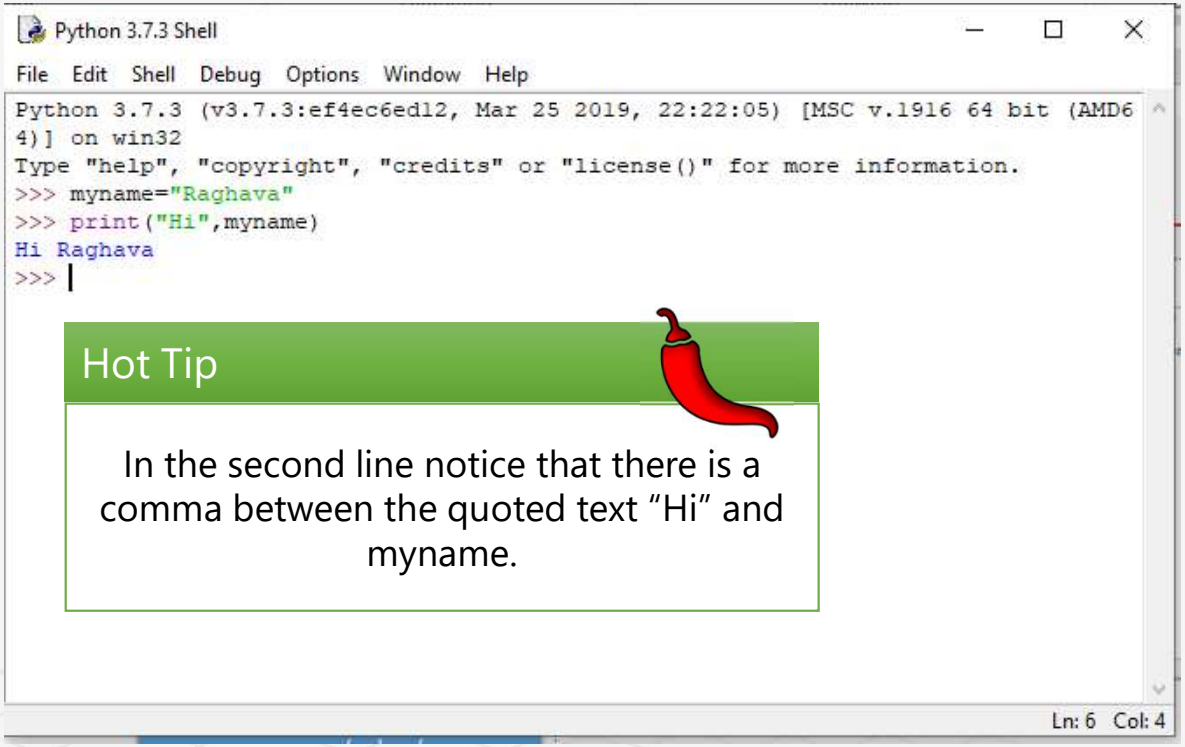
Day – 2: Data Types, Variables and Operators and Inputs

JUST LEARN

Variable

- ✓ A variable is like a container in which data value can be stored inside the computer's memory.
- ✓ The data value can be a number, text or lists of numbers and text
- ✓ The assignment operator (=) is used to assign the data value into variable.
- ✓ The syntax is:

<var> = <var/value/exp>

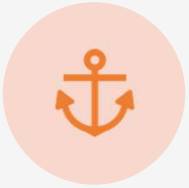


Hot Tip

In the second line notice that there is a comma between the quoted text "Hi" and myname.

JUST LEARN

Does and Don’ts For Naming Variables



The variable name must always start with an alphabet or underscore(_)

The rest of the characters could be alphabet, numbers, or the underscores.

The variable names are case sensitive

They must be unique

Never use special symbols like !,@,#,$,%,etc., within the variable name.

Create a name that make sense.

JUST LEARN

RrT2

Reading input

- ✓ Developers often have a requirement to get data from the user.
- ✓ We use the input() function to read data from user. The syntax is:

```
input(<prompt>)
```

```
#prg2.py
#Read your name and greet
name=input("Enter your name: ")
print("Hi, ", name)
```

```
>>>
==============RESTART: prg2.py =============
Enter your name: Raghava
Hi,  Raghava
>>>
```

Hot Tip

The prompt in input function is string. So, must be enclosed in single or double quotes.

