빅 데 이 터 혁 신 공 유 대 학

파이썬으로 배우는 기계학습

한동대학교 전산전자공학부

김영섭 교수

교육부 NRF 한국연구재단

Data Structures in Python
Chapter 1 - 1

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

Prof. Youngsup Kim, idebtor@gmail.com, CSEE Dept., Grace School Rm204 Handong Global University

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

Operations on Lists - append vs. extend

- extend() - extends the list by appending all the items in the given list (i.e. the argument is a list

```
x = [1, 2, 3]
x.extend([4, 5, 6])
print(x)
```

- append() - adds an item to the end of the lis.

```
x = [1, 2, 3]
x.append([4, 5, 6])
print(x)
```

Operations on Lists - Reversing a list

- reverse() - reverses the list **in place** or alters the content of the list.

```
x = [1, 2, 3]
y = x
x.reverse()
print(x)
```

```
[3, 2, 1]
```

- sort() - sorts the list **in place** or alters the content of the list.

```
x = [1, 2, 3, -1]
x.sort()
print(x)
```

```
[-1, 1, 2, 3]
```

Exercise 1

- What is the output of the following code fragment? Why?

```
p = [1, 2, 3]
print (p[::-1])
print (p)
```

Aliases

- Two references to the same object are known as aliases.

```
x = [1, 2, 3, 4]
y = x
x.append(5)
print(x)
print(y)
```

- When an assignment is performed, **the reference** to the object on the right of the assignment is assigned to the variable on the left.

- When a method of an object is called,
  it sometimes **returns a value** and sometimes it **alters the object**.

Example

- What happens in the following cases? What is the output?

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

x += [4]
```

alters the object


```
print(x)
print(y)
```

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

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

x = x + [4]
```

returns a new object;
x is replaced


```
print(x)
print(y)
```

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

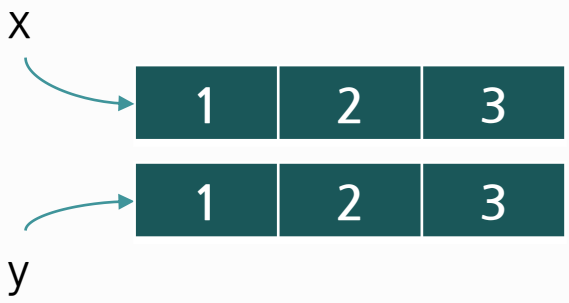
Shallow copy

- Lists and dictionaries have a **copy()** method

```
x = [1, 2, 3]
y = x.copy()

print( x == y )
print( x is y )
```

```
True
False
```




```
a = [ [11], [22], [33] ]
b = a.copy()

print( a == b )
print( a is b )
print( a[0] is b[0])
```

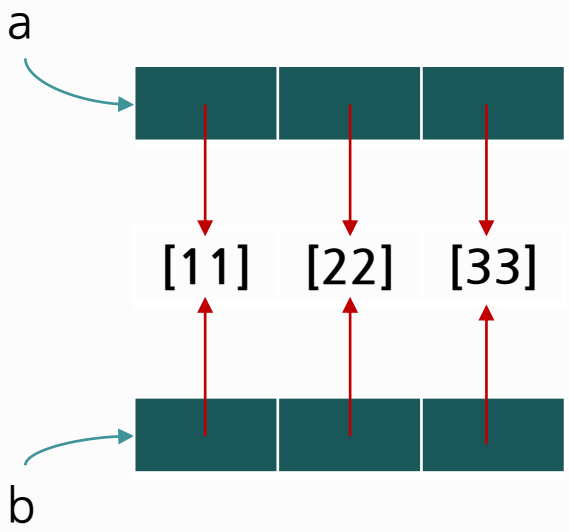
```
True
False
True
```

What does it mean?

a and b are different objects, but
a[0] and b[0] are referencing the same object.

Shallow copy

- New object created
  - Contents of the original object are copied
  - If the contents are references, then the *references* are copied




```
a = [ [11], [22], [33] ]
b = a.copy()

print( a == b )
print( a is b )
print( a[0] is b[0])
```

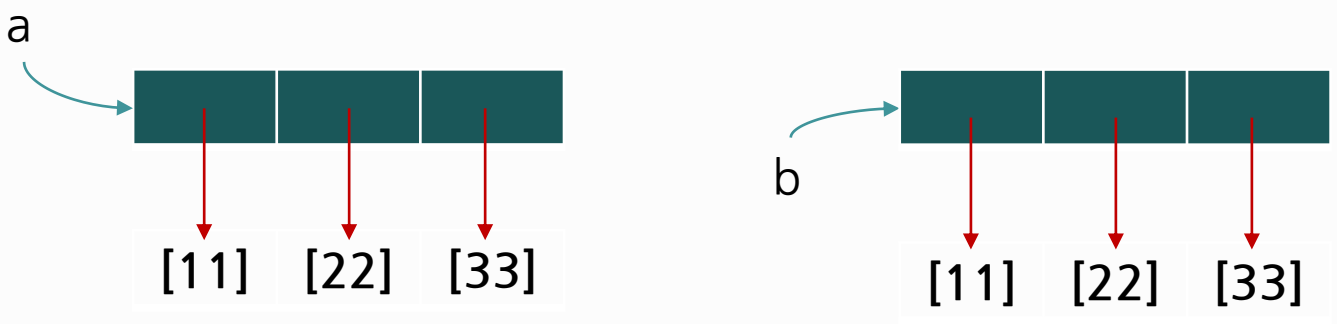
```
True
False
True
```

What does it mean?

a and b are different objects, but
a[0] and b[0] are referencing the same object.

Deep copy

- New object created
  - Contents of the original object are copied
  - If the contents are references, then **the copy the objects referred to** are copied




```
import copy

a = [ [11], [22], [33] ]
b = copy.deepcopy(a)

print( a == b )
print( a is b )
print( a[0] is b[0] )
```

```
True
False
False
```

b[0] has its own copy of the 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**
- When a copy is created, it can be a shallow or deep copy.
  - A shallow copy copies the references.
  - A deep copy recursively copies the objects referred to.
- Lists slower but more powerful then tuples.
  - Lists can be modified and have lots of handy operations and methods.
  - Tuples are immutable and have fewer features.
- To convert between tuples and lists use the **list()** and **tuple()** function.

Data Structures in Python
Chapter 1 - 1

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

Prof. Youngsup Kim, idebtor@gmail.com, CSEE Dept., Grace School Rm204 Handong Global University