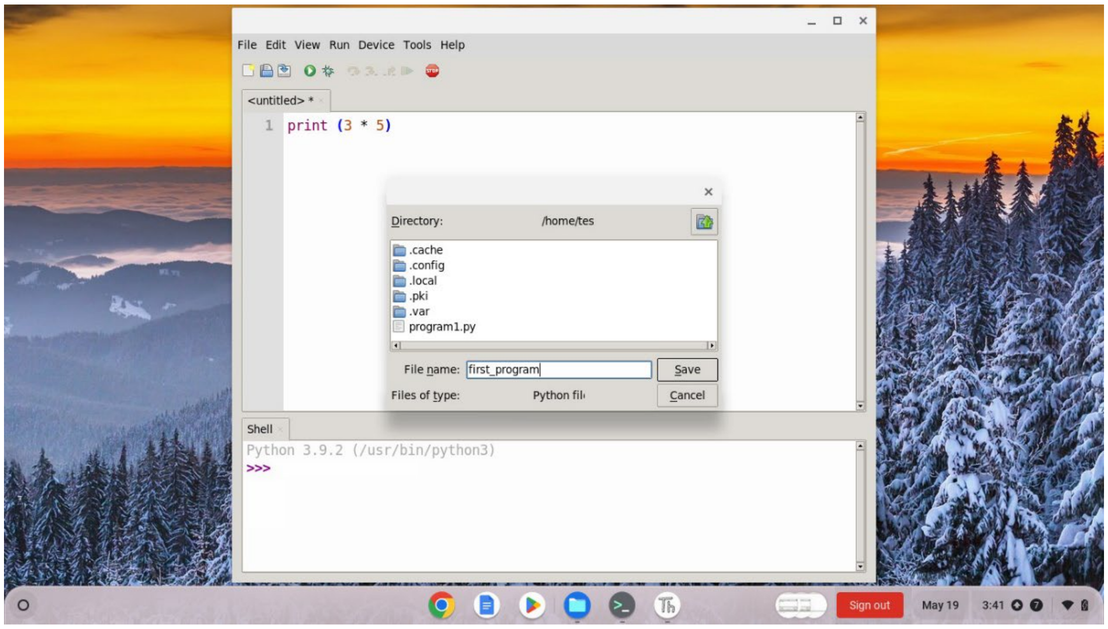
Now click on the green “Run” button on the toolbar.



```
print(3*5)
```

```
Python 3.9.2 (/usr/bin/python3)
>>>
```

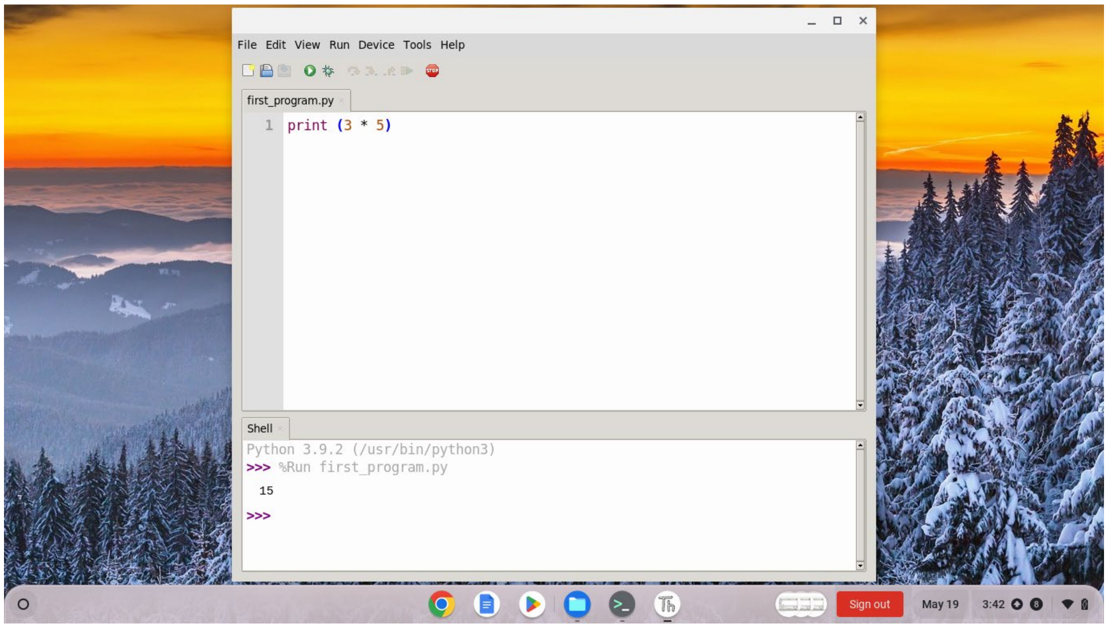
The Thonny editor will then prompt you to save your file. Give your file a name and click "Save."



```
print (3 * 5)
```

```
Python 3.9.2 (/usr/bin/python3)
>>>
```

Your program will now execute and display the output in the shell (bottom window pane).



```
print (3 * 5)
```

```
Python 3.9.2 (/usr/bin/python3)
>>> %Run first_program.py
 15
>>>
```

Print() Function

Syntax

```
print(value1, value2, value3, ..., sep=' ', end='\n')
```

The print function is used to output values, messages or the results of arithmetic or logical expressions to the screen. It shows the results of your program. Whatever you print using the print statement must be within the parenthesis, i.e., ().

