# Business Programming (using Python)

Zhaohu (Jonathan) Fan

Information Technology Management

Scheller College of Business

Georgia Institute of Technology

September 7, 2023

# Main topics

- Go over some of the Severance Chapter 5 concept
- Data structures
  - List
  - Control flow
    - Iteration - Loops

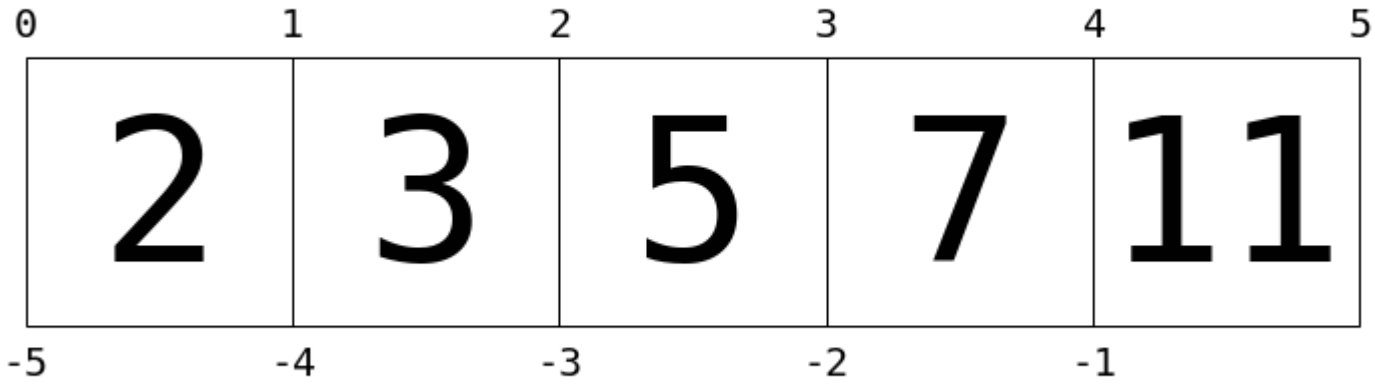
# Data structures

# Built-In data structures

| Type Name | Example | Description |
|---|---|---|
| `list` | `[1, 2, 3]` | Ordered collection |
| `tuple` | `(1, 2, 3)` | Immutable ordered collection |
| `dict` | `{'a':1, 'b':2, 'c':3}` | Unordered (key,value) mapping |
| `set` | `{1, 2, 3}` | Unordered collection of unique values |

# Lists

- Lists are the basic *ordered* and *mutable* data collection type in Python.
  - They can be defined with **comma-separated values** between **square brackets**.
  - For example, here is a list of the first several prime numbers:
  - `L = [2, 3, 5, 7,11]`


0
1
2
3
4
5
2
3
5
7
11
-5
-4
-3
-2
-1

# Questions?

- How can you determine the **length of a list** in Python?
  - How do you **append a value to the end of a list**?
  - How does **addition behave when used with lists**?
  - What does the `sort()` **method** do and how does it **affect** the original list?
  - and more....

# Lists

## Codes & Outputs

```
L = [2, 3, 5, 7, 11]
print(L)
```

```
[2, 3, 5, 7, 11]
```

---

```
[21] # Length of a list
     len(L)
```

```
5
```

```
[24] # Append a value to the end
     L.append(300)
     L
```

```
[2, 3, 5, 7, 11, 11, 300]
```

```
[25] # Addition concatenates lists
     L + [13, 17, 19]
```

```
[2, 3, 5, 7, 11, 11, 300, 13, 17, 19]
```

```
[26] # sort() method sorts in-place
     L = [2, 5, 1, 6, 3, 4]
     L.sort()
     L
```

```
[1, 2, 3, 4, 5, 6]
```

- **Syntax**
  - A list can be defined with **comma-separated values** between **square brackets**.
  - `len`
    - Length of a list
  - `.append`
    - Add an item to the end of the list.
  - `+`
    - Addition concatenates lists
  - `.sort()`
    - Sort the items of the list in place
    - `.sort(*, key=None, reverse=False)`
- More on Lists (with a link).

