파이썬으로 배우는 데이터 구조

한동대학교
전산전자공학부
김영섭 교수

학습 목표

list자료형의 다양한 메소드를 익히고

shallow copy와 deep copy의 차이점을 알 수 있다

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

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 list.

```
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]
```

x

| 1 | 2 | 3 |

y = x

X

| 1 | 2 | 3 |

x.reverse()

X

| 3 | 2 | 1 |

y

y

- sort() - sorts the list **in place** or alters the content of the list.
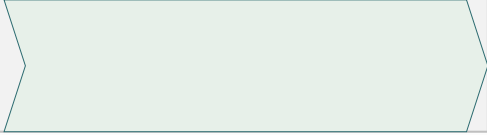
```
x = [1, 2, 3, -1]
x.sort()
print(x)          [-1, 1, 2, 3]
```

Exercise 1

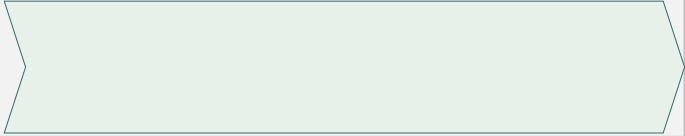- What is the output of the following code fragment? Why?

```
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)
```

X

| 1 | 2 | 3 |

y

X

| 1 | 2 | 3 | 4 |

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)
```
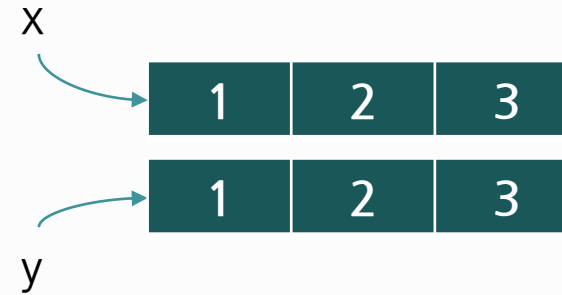
X

| 1 | 2 | 3 |

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 )      True
print( x is y )      False
```

x

| 1 | 2 | 3 |

| 1 | 2 | 3 |

y

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

print( a == b )      True
print( a is b )      False    What does it mean?
print( a[0] is b[0]) True
```

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
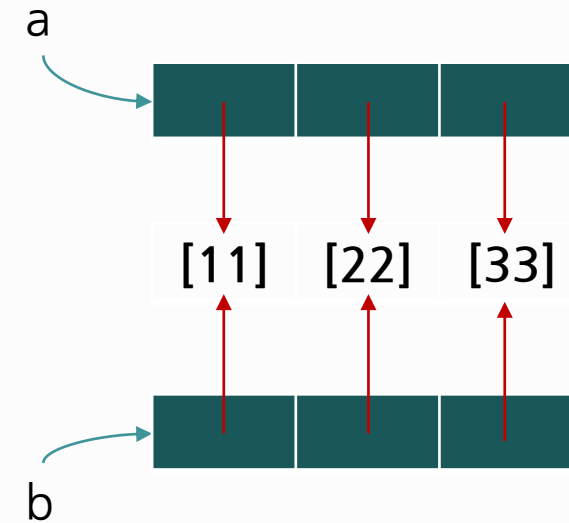 - If the contents are references, then the *references* are copied

a

[11]   [22]   [33]

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

print( a == b )        True
print( a is b )        False    What does it mean?
print( a[0] is b[0])   True
```

b

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

a

[11]   [22]   [33]

b

[11]   [22]   [33]

```
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 than 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.

학습 정리

1) 내장 함수로는 len(), max(), sum(), sort() 등이 있고,
   각 객체는 다양한 메소드(method)들을 제공한다

2) 깊은 복사(deep copy)는 복제된 새 객체와 새 reference를
   반환한다

3) 얕은 복사(shallow copy)는 새 reference를 반환하지만
   같은 객체를 가리킨다