JUST LEARN

RrT2 Add Lab 2-1

Raghavendra rao Tholeti, 5/8/2020

Data Types in Python

- ✓ The data type of an object determines what values it and what operations can be performed on it.
- ✓ Python is strongly and dynamically typed language.
- ✓ Strong Typing
  - Obviously, Python isn't performing static type checking, but it does prevent mixing operations between mismatched types.
  - Explicit conversions are required in order to mix types.
  - Example: 2 + "four" ← not going to fly
- ✓ Dynamic Typing
  - All type checking is done at runtime.
  - No need to declare a variable or give it a type before use.

Let's start by looking at Pythons built-in data types

Strings

- ✓ Created by simply enclosing characters in either single- or double-quotes.
- ✓ It's enough to simply assign the string to a variable.
- ✓ Strings are immutable.
- ✓ Example:

```
mystring = "Hi, I'm a string!"
```

- ✓ Python supports several escape sequences such as '\t', '\n', etc.
- ✓ Placing 'r' before a string will yield its raw value.
- ✓ Two string literals beside one another are automatically concatenated together.

```
#prg3.py
Print("\tHello,\n")
print(r"\tWorld!\n")
print("Python is " "so cool." )
```

```
>>>
======== RESTART: prg3.py =======
        Hello,

\tWorld!\n
Python is so cool.
>>>
```

JUST LEARN

Numeric Types

- ✓ The two primary types of numbers in Python are called Integers and floats
- ✓ Integers: The whole numbers including negatives (like 2, -4, 0)
- ✓ Floats: Real numbers or numbers with decimals (like 2.9, 4.5, 3.14, -0.0099)
- ✓ Another numeric type is complex used to represent complex numbers. The complex numbers are written in in the form of x+yj, where x is the real part and y is the imaginary part.
- ✓ These types are defined as int, float and complex classes in Python.
- ✓ We can use the **type()** function to know which class a variable or value belong to.

```
>>> a = 1234567890123456789
>>> a
1234567890123456789
>>> b = 0.1234567890123456789
>>> b
0.12345678901234568
>>> c = 1+2j
>>> c
(1+2j)
>>> a, "is of type", type(a)
(1234567890123456789, 'is of type',
<class 'int'>)
>>> b, "is of type", type(b)
(0.12345678901234568, 'is of type',
<class 'float'>)
>>> c, "is of type", type(c)
((1+2j), 'is of type', <class
'complex'>)
```

Hot Tip

Integer can be of any length; it is only limited by memory available.
A floating-point number is accurate up to 15 decimal places

JUST LEARN

Arithmetic Operators

- ✓ The arithmetic operators are special symbols that carry out an arithmetic computation

Operator	Operation	Example	Result
+	Addition	5+2	7
-	Subtraction	5-2	3
*	Multiplication	5*2	10
/	Division	5/2	2.5
%	Modulus	5%2	1
//	Floor Division	5//2	2
**	Exponent	5**2	25

```
#prg4.py : Arithmetic Operations
x = 7
y = 2

print('x + y =',x+y)
print('x - y =',x-y)
print('x * y =',x*y)
print('x / y =',x/y)
print('x // y =',x//y)
print('x ** y =',x**y)
```

```
>>>
======== RESTART:prg4.py ========
x + y = 9
x - y = 5
x * y = 14
x / y = 3.5
x // y = 3
x ** y = 49
>>>
```

Programming Sample with Arithmetic Operators: A Burger Calculator

- ✓ Let's try a program to calculate the total cost of a simple burger order, including GST (Goods and Services Tax). Let's assume we are ordering one or more burgers of same cost and the tax percentage is 18. This program can be modeled in words as follows:
  - Ask how many burger they want
  - Ask the cost of each burger
  - Calculate the total cost of the burgers as our subtotal
  - Calculate the GST owed, at 18 percent of the subtotal
  - Add the GST and subtotal for the grand total
  - Show all the details

Beware

By default the data read by input() is treated as a string. Use eval() to evaluate the mathematic expression.

```
#prg5.py
#Ask How many burgers they want
no_of_burgers = eval(input("Enter how many burgers do you want: "))
#Ask the cost of each burger
cost_of_burger = eval(input("Enter the cost of each burger: "))
#Calculate the total cost of the burgers as our subtotal
sub_total = no_of_burgers * cost_of_burger
#Calculate the GST owed, at 18 percent of the subtotal
gst = sub_total * 18/100
#Add the GST and subtotal for the grand total
total = sub_total + gst
#Show all the details
print("Subtotal is $",sub_total)
print("GST is $",gst)
print("Total cost is $",total)
>>>
================ RESTART: prg5.py ================
Enter how many burgers do you want: 20
Enter the cost of each burger: 2
Subtotal is $ 40
GST is $ 7.2
Total cost is $ 47.2
>>>
```