# Lists

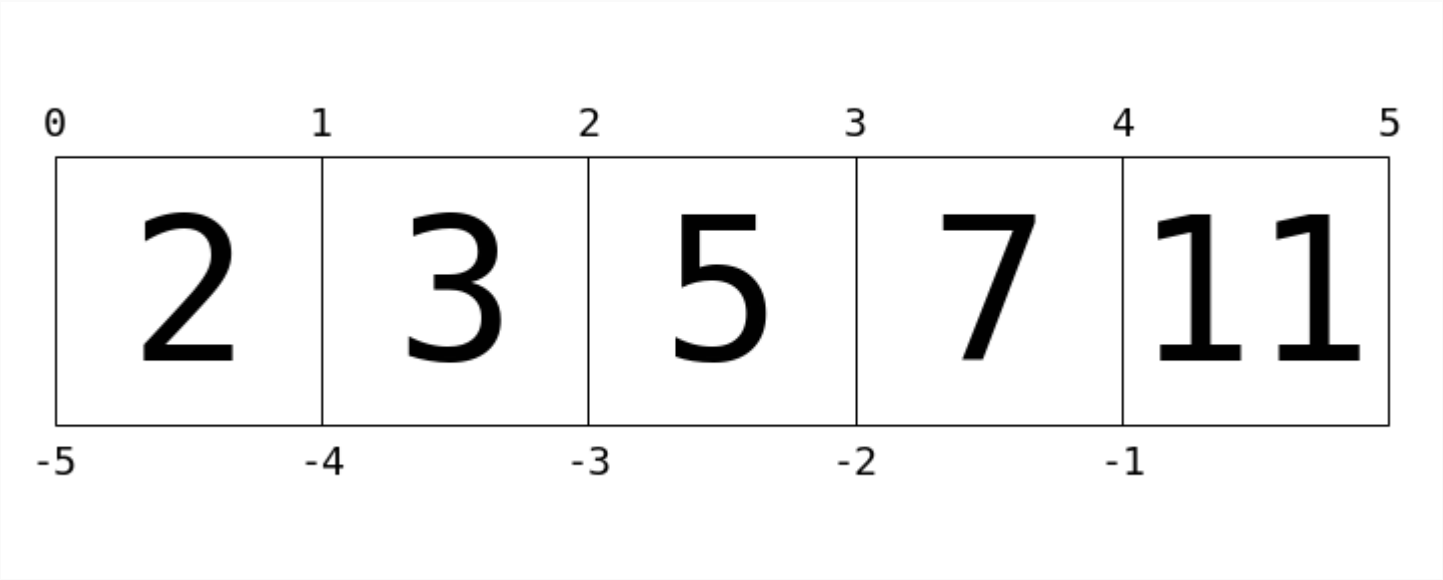
- Lists in Python can contain objects of ***any*** type or even a **mix** of types.
  - They are not restricted to a single type.
  - **Examples**:

```
[38] L = [1, 'two', 3.14, [0, 3, 5]]
     L
```

```
[1, 'two', 3.14, [0, 3, 5]]
```

```
GPAs = ['John', 3.3, 'Sally', 2.2, 'Bernis', 3.8, 'Fred', 3.2, 'Victoria', 3.4, 'Valerie', 2.6, 'Eric', 2.6]
print(GPAs)
```

```
['John', 3.3, 'Sally', 2.2, 'Bernis', 3.8, 'Fred', 3.2, 'Victoria', 3.4, 'Valerie', 2.6, 'Eric', 2.6]
```


0
1
2
3
4
5
2
3
5
7
11
-5
-4
-3
-2
-1

# List indexing and slicing

- Where **indexing** is a means of fetching a single value from the list, **slicing** is a means of accessing multiple values in sub-lists.
- It uses a `colon` to indicate the start point (inclusive) and end point (non-inclusive) of the sub-array.

# List indexing and slicing

### Codes & Outputs

```
[41] L = [2, 3, 5, 7, 11]
     L
```

```
[2, 3, 5, 7, 11]
```

```
[42] L[0]
```

```
2
```

```
L[1]
```

```
3
```

```
[44] L[-1]
```

```
11
```

```
[45] L[-2]
```

```
7
```

- **Comments**
  - Python uses **zero-based indexing**.
  - Access the first element in using the following syntax `L[0]`
  - `L[1]` returns `3`, because that is the next value at index `1`.
  - Elements at the end of the list can be accessed with negative numbers, starting from -1
    - using the following syntax `L[-1]`

