**Topic 3 - Variables for Numbers**

**1. Assigning Numbers to Variables**

In Python, you can assign more than just strings to variables; you can also assign numbers. When you assign a number to a variable, Python recognizes it as a numeric value that can be used in calculations.

For example:

***weight = 70***

Here, the variable weight refers to the value 70. If you want to add 25 to weight, you can simply write:

***weight + 25***

Python will understand that weight means 70, and it will calculate 70 + 25, resulting in 95.

**2. Numbers vs. Strings**

A *number* in Python is different from a *string*:

* A **number** is *not* enclosed in quotes, allowing Python to use it for math operations.
* A **string** is enclosed in quotes, and Python treats it as text rather than a number.

For example:

*height = "170" # height is a string*

*age = 25 # age is a number*

Python can’t add a string and a number directly. So, if you try to add 7 to "170", Python will throw an error because "170" is in quotes and is therefore treated as a string.

**3. Using Variables in Calculations**

You can use variables for math calculations, and even create new variables based on existing ones.

**Example 1: Assigning Calculations to New Variables**

*original\_num = 23*

*new\_num = original\_num + 7*

*print(new\_num) # Output: 30*

Python replaces original\_num with 23, then adds 7, and assigns the result 30 to new\_num.

**Example 2: Using Multiple Variables in a Calculation**

*original\_num = 23*

*num\_to\_be\_added = 7*

*new\_num = original\_num + num\_to\_be\_added*

*print(new\_num) # Output: 30*

**4. Variables as Their Own New Value**

A variable can even update itself with a new calculated value:

*original\_num = 90*

*original\_num = original\_num + 10*

*print(original\_num) # Output: 100*

The variable original\_num now holds the value 100 after adding 10 to its previous value of 90.

**5. Variables Can Change Types**

In Python, a variable can change from one type to another. For instance, your\_age might start as a string:

*your\_age = "21" # Now `your\_age` is a string*

But later, you can change it to a number:

*your\_age = 21 # Now `your\_age` is a number*

After assigning 21 as a number, you can perform mathematical operations on your\_age.

**6. Naming Variables with Numbers**

While variable names can include numbers, they cannot *start* with a number:

*# Incorrect:*

*1st\_prime\_number = 2 # Starts with a number - not allowed*

*# Correct:*

*prime\_number\_1st = 2 # Number comes after the letters - allowed*

Python doesn’t differentiate between variable names for numbers or strings based on the name alone, so choosing meaningful names helps others (and yourself!) understand your code.

**7. Integer and Float Variables**

In the examples so far, we’ve used integers (whole numbers like 2, 47, 0, and -5). You can also assign floats (numbers with decimal points) to variables:

*temperature = 37.5 # Float*

*balance = -100.75 # Float*

*percentage = 0.003 # Float*

Both integers and floats are valid types for Python variables, allowing flexibility for different types of calculations.

**Summary**

* Use variables to store numbers, making calculations easier and cleaner.
* Numbers aren’t enclosed in quotes; strings are.
* You can add or change variable values, even using a variable to calculate a new value for itself.
* Variable names should be meaningful but can include numbers as long as they don’t start with one.