Math and Data Types

In this example, we'll build on what we've learned for input and output and learn about fundamental data types and math operations in Python.

Why is this important?

Math is something that you're going to use a lot in programming. It's important to understand how to do math in Python and how to use the different data types.

Doesn't matter if you're using Artificial Intelligence, Web Development, or Data Science, you'll be using math and the data types we're going to introduce in this example.

Steps

1. Create a file called `math_and_data_types.py` and open it in your text editor.

2. Let's start with introducing the `int` data type.

- So far we've only used the `string` data type in this course. This is probably one of the most common data types you'll use in Python, to display information to the user.
- Next, we'll introduce the `int` and `float` data types. This is short for integer and is used to represent whole numbers. An important note is that Python is dynamically typed, so you don't have to specify the data type when you create a variable. You can just assign a value to a variable and Python will figure out what data type it is.
- Type the following code into your `math_and_data_types.py`

```python
# Learning about Math
days_until_assignment_due = 14
```

- a few things to observe here:
    - the number `14` is an integer and doesn't have any quotes around it. This is how you define an integer in Python.
    - `days_until_assignment_due` is using "Snake Case" which is the convention for naming variables in Python. This is where you use all lowercase letters and separate words with an underscore `_`. This is the convention for naming variables in Python.
- Let's print this string out in an f-string. Type the following code into your `math_and_data_types.py` file.

```python
# Learning about Math
days_until_assignment_due = 14
print(f"Assignment due in {days_until_assignment_due} days")
```

- run that code in your terminal.
    - navigate to the directory where your `math_and_data_types.py` file is located.
    - run the command `python math_and_data_types.py` in your terminal and press enter.
        - the terminal should print out `Assignment due in 14 days`

3. Let's learn about and use some math operators in Python

- Python has a lot of operators, but we'll just be focusing on the fundamental math operators for now. Here's a list of common operators that you'll need in Python.
    - `+` addition
    - `-` subtraction
    - `*` multiplication
    - `/` division (this will always return a float)
    - `**` exponent (you can think of this as `^` in math)
    - `%` modulus (you can think of this as the remainder)
- In our code, let's create a variable named `days_playing_squash` because we want to play squash every day. This is going to subtract from our `days_until_assignment_due`. Type the following code into your `math_and_data_types.py` file to display this.

```python
# Learning about Math
days_until_assignment_due = 14
print(f"Assignment due in {days_until_assignment_due} days")

# Subtraction Example
days_playing_squash = 2
print(f"Days spent playing squash: {days_playing_squash}")
print(F"Assignment due in: ")
print(days_until_assignment_due - days_playing_squash)
```

- run the command `python math_and_data_types.py` in your terminal and press enter, the output in your terminal should look like this:

```
$ python math_and_data_types.py
Assignment due in 14 days
Days spent playing squash: 2
Assignment due in:
12
```

- A few notes about this.
    - We're using the `-` operator to subtract the `days_playing_squash` from the `days_until_assignment_due` variable. You can see that the output is `12` which is the result of `14 - 2` where we're using the variables!
    - Currently `days_until_assignment_due` is still 14, we haven't changed the value of the `days_until_assignment_due` variable. We've just used it in the print statement.

4. Let's do some more math, and learn about reassigning variables.

- For some reason, someone just gave you a time machine. This is great because you just lost a couple of days to work on your assignment playing squash. Let's use addition to add 3 days to the `days_until_assignment_due` variable. Type the following code into your `math_and_data_types.py` file.

```python
# Learning about Math
days_until_assignment_due = 14
print(f"Assignment due in {days_until_assignment_due} days")

# Subtraction Example
days_playing_squash = 2
print(f"Days spent playing squash: {days_playing_squash}")
print(F"Assignment due in: ")
print(days_until_assignment_due - days_playing_squash)

# Addition Example
time_machine_days_added = 3
print(f"Days added by the time machine: {time_machine_days_added}")
print(f"Assignment due in: ")
print(days_until_assignment_due + time_machine_days_added)
```

- When you run this code in your terminal you should see something like this:

```
$ python math_and_data_types.py
Assignment due in 14 days
Days spent playing squash: 2
Assignment due in:
12
Days added by the time machine: 3
Assignment due in:
17
```

- Let's take a look at this output.
  - You can see that we've added 3 days to the `days_until_assignment_due` variable because `14 + 3` is `17`. The `days_until_assignment_due` variable is still 14, but we've added 3 to it when we printed it out, but we want to take into account the `days_playing_squash` variable. Let's do that now so that we have the correct number of days until the assignment is due.
- Change the existing code in your `math_and_data_types.py` file to the following:

```python
# Learning about Math
days_until_assignment_due = 14
print(f"Assignment due in {days_until_assignment_due} days")

# Subtraction Example
days_playing_squash = 2
print(f"Days spent playing squash: {days_playing_squash}")
# Reassign days_until_assignment_due to the result of the subtraction
```

```
days_until_assignment_due = days_until_assignment_due -
days_playing_squash
print(f"Assignment due in {days_until_assignment_due} days")

# Addition Example
time_machine_days_added = 3
print(f"Days added by the time machine: {time_machine_days_added}")
# Reassign days_until_assignment_due to the result of the addition
days_until_assignment_due = days_until_assignment_due +
time_machine_days_added
print(f"Assignment due in {days_until_assignment_due} days")
```