1. The following line of code will output the number 3.

```
print(3)
```

2. The following line of code will output the sentence, “Python is fun!”.

```
print("Python is fun!")
```

Operators

Arithmetic Operators

Symbol	Operation
()	Parentheses
**	Exponent
*	Multiplication
/	Division
%	Modulus – returns the remainder of the first number divided by the second number
+	Addition
–	Subtraction

We use an asterisk ,* , for multiplication and forward slash, /, for division. There is no ÷ symbol on the keyboard.

Operator Precedence

Python follows **PEMDAS** which stands
for **P**arentheses **E**xponents **M**ultiplication **D**ivision **A**ddition **S**ubtraction.

Examples

1. The following line of code will output the value 11.

```
print(10-5+6)
```

2. The following line of code will output the value 305.

```
print(5+30*10)
```

In the above example, multiplication takes precedence over addition. So the program will multiply 30 by 10 and then add 5 to the result, hence the ouptut, 305. If you had wanted to do the addition first, you would need to enclose that part of the expression within parentheses as follows:

```
print((5+30)*10)
```

The above expression would yield 350 as the answer.

Parentheses can be nested (one inside the other) as follows:

```
print(((5+30) * 10) / 5)
```

Python evaluates what's in the innermost parentheses first, followed by the outer ones and finally executes the division operator, giving the result of 70. If we had not used parentheses, the results would have been slightly different:

```
print(5+30*10/5)
```

The above case would have given us the result of 65.

Data Types

There are three datatypes in Python, **numeric**, **string** and **Boolean**.

Numeric Datatypes

There are two types of numeric data in Python:

- Integers (whole numbers that do not have a decimal point)
- Floating-point numbers (numbers with a decimal point or fraction)

String Datatype

The string or text datatype is a number of characters enclosed in single or double quotes. Some examples are:

```
'this is some text'

"This is known as a string"

'345'

"?#$%^"

" " (one or more blank spaces are also considered string data)
```

Special String Characters

The ‘\’, ‘\t’ , ‘\n’ and ‘+’ characters are special operators with strings.

When printing string data, the string data must be enclosed using matching quotation marks, i.e., either single or double quotation marks.

The following example will output an error as the quotation marks enclosing the string, ‘Hello World” do not match on both sides.

```
print('Hello World")

>>>
SyntaxError: EOL while scanning string literal
```

If you need to add apostrophes (or single quotation marks) to a string value, you either need to use double quotes to enclose the string value or use the backslash character, “\” just before the apostrophe. This will force Python to ignore the apostrophe as denoting the end of the string value.

```
print("The language, 'Python' is named after Monty Python, not the snake.")

>>>
The language, 'Python' is named after Monty Python, not the snake.
```

In the above example, we used double quotes to denote the beginning and end of the string, so there is no problem with the apostrophe (the single quotation mark) inside the string and we get the correct output.

However, if you had used single quotes around the string text, you would get the following error as Python thinks you have reached the end of the string when it encounters the single quotation mark in front of the word, “Python.”

```
print('The language, 'Python' is named after Monty Python, not the snake.')

>>>
SyntaxError: EOL while scanning string literal
```

In order to avoid this error, use the backslash character before the apostrophe. The backslash tells Python to ignore the apostrophe after it as denoting the end of the string value. You should then get the correct output.

```
print('The language, \'Python\' is named after Monty Python, not the snake.')

>>>
The language, 'Python' is named after Monty Python, not the snake.
```

The “\t” or ‘\t’ and “\n” or ‘\n’ are special characters that are used with strings.

- “\t” or ‘\t’ inserts a tab space
- “\n” or ‘\n’ adds a new line

Examples

The following example inserts a tab space between in the string, “HelloWorld”.

```
code>print("Hello\tWorld")

>>>

Hello    World
```

In this example, tab spaces are inserted between the words, “Python Programming Language”.

```
print("Python\tProgramming\tLanguage")

>>>

Python    Programming    Language
```

The “\n” character adds a new line after the words, “Languages”, “Python”, “Java” and “C”.

```
print("Languages\nPython\nJava\C").

>>>

Languages

Python

Java

C
```

The Concatenation Operator

The concatenation or “+” operator can be used to join two strings together.

```
print("Hello "+"World")
```
The above result will be: Hello World.