JUST LEARN

Lists

- ✓ Lists are an incredibly useful *compound* data type.
- ✓ Lists are mutable – it is possible to change their contents. They contain the additional mutable operations.
- ✓ Lists are nestable. Feel free to create lists of lists…

```
mylist = [42, 'apple', u'unicode apple', 5234656]
print (mylist)
mylist[2] = 'banana'
print (mylist)
print (mylist[2])
```

```
[42, 'apple', u'unicode apple', 5234656]
[42, 'apple', 'banana', 5234656]
'banana'
```

JUST LEARN

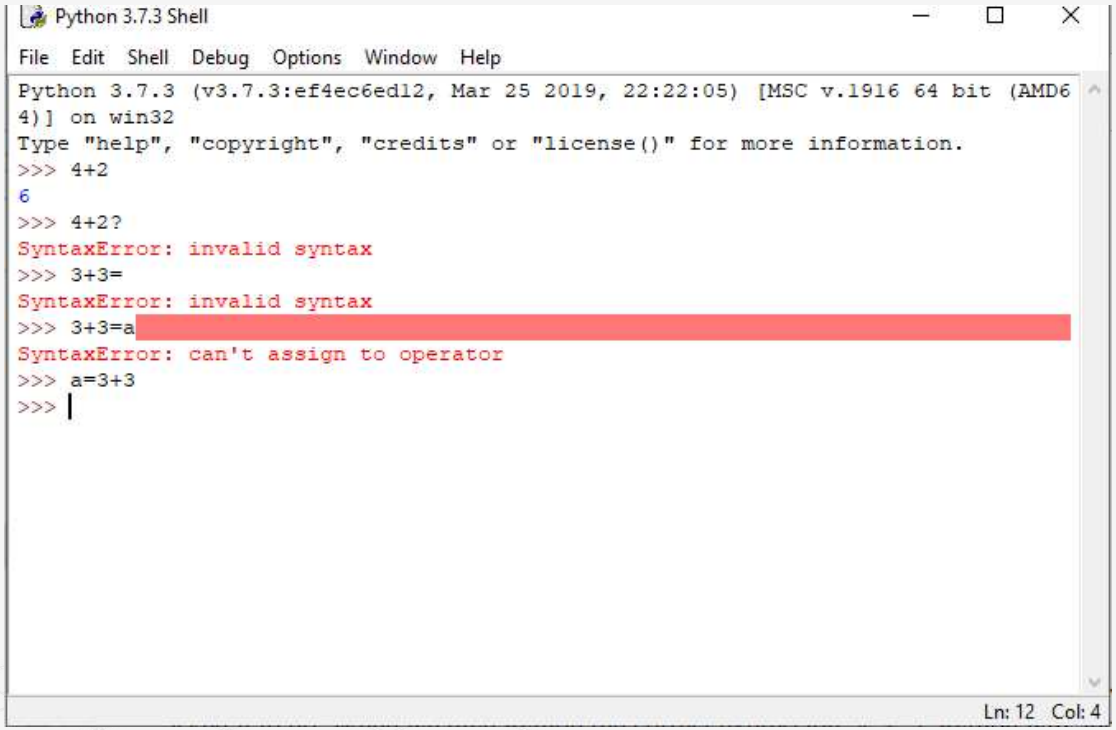
Common sequence operations

- ✓ All sequence data types support the following operations.

Operation	Result
`x in s`	True if an item of s is equal to x, else False.
`x not in s`	False if an item of s is equal to x, else True.
`s + t`	The concatenation of s and t.
`s * n, n * s`	n shallow copies of s concatenated.
`s[i]`	ith item of s, origin 0.
`s[i:j]`	Slice of s from i to j.
`s[i:j:k]`	Slice of s from i to j with step k.
`len(s)`	Length of s.
`min(s)`	Smallest item of s.
`max(s)`	Largest item of s.
`s.index(x)`	Index of the first occurrence of x in s.
`s.count(x)`	Total number of occurrences of x in s.

Syntax Errors

- ✓ Syntax is a set rules we follow to build a sentence or statement in a language.
- ✓ The computer can not understand the statement if proper syntax is not followed and that is show as syntax error.
- ✓ You can use Python shell to understand the syntax errors



JUST LEARN

What You Learned

- ✓ At this point you should
  - Understand
    - Variable and rules for creating variables
    - Assignment and arithmetic operators
    - Integers and floating-point numbers
    - Strings and text
    - How to evaluate mathematical expressions from strings
    - Lists

JUST LEARN

Lab Time

JUST LEARN