Data Types

Python has a wide variety of built-in types for storing anything from numbers and text (e.g., `int`, `float`, `str`) to common data structures (e.g., `list`, `tuple`).

Manager: Recorder:

Presenter: Reflector:

Content Learning Objectives

After completing this activity, students should be able to:

- Explain how using roles improves the team's success.
- Explain differences between integer and floating-point data.
- Reference a specific element of a sequence by an index.
- Compare and contrast numeric and sequence data types.

Process Skill Goals

During the activity, students should make progress toward:

- Providing feedback on how well other team members are working. (Teamwork)

Model 1 Integers and Floats

Every value in Python has a ***data type*** which determines what can be done with the data. Enter the following code, one line at a time, into a Python Shell. Record the output for each line (if any) in the second column.

Python code	Shell output
`integer = 3`	
`type(integer)`	
`type("integer")`	
`pi = 3.1415`	
`type(pi)`	
`word = str(pi)`	
`word`	
`number = float(word)`	
`print(word * 2)`	
`print(number * 2)`	
`print(word + 2)`	
`print(number + 2)`	
`euler = 2.7182`	
`int(euler)`	
`round(euler)`	

Questions (15 min) Start time:

2. What is the data type (`int`, `float`, or `str`) of the following values? (Note: if you're unsure, use the `type` function in a Python Shell.)

a) pi

b) integer

c) word

d) number

3. List the function calls that convert a value to a new data type.

~~**4**. How does the behavior of the operators (+ and *) depend on the data type?~~

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

6. What is the value of `3 + 3 + 3`? What is the value of `.3 + .3 + .3`? If you enter these expressions into a Python Shell, what do you notice about the results?

7. In order to store a number with 100% accuracy, what data type is required? How might you precisely represent a bank account balance of $123.45?

8. 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?

9. 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?

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

Model 2 Lists

A variable can hold multiple values in the form of a ***list***. The values are separated by commas and wrapped in square brackets. For example:

```
primes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

Each ***element*** of the list can be referenced by an ***index,*** which is the sequential position starting at 0. For example, `primes[4]` is 11.

index	0	1	2	3	4	5	6	7	8	9
value	2	3	5	7	11	13	17	19	23	29

Do not type anything yet! Read the questions first!

Python code	Shell output
`odd = [1, 3, 5, 7]`	
`odd`	
`odd[2]`	
`odd[4]`	
`len(odd)`	
`number = odd[1]`	
`number`	
`odd[1] = 2`	
`odd`	
`number`	

Questions (10 min) Start time:

11. What is the index of the second element of `primes`? What is the value at that index?

12. How does the index number compare to the position of the element?

13. Type each line of code in a Python Shell and write the corresponding output in the space above. If an error occurs, write what type of error. Place an asterisk (*) next to any output for which you were surprised.

14. How did you reference the value of the 3rd element of `odd`?

15. What did the output of the len() function tell you about the list?

16. One of the lines in Model 2 displayed an error. Explain the reason for the error.

17. Write a statement that assigns a list of three integers to the variable `run`.

18. Write a statement that assigns the value 100 to the last element of `run`.

19. Write a statement that assigns the first value of `run` to a variable named `first`.

Model 3 Sequences

Lists and strings are examples of ***sequence*** types. Complete the table below to explore how sequences work.

Python code	Shell output
`seq1 = "one two"`	
`type(seq1)`	
`len(seq1)`	
`seq1[1]`	
`seq1[1] = '1'`	
`seq2 = "one", "two"`	
`type(seq2)`	
`len(seq2)`	
`seq2[1]`	
`seq2[1] = '1'`	
`seq3 = ["one", "two"]`	
`type(seq3)`	
`seq3[1]`	
`seq3[1] = 1`	
`seq4 = ("one", 1)`	
`type(seq4)`	
`number = 12345`	
`number[3]`	

Questions (15 min) Start time:

20. How does a sequence type differ from a number? (See the last row of the table.)

21. What are the names of the three sequence types introduced in Model 3?

22. How does the syntax of creating a tuple differ from creating a list?

23. Is there more than one way (syntax) to create a tuple? Justify your answer.

24. Which sequence types allow their elements to be changed? Which do not?

25. Is it possible to store values of different types in a sequence? If yes, give an example from the table; if no, explain why not.

26. Summarize the difference between lists and tuples. How do they look differently, and how do they work differently?