- let's take a look at the output of this code:

```
$ python math_and_data_types.py
Assignment due in 14 days
Days spent playing squash: 2
Assignment due in 12 days
Days added by the time machine: 3
Assignment due in 15 days
```

- You can see that we've used the `=` operator to reassign the `days_until_assignment_due` variable to the result of the subtraction and addition. This is important to understand because you'll be using this a lot in your programs.
    - When reassigning variables should **always** be the same data type as the original variable. In this case, we're subtracting an integer from an integer and adding an integer to an integer. This is why we don't have to worry about the data type.
        - in statically typed languages this is enforced by the compiler, but in dynamically typed languages like Python, it's up to the programmer to make sure that the data types are the same.

5. More Practice with multiplication

- Someone has given you a device that allows you to warp time so it goes twice as slowly. This is great because you can get twice as much done in the same amount of time. Let's use the `*` operator to multiply the `days_until_assignment_due` variable by `2`. Type the following code into your `math_and_data_types.py` file.

```
# Learning about Math
days_until_assignment_due = 14
print(f"Assignment due in {days_until_assignment_due} days")

# ... Subraction and Addition code removed for brevity

# Multiplication Example
time_warp_mutitplier = 2

# Reassign days_until_assignment_due to the result of the multiplication
```

```
print(f"Time warp multiplier: {time_warp_mutitplier}")
days_until_assignment_due = days_until_assignment_due *
time_warp_mutitplier
print(f"Assignment due in {days_until_assignment_due} percieved days")
```

- The output of the code should be the following:

```
$ python math_and_data_types.py
Assignment due in 14 days
Days spent playing squash: 2
Assignment due in 12 days
Days added by the time machine: 3
Assignment due in 15 days
Time warp multiplier: 2
Assignment due in 30 percieved days
```

- This is great because we've doubled the amount of time we have to work on our assignment. Let's take a look at the code.
  - We've created a variable called `time_warp_multiplier` and assigned it the value of `2`. This is because we want to double the amount of time we have to work on our assignment.
  - We've then used the `*` operator to multiply the `days_until_assignment_due` variable by the `time_warp_multiplier` variable. This is because we want to double the amount of time we have to work on our assignment.
  - We've then reassigned the `days_until_assignment_due` variable to the result of the multiplication. This is because we want to double the amount of time we have to work on our assignment.

6. More Practice with division, modules and floats

Division is a tad bit more complicated because we can have fractional numbers (named floats in Python). Let's take a look at how this works.

- Your instructor has unfortunately given put a slowing spell on you that makes your typing speed go down by 7 times (yep 7 times!). This is bad because you're going to take 7 times longer to write your assignment. Let's use the `/` operator to divide the `days_until_assignment_due` variable by `7`. Type the following code into your `math_and_data_types.py` file, you'll see that the result is a `float`.

```
# Learning about Math
days_until_assignment_due = 14
print(f"Assignment due in {days_until_assignment_due} days")

# ... code removed for brevity but days_until_assignment_due is 30 at this
point

# Division Example
slowing_spell_divisor = 7
```

```
print(f"Instructor Typing slowing spell {slowing_spell_divisor} times")

# Result of division
print("Result of the division using the slowing_spell_divisor: ")
print(days_until_assignment_due / slowing_spell_divisor)
```

- The output of the code should be the following:

```
$ python math_and_data_types.py
Assignment due in 14 days
Days spent playing squash: 2
Assignment due in 12 days
Days added by the time machine: 3
Assignment due in 15 days
Time warp multiplier: 2
Assignment due in 30 percieved days
Instructor Typing slowing spell 7 times
4.285714285714286
```

- You can see that the result of the division is `4.285714285714286`. This is a `float` because we're dividing an integer by an integer. This is because we want to take into account the slowing spell that your instructor put on you.
  - Note that the result of the division is a `float` because we're dividing an integer by an integer. This is because we want to take into account the slowing spell that your instructor put on you.
  - If we want to just get the `4` from the `4.285714285714286` we can use the `//` operator. This is called the floor division operator and will always return an integer. Let's use this operator to get the number of days until the assignment is due. Type the following code into your `math_and_data_types.py` file.
- Let's take a look at using the floor division operator `//` in your code under the division that you just type the following code.

```
# other code removed for brevity
# Division Example
slowing_spell_divisor = 7
print(f"Instructor Typing slowing spell {slowing_spell_divisor} times")

# Result of division
print("Result of the division using the slowing_spell_divisor: ")
print(days_until_assignment_due / slowing_spell_divisor)
print("Result of the floor division using the slowing_spell_divisor: ")
print(days_until_assignment_due // slowing_spell_divisor)
```