The Multiplication Operator

The multiplication or “*” operator is used to repeat a character or string by the number of times it is multiplied with.

```
print (10 * ’a’)
```
The above result will be: aaaaaaaaaa.

Titlecase, Lowercase, Uppercase Functions

- upper() – outputs the string in all capital letters
- lower() – outputs the string in all lowercase letters
- title() – outputs the string with the first letter in each word capitalized

The example below shows what each of these functions do:

```
print(upper("Juan Alvarez"))

JUAN ALVAREZ

print(lower("Juan Alvarez"))

juan alvarez

print(title("Juan Alvarez"))

Juan Alvarez
```
The str() function converts a numeric value to a string value. It is only used with numeric values or variables such as integers or floats.

```
print(“Happy “ + 12 + ”th Birthday!”)

print(message) Traceback (most recent call last):

File "", line 1, in

    print("Happy "+age+"th Birthday!")

TypeError: must be str, not int
```
You cannot concatenate a number with a string. You will need to convert the number to a string first.

```
print("Happy "+str(12)+"th Birthday!")

>>>
```

```
Happy 12th Birthday!
```

Boolean Datatype

Boolean is a special data type that can store either of two values, true or false but not both. Some examples are:

```
likesBaseball = True

is_hungry = False
```

Examples

The following table lists different types of data and what their data type ought to be:

Data	Data type
-4.3	float (the number has a decimal point, so it is a float)
3454	integer
True	boolean
“friend”	string
False	boolean
‘True’	string (because of the quotes around ‘True’)
0.000	float
-3	Integer

Variables

A variable is a place to store information such as string, numeric and logical values. It is like a label for something.

There are three types of variables in Python, **numeric variables**, **string variables** and **logical (Boolean) variables**. Numeric variables store numeric data or values, i.e., numbers and the results of arithmetic expressions. String variables store string data like letters, words, sentences, paragraphs, or special characters. Logical variables store the values True or False.

- Variable names can be made up of letters, numbers and the underscore _ character, but they can't start with a number. For example, you can call a variable message_1, but not 1_message.
- A variable name also cannot have a space in it. Use the underscore character to separate words, e.g., hours_worked.
- Variable names should be short, but meaningful i.e., tell you what you are using it for. For example, name is better than n, student_name is better than s_n and name_length is better than length_of_persons_name.

Examples

1. In the following example, the variable number is assigned the value 100. The print(number) command causes the value in number to be printed out and 100 will be output.

```
number = 100

print(number)

>>>

100
```

2. In this example, the value in number1 is assigned to number2. Since the value in number1 is 100, the print(number2) command will result in the value in number2 is displayed, which would be 200.

```
number1 = 200

number2 = number1

print(number2)

>>>

200
```

3. When you assign a value to a variable, you need to ensure that you spell the name of the variable correctly in every instance where it is used. Not doing so will result in an error.

```
spam = 5

print(spma)
```

```
>>>

Traceback (most recent call last):

  File "", line 2, in

    print(spma)

NameError: name 'spma' is not defined
```

The Input() Function

Syntax

```
variable = input(prompt)
```

The **input()** function pauses your program and waits for you to enter a value. Once you enter a value, Python stores it in a variable.

Numbers that are entered using the input() function are stored as strings by default. So any variable that stores the number entered into it using input() will also be a string variable. In order to use the variable in an arithmetic expression, you will need to first convert the string variable into a numeric variable by using either the **int()** or **float()** function (depending on whether you entered an integer or a decimal number.

The following example illustrates what happens when you input a number and try to use it in an arithmetic expression without first converting it to a numeric value.

```
age = input("Please enter your age")

print(age+5)

>>>

Traceback (most recent call last):

  File "", line 1, in

    print(age+5)

TypeError: must be str, not int
```

You can solve this problem by first converting age to an integer variable before adding 5 to it.

```
age = int(age)

print(age+5)
```

```
name = input(“Please enter your name: “)

print(name)

print(“Hello, “ + name + “!”)
```

The above program accepts a name from the user. The name entered by the user is stored in the variable, name. The next statement will print “Hello” followed by the name you entered and the exclamation point, i.e., “!”.

In the next example, the input() function is used to prompt the user to enter two numbers. The float() function is used to convert the user input into floating-point numbers, which allows for decimal values.

The program then performs addition, subtraction, multiplication and division (quotient) using the entered numbers and the results are displayed using the print() function.

```
num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

sum = num1 + num2

difference = num1 - num2

product = num1 * num2

quotient = num1 / num2

print("Sum:", sum)

print("Difference:", difference)

print("Product:", product)

print("Quotient:", quotient)
```