**Word Count:1127**

Video, Typecasting & Loops in Python

In this video, we will learn about programming concepts like Typecasting & Loops in Python for the Raspberry Pi 4.

When you create a variable in Python, it’s automatically assigned a type based on what it is. You can check this with the type() function. In the shell interface, enter:

a = 1

b = “two”

Now use the type function to check the type of each variable.

type(a)

It will say class str & for

type(b)

It will say class int

This concept is important because different data types work together in a variety of ways, and they don’t always play nicely together always. For example, if you add together two strings they are combined.

name = “Naveen”

job = “Product Engineer"

print( “My name is ” + name + “ and I am a “ + job)

The output will be a concatenated string. Concatenated is the fancy programming term for “joined up”. Please don’t forget to put spaces at appropriate places so that the words won’t stick with each other.

Lets now try out how numbers work with this method.

Type

num1 = 4

num2 = 7

print(num1 + num2)

Instead of concatenation, python performs a mathematical operation, to get an output of 11 rather than 47.

Now, what happens when you want to add a string and integer together. Type the following and try out:

name = pi

number = 4

print(name+number)

You will get an error message: “TypeError: can only concatenate str (not “int) to str”. This error is because python can’t add together a string and an integer. Now try out the following

print (name\*number)

Obviously, it should also be an error right? You are in for a surprise. Running the code reveals that you will get “pipipipi”. Thus you can multiply strings with integer.

But how can you work with an integer and a string? For example, we need to use a combination of integer and string at places where we need to display information along with a variable number. For instance, it will be needed in a calculator project.

In such situations, we need to use a concept called typecasting. It is the process of converting a data type to another. So we need to convert the integer to string, using the str() function. So if we want to produce an output : “pi4”, we have to type the following code

name = “pi”

number = 4

print(name + str(number))

Now let’s look at a situation where we need to convert a string to an integer. You will need to use it when accepting an input number from the user. For example, let's create a program that asks for a number and exponent and raise the number to the power of the exponent using the double-asterisk symbol

Type the following

number = input( “Enter a number: “)

exponent = input(“Enter an exponent: “)

result = int(number) \*\* int(exponent)

print(result)

Here you can see that the int() function is used to convert the string to a number before doing the mathematical operation.

Instead of doing in the shell, let's continue by saving this code in the editor.

Now let's modify the project. If you want to include a message along with the result, you need to use the str() function like this

print(number + “ raised to the power “ + exponent + “ is “ + str(result))

Variables, types, and typecasting can be a bit tricky at first. Python is a lot easier to use because it dynamically changes the type of a variable to match the thing you put in it. However, it does mean you have to be a bit careful.

Now let's look at the concept of loops in Python. Computers are great because they don’t mind doing the same stuff over and over again. Their hard-working nature makes them ideal for repetitive work. This is where the concept of loops come into play. There are mainly two types of loops in Python. One is a while loop, and the other is a for loop.

The syntax for a while loop is the following

While followed by a condition then a colon. On the next line, after an indent, we put the instructions we want to execute on the condition getting satisfied. In case of the while loop, as long as the condition is satisfied, the instruction will keep on executing in a loop. So what are these conditions? These are instructions that either produces a true or false as the return type. Thus as long as the condition is True, the while loop will execute the instructions inside it.

Let's look at an example.

In this example, we will print out “Hello World” three times using while loop.

The code will look like this

i=0

while i<3:

print(“Hello World”)

i=i+1

Now save and run the program. You will get the following results

The ‘lesser than’ symbol is called a conditional operator. Some of the other types of conditional operators are shown here. If you want to know more about them, please check out the resources section.

The While loop is traditionally used when you have a block of code which you want to repeat forever until the condition becomes false. It is also used to have an infinite loop using the ‘while True” command. You could use a for loop with a huge number in order to gain the same effect as a while loop, but what's the point of doing that when you have a construct that already exists? As the old saying goes, "why try to reinvent the wheel?".

There is another type of loop in Python called the for loop, which are used to repeat a block of code a fixed number of times.

This is the syntax for the for loop

For followed by the variable, then in followed by sequence.

Here <variable> is a variable that is used for iterating over a <sequence>.

On every iteration, it takes the next value from the sequence until the end of the sequence is reached.

Let’s do an example project

Let's say you want to print numbers from 1 to 10

So the code will look like this

for i in range(1,11):

print(i)

Now you can see that numbers from 1 to 10 are printed out

Now modify the code to this

for i in range(0,11,2)

And rerun the code.

Now you can see that numbers are from 0 to 10 are printed with a sequence jump of 2. Thus the third parameter determines the incrementing.

Thus, we have covered both the types of loops in python.

Summary

In this video, we have covered the following

* Typecasting
* Loops

In the next video, we will learn about conditional branching and functions in python.