# Basic Data Structures

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`, `dict`).

## Content Learning Objectives

*After completing this activity, students should be able to:*

- Reference a specific element of a sequence by an index.
- Compare and contrast numeric and sequence data types.
- Create a dictionary of strings and look up values by key.

## Process Skill Goals

*During the activity, students should make progress toward:*

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

# Model 1 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:** _____

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

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

3. 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.

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

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

6. The output of Model 1 displayed an error. Explain the reason for the error.

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

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

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

# Model 2 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: _____

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

11. What are the names of the three sequence types introduced in Model 2?

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

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

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

15. 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.

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

# Model 3 Dictionaries

In Python, a ***dictionary*** stores "`key: value`" pairs. The pairs are separated by commas and wrapped in curly braces. For example:

```
elements = {'C': 'carbon', 'H': 'hydrogen', 'O': 'oxygen', 'N': 'nitrogen'}
```

| Key | Value |
|---|---|
| `'C'` | `'carbon'` |
| `'H'` | `'hydrogen'` |
| `'O'` | `'oxygen'` |
| `'N'` | `'nitrogen'` |

In contrast to sequence types, a dictionary is a ***mapping*** type. Values are referenced by ***keys***, rather than by indexes.

Type the `elements` dictionary above into a Python Shell, and then complete the following table to explore how it works.

| Python code | Shell output |
|---|---|
| `type(elements)` | |
| `elements.keys()` | |
| `elements.values()` | |
| `elements['C']` | |
| `atom = 'N'` | |
| `elements[atom]` | |
| `elements[N]` | |
| `elements['nitrogen']` | |
| `elements[1]` | |
| `len(elements)` | |
| `elements['B'] = 'Boron'` | |
| `elements.items()` | |

## Questions (20 min) Start time: _____

17. List all the keys stored in the `elements` dictionary after completing the table.

18. What is the data type of the keys in the `elements` dictionary?

19. Explain the reason for the error after entering each of the following lines:

a) `elements[N]`

b) `elements['nitrogen']`

c) `elements[1]`

20. Ignoring the "dict_items()" part, describe the contents and type of data returned by the `items()` method.

21. Write a Python expression that creates a dictionary for the seven days of the week, i.e., Sun=1, Mon=2, Tue=3, etc. Assign the dictionary to the variable `dow`.

22. If you assign two different values to the same key (i.e., two assignment statements with one value each), which value is stored in the dictionary? Justify your answer with an example.

23. Another way to store the data in Model 3 is to use two lists:

```
keys = ['C', 'H', 'O', 'N']
vals = ['carbon', 'hydrogen', 'oxygen', 'nitrogen']
```

What is a disadvantage of this approach? Explain your reasoning.