- The output of the code should be the following:

```
$ python math_and_data_types.py
Assignment due in 14 days
```

```
Days spent playing squash: 2
Assignment due in 12 days
Days added by the time machine: 3
Assignment due in 15 days
Time warp multiplier: 2
Assignment due in 30 percieved days
Instructor Typing slowing spell 7 times
Result of the division using the slowing_spell_divisor:
4.285714285714286
Result of the floor division using the slowing_spell_divisor:
4
```

- Note that the result of the floor division is `4` and not `4.285714285714286`. This is because we're using the floor division operator `//` which will always return an integer.
  - You can see that `4 * 7` is not `30` but `28`. This is because we're using the floor division operator `//` which will always return an integer or `int`.
- Let's take a look at how we can get the remainder, this uses the modulus operator `%`. Type the following code into your `math_and_data_types.py` file.

```python
# other code removed for brevity

# Division Example
slowing_spell_divisor = 7
print(f"Instructor Typing slowing spell {slowing_spell_divisor} times")

# Result of division
print("Result of the division using the slowing_spell_divisor: ")
print(days_until_assignment_due / slowing_spell_divisor)
print("Result of the floor division using the slowing_spell_divisor: ")
print(days_until_assignment_due // slowing_spell_divisor)
print("Remainder of the division using the slowing_spell_divisor: ")
print(days_until_assignment_due % slowing_spell_divisor)
```

- The output of the code should be the following:

```
$ python math_and_data_types.py
Assignment due in 14 days
Days spent playing squash: 2
Assignment due in 12 days
Days added by the time machine: 3
Assignment due in 15 days
Time warp multiplier: 2
Assignment due in 30 percieved days
Instructor Typing slowing spell 7 times
Result of the division using the slowing_spell_divisor:
4.285714285714286
Result of the floor division using the slowing_spell_divisor:
4
```

```
Remainder of the division using the slowing_spell_divisor:
2
```

- You can see that the remainder of the division is `2`. This is because we're using the modulus operator `%` which will always return the remainder of the division.
  - Explained: `7` fits into `30` four times and `7 * 4` is `28`, the remainder is `2`. The modulus operator `%` will always return the remainder of the division and will be an integer or `int`.
- let's create a variable named `special_remainder_power` that contains the remainder, and let's reassign the floor division to the `days_until_assignment_due` in that order.. The code should now look like this:

```
# Division Example
slowing_spell_divisor = 7
print(f"Instructor Typing slowing spell {slowing_spell_divisor} times")
# special_remainder_power and days_until_assignment_due variables
special_remainder_power = days_until_assignment_due %
slowing_spell_divisor
days_until_assignment_due = days_until_assignment_due //
slowing_spell_divisor

# Result of division
print("Result of the floor division using the slowing_spell_divisor: ")
print(days_until_assignment_due)
print("Remainder of the division using the slowing_spell_divisor: ")
print(special_remainder_power)
```

- The output should be fairly similar as the above output, but now we have some variables we can use.

7. Let's use the `**` operator to raise a number to a power.

Let's take a look at how to use exponents in Python. This is done using the `**` operator.

- Using your `special_remainder_power` variable, you have gotten a power to multiply the amount of time you have to work on your assignment. Let's use the `**` operator to raise the `days_until_assignment_due` variable to the power of `special_remainder_power`. Type the following code into your `math_and_data_types.py` file.

```
# other code removed for brevity (you need special_remainder_power)

print("Special Remainder Power")
days_until_assignment_due = days_until_assignment_due **
special_remainder_power
print(f"Assignment due in {days_until_assignment_due} percieved days")
```

- The output of the code should be the following:

```
$ python math_and_data_types.py
Assignment due in 14 days
Days spent playing squash: 2
Assignment due in 12 days
Days added by the time machine: 3
Assignment due in 15 days
Time warp multiplier: 2
Assignment due in 30 percieved days
Instructor Typing slowing spell 7 times
Result of the floor division using the slowing_spell_divisor:
4
Remainder of the division using the slowing_spell_divisor:
2
Special Remainder Power
Assignment due in 16 percieved days
```

- The result is `16` because `4 ** 2` (or 4^2 in plain math) is `16`. This is because we're using the `**` operator to raise the `days_until_assignment_due` variable to the power of `special_remainder_power`.

Conclusion

In this example we learned about new data types and how to do math using python. As well hopefully we've reinforced the concept of reassigning variables.

Any of the math that we did here with `int` can also be done with `float`. The only difference is that the result will be a `float` instead of an `int`.

In the future we'll also introduce the `math` module which gives you even more math tools to use in your programs.