빅데이터 혁신공유대학

파이썬으로 배우는 기계학습

한동대학교 전산전자공학부

김영섭 교수

Data Structures in Python
Chapter 1 - 1

- Introduction - Review Python
- **Objects and References**
- List Operations
- GitHub & Jupyter-Lab
- Markdown Tutorial

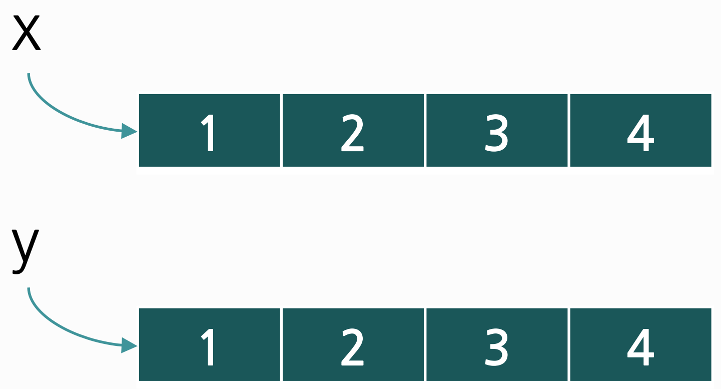
너는 청년의 때에 너의 창조주를 기억하라 곧 곤고한 날이 이르기 전에, 나는 아무 낙이 없다고 할 해들이 가깝기 전에 (전12:1)

Agenda

- Topics:
  - **Objects and References**
    - **Objects in memory**
    - **References**
    - **Equality**
    - **Mutability vs. Immutability**
  - List Operations
    - List operations (methods)
    - Shallow copy vs. Deep copy
- References:
  - DSpy: Chapter 1: Python Review
  - Problem Solving with Algorithms and Data Structures using Python
    - Chapter 1

Objects in memory

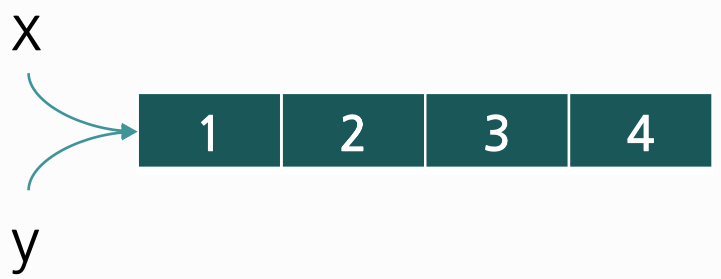
- Value equality



Two different objects that store the same information.

```
x = [1, 2, 3, 4]
y = [1, 2, 3, 4]
```

- Reference equality



Two different references (or names) for the same object.

```
x = [1, 2, 3, 4]
y = x
```

Different ways to compare equality

- ==
  - Calls a method of the object
  - Typically involves checking the contents of the objects.
  - We should always use this for literals.

- is
  - Checks the references of the objects.
  - Evaluates to True if they are the same object.

```
x = [1, 2, 3, 4]
y = [1, 2, 3, 4]
print(x == y)
print(x is y)
```

```
x = [1, 2, 3, 4]
y = x
print(x == y)
print(x is y)
```

String

- Every **UNIQUE string** you create will have it's own address space in memory

```
a = 'foo'
b = 'foo'
print(id(a))
print(id(b))
print(a == b)
print(a is b)
```

```
2026110321456
2026110321456
```

immutable object

```
x = [1, 2, 3, 4]
y = [1, 2, 3, 4]
print(id(x))
print(id(y))
print(x == y)
print(x is y)
```

```
2026159683136
2026159685184
```

mutable object

Mutable and Immutable objects

- An immutable object is an object whose state cannot be modified after it is created.

- Examples of **immutable** objects:
  - **integer, boolean, float, string, tuple**

- Examples of **mutable** objects
  - **lists, dictionaries, sets,** most data structures studied in this course

```python
a = 'hello'
b = 'hello'
print(id(a))
print(id(b))
```

```
2026159684288
```

```python
a = 'hello'
print(id(a))
a = 'jello'
print(id(b))
```

```
2026159684288
```

Lists are mutable

- Lists are **mutable**
  - i.e. We can change lists in place, such as reassignment of a sequence slice, which will work for lists, but raise an error for tuples and strings.

- Example:
  - rgb = ['red', 'green', 'blue']
  - rgb[0] = 'RED'
  - rgb still points to the same memory when you are done.

```
rgb = ['red', 'green', 'blue']
print(id(rgb))
rgb[0] = 'RED'
print(id(rgb))
print(rgb)
```

```
2026159684288
```

Tuples are immutable

- Strings and tuples are immutable sequence types: such objects cannot be modified **once created.**
  - i.e. you can’t change a tuple.

- Example:

```
rgb = ('red', 'green', 'blue')
rgb[0] = 'RED'
```

```
TypeError: 'tuple' object does not
support item assignment
```

- The immutability of tuples means they are **faster** than lists.

Operations on Strings

- Whenever you call a method of an object, make sure you know if **changes** the contents of the object or **returns** a new object.

- Example:

```
truth = 'Sola Gratia'
print(id(truth))
truth = 'Sola Fide'
print(id(truth))
```

```
2026160287536

2026160287088
```

a new String object is instantiated and given the data “Sola Gratia" during its construction.

- lower(), upper(), lstrip(), rstrip(), …
  - Return a new copy of the string

```
truth = 'Sola Gratia'
print(id(truth))
facts = truth.upper()
print(id(facts))
```

returns a new object.

Summary

- Variables store references to the objects, not the actual objects.
  - When you assign a variable, **a reference is copied**, not the object. Even it creates a new object and assigns its new reference to it in case of an immutable object.
- There are two kinds of equality.
  - Equality of content (value equality) can be tested with **==**
  - Equality of identity (reference equality) can be tested with is

Data Structures in Python Chapter 1 - 1

- Introduction - Review Python
- **Objects and References**
- List Operations
- GitHub & Jupyter-Lab
- Markdown Tutorial