Model 0 Integers and Floats

Every value in Python has a ***type*** which determines what can be done with the value. Consider the following statements and expressions that were entered into a Python Shell.

Python code	Shell output
`integer = 3`	
`type(integer)`	<class ′int′>
`type("integer")`	<class ′str′>
`pi = 3.1415`	
`type(pi)`	<class ′float′>
`word = str(pi)`	
`word`	′3.1415′
`number = float(word)`	
`print(word * 2)`	3.14153.1415
`print(number * 2)`	6.283
`print(word + 2)`	TypeError
`print(number + 2)`	5.14159
`euler = 2.7182`	
`int(euler)`	2
`round(euler)`	3

Questions (15 min) **Start time:**

1. What is the value and type (`int`, `float`, or `str`) of the following variables?

Variable	Value of Variable	Type of Value
`integer`		
`word`		
`number`		
`euler`		

2. List the function calls that convert a value to another type.

3. How does the behavior of the operators (+ and *) depend on the data type?

4. What is the difference between the `int` function and the `round` function?

5. What is the value of `3 + 3 + 3`? What is the value of `.3 + .3 + .3`? Enter these expressions into a Python Shell—what do you notice about the results?

6. Based on the previous question:

a) In order to store a number with 100% accuracy, what data type is required?

b) How might you precisely represent a bank account balance of $123.45?

7. Try calculating a very large integer in a Python Shell, for example, 123^{456}. Is there a limit to the integers that Python can handle?

8. Try calculating a very large floating-point number in a Python Shell, for example, 123.0^{465}. Is there a limit to the floating-point numbers that Python can handle?

9. Summarize the difference between the numeric data types (`int` and `float`). What are their pros and cons?