# Python Level 1

Table of Contents

[Python Level 1 1](#_Toc69981808)

[Course Readings 2](#_Toc69981809)

[Reserved Words 2](#_Toc69981810)

[Your First Program 2](#_Toc69981811)

[Variables 3](#_Toc69981812)

[Getting Input from the User 3](#_Toc69981813)

[Type Conversion 4](#_Toc69981814)

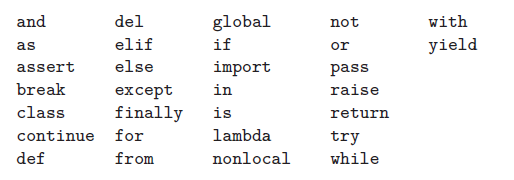
[Using Python to Discover the Variable Type 4](#_Toc69981815)

## Course Readings

* <https://www.py4e.com/>
  + Lessons => <https://www.py4e.com/lessons>
  + Book => <https://www.py4e.com/book>
* Youtube Course => <https://youtu.be/_uQrJ0TkZlc?list=PLF-r7uGjIN7CZ-zMa-YOcU4Bi4otMnTgL>

## Reserved Words

* Python’s reserve words are as follows:



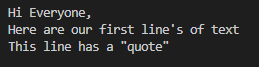
## String Information/Modification

### Escape Characters

* If you were to write the following line of code, because of the use of single quotes *and* apostrophe, python would not be able to tell where the string quotes begin and end:
  + print('Python's Course')
* To avoid this, we can:
  + use double quote for the string text, and the apostrophe will function as normal
    - print("python's course")
* But what if you wanted to have the word *course* enclosed in double quotes? We would simply reverse what we did above: enclose our text in single quotes and use the double quotes inside:
  + print('Python "course"')

### Multiline Text

* For our examples so far we have been using single line text outputs. In order to write multiple lines of text (i.e. an email or para) we need to enclose our text in triple quotes: ‘’’ TEXT HERE ‘’’
  + # Multi line text
  + print('''
  + Hi Everyone,
  + Here are our first line's of text
  + This line has a "quote"
  + ''')



### Selecting a Single Character from a String

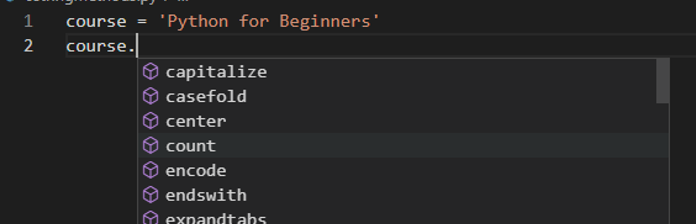
* Pretend we have the following line of code:
* message = 'Hello, how are you?'
* If we wanted to extract a single character based on its position, we could do the following:
  + message = 'Hello, how are you?'
  + print(message[0])
  + (Where [0] is the position of the character we want to isolate. The above, for example, will give us the first position character “H”
* We can also use a negative index to get our characters starting from the end.
  + message = 'Hello, how are you?'
  + print(message[-3])
  + Will give us the value “o”
* We can also get *multiple* characters at once:
  + message = 'Hello, how are you?'
  + print(message[0:3])
  + Will give us the first three letters
  + If you were to write the following [0:] and give no ending index, it would print all the characters from whatever you noted the starting index as – in this case it would just print the entire sentence

### Formatted Strings

* Pretend we had the following code:
  + A variable for a first name, last name, and a message
  + firstName = 'John'
  + lastName = 'Smith'
  + message = firstName + ' [' + lastName + '] is a coder'
  + print(message)
* While this is a valid solution to print the message, we can make use of **python formatted strings** to clean up our concatenation in the following way:
  + We use an ‘f’ to declare the string as formatted and curly braces to surround our variables
  + firstName = 'John'
  + lastName = 'Smith'
  + message = firstName + ' [' + lastName + '] is a coder'
  + msg = f'{firstName} [{lastName}] is a coder'
  + print(msg)

### String Methods

* A **method** is any function that is *specific* to a particular object – i.e. a series of function that are relative to strings. Python has a variety of built-in methods that we can apply specifically to strings (i.e. making text upper or lower case).
* We can use these functions via the dot-operator and we can see a full list of them in the code editor by creating a variable and setting a period by it:



#### Upper/Lower Case Text Method

* + course = 'Python for Beginners'
  + print(course.upper())
  + Will turn our course text variable to uppercase letters. Now, this **method** *will not modify the original string*. It takes our original, title-case, string value, applies the method to it, and returns that altered value.

