

**Variables**:

We use variables to store data in computer’s memory.



When we run the above code, python interpreter will allocate some memory and store this number 1000 in that memory space.

Then the variable *students\_count* will reference to that memory location. *This variable is just like a label to that memory location*.

We can use this variable or label anywhere in our program in order to access that memory location and the data stored there.

*What kind of data can we store in computer’s memory*?

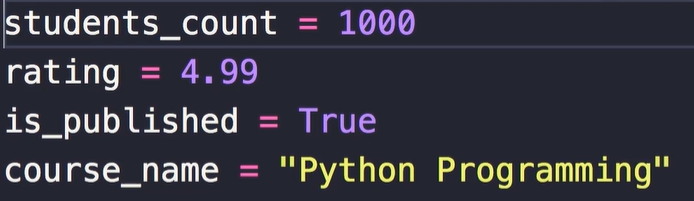
🡪 We have several different kinds of data. First, we are going to look at built in *primitive types* in Python.

🡪 Primitive types can be number, Booleans and Strings.

Note: Boolean values True and False must always start with capital letter.

**Variable Names**:

Take these variables as an example,



Here we use best practices to name our variables which are:

🡪 Variable names must be descriptive and meaningful.

🡪 Use lower case letters to name variables.

🡪 Use underscore to separate multiple words.

🡪 Put some space around equal sign.

Write clean and beautiful code so that other people can read it like a story. It should be formatted properly

**Strings**:

We use triple quotes to enclose a string which is too long like an email message.

message = """

Hi John,

This is Himanshu

Blah blah blah...

"""

*Get length of string*:

To get length of string or number of characters in a string use len function.

course = "Python Programming"

print(len(course))

O/P: 18

Note: If you want to access a particular character in the string, we use square bracket notation. First letter always starts with index 0, so use [0] for accessing first letter of string.

*Print last character of the string*:

course = "Python Programming"

print(course[-1])

O/P: g

*Slicing*:

To slice a part of the string we need two parameters, a start index and an end index with colon **:** in between.

*Get first six letters of the string*:

course = "Python Programming"

print(course[0:6])

O/P: Python

Note: If we do not put anything in place of start index and end index we get a copy of the original string

course = "Python Programming"

print(course[:])

O/P: Python Programming

**Escape Sequences**:

We have this string here…

course = "Python Programming"

Now what if we want to add a double quote in the middle of the string,

 we see an error.

We can prefix this with \ (backslash), which is a special character in python. We have a jargon for that called *escape character*.

We use it to escape the character after like this…



If we print this string, we get

O/P: Python "Programming

So backslash is an escape character and backslash + double quote is an *escape sequence*.

In Python strings we have a few other escape sequences that you should be aware of,

**#**, **\**, **\\**, **\n**

**Formatted Strings**:

Concatenate two strings using formatted string approach:

first\_name = "Himanshu"

last\_name = "Pandey"

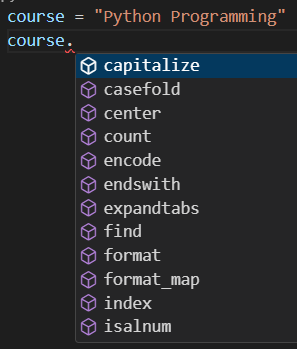
full\_name = f"{first\_name} {last\_name}"

print(full\_name)

O/P : Himanshu Pandey

When using formatted string we can put any valid expression inside curly braces.

**String Methods**:



When we use dot operator on a string, we see multiple methods that support this string.

course = "Python Programming"

print(course.upper())

print(course)

upper method converts string to upper case letters and return a new string.

O/P: PYTHON PROGRAMMING

Python Programming

Note: In above example, upper method returns a new string and not modify the original string.

*Remove whitespaces from the string*:

We use strip method, to remove leading and trailing whitespaces.

course = "  Python Programming "

print(course.strip())

print(course)

O/P:



*Get index for a sequence of characters*:

We use the find method where as an argument we pass another string for which we want to know the index of.

course = "Python Programming"

print(course.find("Pro"))

O/P : 7

If we pass ‘pro’ instead we get O/P: -1, which means It cannot find this sequence of characters anywhere in the string. So make sure you get the case sensitive part right.

*Replace sequence of characters with another sequence*:

We use replace method and pass two arguments, first one is string that we want to replace and second is what string we want it to replace with.

course = "Python Programming"

print(course.replace("P", "J"))

O/P: Jython Jrogramming

*Check the existence of a character or a sequence of characters in your string*:

We use in operator where we use it in form of an expression. An expression which returns a Boolean value.

course = "Python Programming"

print("Pro" in course)

O/P: True

Similar to in operator we have not operator which we use to see if our string *does not contain* a character or sequence of characters.

**Numbers**:

We have two different types of divisions.

print(10 / 3)

O/P: 3.3333333333333335

With **/** division operator, we get a floating point number.

If you want an integer after division use **//**(double slash).

print(10 // 3)

O/P: 3

There is an augmented operator as well,

x=10

Suppose we want to increase the value of x by 3, we can write,

x += 3  # same as x = x + 3

print(x)

O/P: 13

**Working with Numbers**:

In this we will look at few useful functions to work with numbers.

*Round up a float/decimal value*:

We have a function called round for rounding the numbers.

print(round(2.9))

O/P: 3

*Return absolute value of a number*:

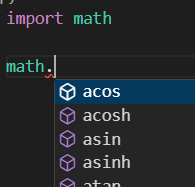
If we pass -2.9 to abs function,

print(abs(-2.9))

O/P: 2.9

If you want to write a program with complex mathematical calculations, you need to use math module.

Note: A Python module is like a separate file with some python code. So in Python we have this math module which includes many mathematical functions for working with numbers.

🡨We import math which is an object so with *dot operator* we can access various methods available to this object.

*Get the ceiling of a number*:

print(math.ceil(2.2))

O/P: 3

Note: To get the list of all methods available in this math module. Google *python 3 math module* and on the page you will see all the available functions.

**Type Conversions**:

Let us take a look at another useful built in function in Python.

*Get input from the user*:

We use input function for this purpose. As an argument we pass a string which is a label that will be displayed in the terminal.

x = input("X: ")

Here we save this input into a variable called x.

x = input("X: ")

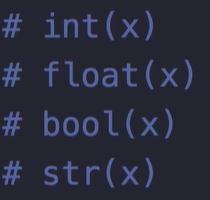
print(type(x))

Now if we want to check the type of this variable we get,



So the return we get from input function is always a string.

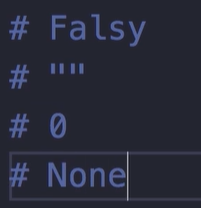
We can convert the type of this variable using inbuilt type converters.



int, float, bool, str are the functions which convert variable types to integer, float, Boolean and string respectively.

Note: Most of the functions are self- explanatory, the only tricky one is bool. Because in python we have a concept of *truthy* and *falsy* values.

These are the values which are not exactly Boolean True or False, but they can be interpreted as one.

🡨These are Falsy values, empty string, 0 and None