**Data Types Recap**

In Python, text in between quotes -- either single or double quotes -- is a string data type. An integer is a whole number, without a fraction, while a float is a real number that can contain a fractional part. For example, 1, 7, 342 are all integers, while 5.3, 3.14159 and 6.0 are all floats. When attempting to mix incompatible data types, you may encounter a **TypeError**. You can always check the data type of something using the *type()* function.

**Implicit vs Explicit Conversion**

As we saw earlier in the video, some data types can be mixed and matched due to implicit conversion. Implicit conversion is where the interpreter helps us out and automatically converts one data type into another, without having to explicitly tell it to do so.

By contrast, explicit conversion is where we manually convert from one data type to another by calling the relevant function for the data type we want to convert to. We used this in our video example when we wanted to print a number alongside some text. Before we could do that, we needed to call the *str()* function to convert the number into a string. Once the number was explicitly converted to a string, we could join it with the rest of our textual string and print the result.

# Study Guide: Expressions and Variables

This study guide provides a quick-reference summary of what you learned in this lesson and serves as a guide for the upcoming practice quiz.

In the Expressions and Variables segment, you learned about expressions, variables, and the data types: string, integer, and float. You learned how to convert a value from one data type to another and you learned how to resolve a few common errors in Python.

# Terms

* **expression** - a combination of numbers, symbols, or other values that produce a result when evaluated
* **data types** - classes of data (e.g., string, int, float, Boolean, etc.), which include the properties and behaviors of instances of the data type (variables)
* **variable** - an instance of a data type class, represented by a unique name within the code, that stores changeable values of the specific data type
* **implicit conversion** - when the Python interpreter automatically converts one data type to another
* **explicit conversion** - when code is written to manually convert one data type to another using a data type conversion function:
  + **str()** - converts a value (often numeric) to a **string** data type
  + **int()** - converts a value (usually a float) to an **integer** data type
  + **float()** - converts a value (usually an integer) to a **float** data type

# Coding skills

**Skill Group 1**

* Use the assignment operator **=** to assign values to variables
* Use basic arithmetic operators with variables to create expressions
* Use explicit conversion to change a data type from float to string

# The following lines assign the variable to the left of the =

# assignment operator with the values and arithmetic expressions

# on the right side of the = assignment operator.

hotel\_room = 100

tax = hotel\_room \* 0.08

total = hotel\_room + tax

room\_guests = 4

share\_per\_person = total/room\_guests

# This line outputs the result of the final calculation stored

# in the variable "share\_per\_person"

print("Each person needs to pay: " + str(share\_per\_person)) # change a data type

**Skill Group 2**

* Output multiple string variables on a single line to form a sentence
* Use the plus (+) connector or a comma to connect strings in a print() function
* Create spaces between variables in  a print() function

# The following 5 lines assign strings to a list of variables.

salutation = "Dr."

first\_name = "Prisha"

middle\_name = "Jai"

last\_name = "Agarwal"

suffix = "Ph.D."

print(salutation + " " + first\_name + " " + middle\_name + " " + last\_name + ", " + suffix)

# The comma as a string ", " adds the conventional use of a comma plus a

# space to separate the last name from the suffix.

# Alternatively, you could use commas in place of the + connector:

print(salutation, first\_name, middle\_name, last\_name,",", suffix)

# However, you will find that this produces a space before a comma within a string.

**Skill Group 3**

* Resolve TypeError caused by a data type mismatch issue
* Use an explicit conversion function

print("5 \* 3 = " + (5\*3))

# Resolution:

# print("5 \* 3 = " + str(5\*3))

#

# To avoid a type error between the string and the integer within the

# print() function, you can make an explicit data type conversion by

# using the str() function to convert the integer to a string.

**Skill Group 4**

* Resolve a ZeroDivisionError caused by an attempt to divide by 0

numerator = 7

denominator = 0   # Possible resolution: Change the denominator value

result = numerator / denominator

print(result)

# One possible assumption for a number divided by zero error might

# include the issue of a null value as a denominator (could happen when

# using a loop to iterate over values in a database). In such cases, the

# desired outcome may be to leave the numerator value intact. The

# numerator value can be preserved by reassigning the denominator with

# the integer value of 1. The result would then equal the numerator.

# Python practice information

For additional Python practice, the following links will take you to several popular online interpreters and codepads:

* [Welcome to Python](https://www.python.org/shell/)
* [Online Python Interpreter](https://www.onlinegdb.com/online_python_interpreter)
* [Create a new Repl](https://repl.it/languages/python3)
* [Online Python-3 Compiler (Interpreter)](https://www.tutorialspoint.com/execute_python3_online.php)
* [Compile Python 3 Online](https://rextester.com/l/python3_online_compiler)
* [Your Python Trinket](https://trinket.io/python3)