# List indexing and slicing

### Codes & Outputs

```
L = [2, 3, 5, 7, 11]
L
```

```
[2, 3, 5, 7, 11]
```

```
[47] L[0:3]
```

```
[2, 3, 5]
```

```
[49] L[:3]
```

```
[2, 3, 5]
```

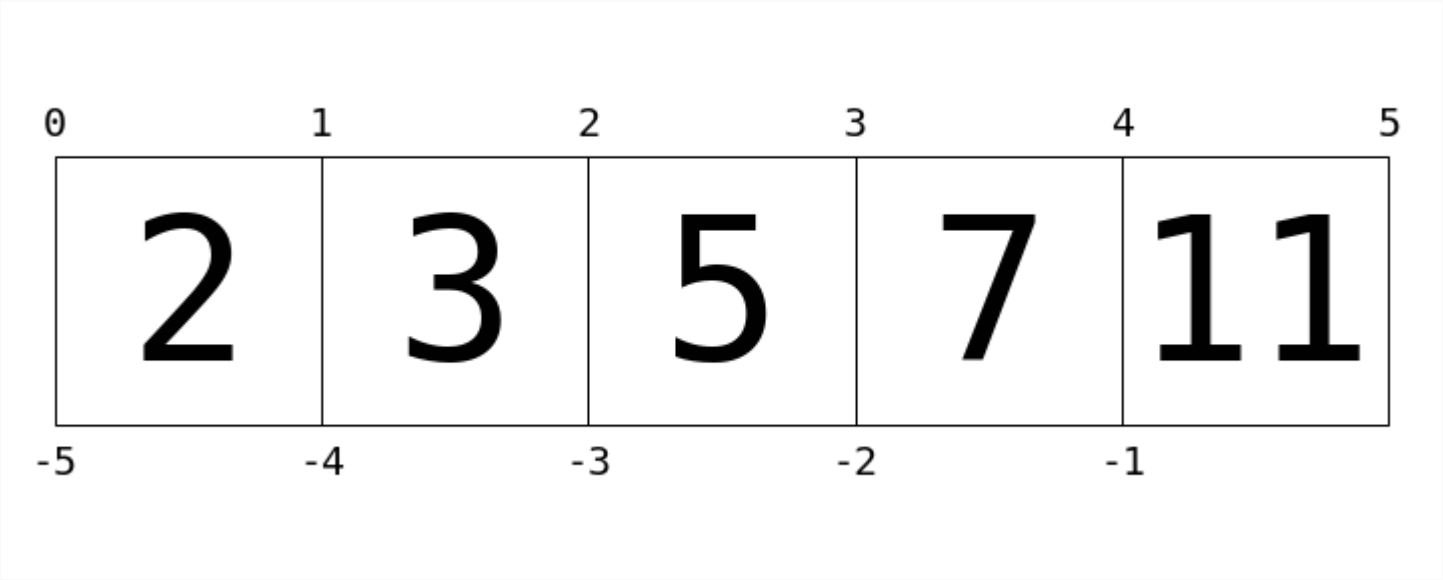
```
[50] L[-3:]
```

```
[5, 7, 11]
```

```
[51] L[::2]  # equivalent to L[0:len(L):2]
```

```
[2, 5, 11]
```

- **Comments**
  - Access the first three elements of the list in using the following syntax `L[0:3]`
  - **Slice** takes just the values between the indices. If we leave out the first index, `0` is assumed.
  - Access the last three elements of the list in using the following syntax `L[-3:]`
  - Specify a third integer that represents the **step size**; for example, to select every second element of the list
    - using the following syntax `L[::2]`


0
1
2
3
4
5
2
3
5
7
11
-5
-4
-3
-2
-1

# List indexing and slicing

**Codes & Outputs**

```
L[::-1]
```

```
[11, 7, 5, 3, 2]
```

```
[53] L[0] = 100
     print(L)
```

```
[100, 3, 5, 7, 11]
```

```
[54] L[1:3] = [55, 56]
     print(L)
```

```
[100, 55, 56, 7, 11]
```

- **Comments**
  - Specify a **negative step** , which will reverse the array: `L[::-1]`.
  - **Both indexing and slicing** can be used to set elements as well as access them.

# Revisit control flow

- `for` loops
  - `while` loops
  - Download the lab notes from the Canvas page ([L5] lab notes.ipynb)
  - Sample solution