#### The Find Method

* We can use the **find()** method to return the first instance of a character positions of a given search. This can be single characters or entire words/phrases.
* ***Note:***the **fund()** method is case sensitive, so searching for “P” will produce different results than “p”
  + course = 'Python for Beginners'
  + print(course.find('P'))
  + 
  + This returns the character position 0 because the first instance of “P” is at the 0-index
  + course = 'Python for Beginners'
  + print(course.find('for'))
  + 
  + This returns the character position 7 because the word “for” begins at the 7th-index position
  + If we search for a letter/word that *is not* in our variable, we will receive a return index value of -1
* **Calculating the Number of characters in a string**
  + We can use the built in **len()** function to count the number of characters in a string.
  + ***Note:***The **len()** function is a general purpose function because it can be applied to *other* things that are not strings. This means it can be used to count more than just characters (i.e. items in a list, array elements, etc.)
  + course = 'Python for Beginners'
  + print(len(course))

## Your First Program

* Our first program will be a simple “Hello World” program:
  + print ('Hello World')
* Let’s add a few more print statements:
  + print ('Hello World')
  + print ('And hello Theo')
  + print ('And hello Pisha')
* If we run the following again, you will see the order in which the code was executed – in this case, line by line, starting from the top.
* Now, let’s add the following line of code to our list:
  + print ('\*' \* 10)
  + When we run the code we see that the “\*” icon has been duplicated x10. The code itself is called an **expression** – a piece of code that produces a value

## Variables

* **Note: Variables in python are *case sensitive*.** It is considered best-practices to define all variables in lower cases and, if made up of multiple words, separated by an underscore.
* Variables in python are written using the following syntax:
  + x = 10
  + x=10
  + (spaces will not affect the running of the code)
* What if we were to update the value of our x-variable, like so, what would get printed:
  + x=10
  + x=11
  + print(x)
  + Because we *know* python processes code line-by-line, from the top down, the x-variable will be *reset* to it’s *latest* value – in this case 11
* So far we have been dealing with whole integers, but python can also process float-point operators:
  + x=1.1
  + y=1.2
  + print(x\*y)
  + This will still produce the valid result of 1.32
* Python can also use Boolean logic for variables:
  + # Boolean logic (True and False need to be capitalized)
  + is\_this\_a\_program = True
  + print(is\_this\_a\_program)
  + **Note: True and False** **need to be capitalized variables when Boolean logic is employed**

## Getting Input from the User

* To gather input from the user we use the following syntax:
  + name = input('What is your name? ')
  + print('Hi ' + name)
  + In this example:
    - the input value is stored to the variable name
    - We then print a message “Hi” and concatenate the name that the user input

## Type Conversion

* To illustrate this section, we are going to write a program that will:
  + Ask the year we were born in
  + Calculate our age

Print it in the terminal

* year = input('What year were you born? ')
* ageCalculator = 2021 - year
* print('Your are ' + ageCalculator + ' years old')
* If we run this program, we will get the following error: “Type Error. Unsupported operand…” This is because, though the user will input a number, anything that is typed into the terminal **is treated as a string** – even if it is a number.
* In this case our year variable, to python, looks like the following:
  + year = ‘1987’
* To properly process a mathematical expression, we need to first ensure that *both* our variables are integers, and so we need to convert the **string** value to an **integer** value, and there are a variety of built-in conversion functions we can leverage
  + **int()** function
    - To convert a **string** to an **integer**
  + **str()** function
    - To convert an **integer** to a **string**
  + **float()** function
    - To convert a **string** to a **float-point number**
  + **bool()** function
    - To convert a **string** to a **Boolean** value
* So, to have our code run properly, we need to amend it in the following way:
  + year = input('What year were you born? ')
  + age = 2021 - int(year)
  + print(age)

### Using Python to Discover the Variable Type

* In addition to converting a given variable to a type, we can use python to tell us what variable type the current variable exists as. To do this we can use the **type()** function
  + year = input('What year were you born? ')
  + print(type(year))
  + age = 2021 - int(year)
  + print(type(age))
* If we run this code, we will get the following result:

