웹프로그래밍의 기초

Week4

Python data structure #1

List, Tuple

Python Collections (Arrays)

- There are four collection data types in the Python programming language:
  - List is a collection which is ordered and changeable. Allows duplicate members.
  - Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
  - Set is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
    - Set items are unchangeable, but you can remove and/or add items whenever you like.
  - Dictionary is a collection which is ordered** and changeable. No duplicate members.
    - As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered.

List

List - introduction

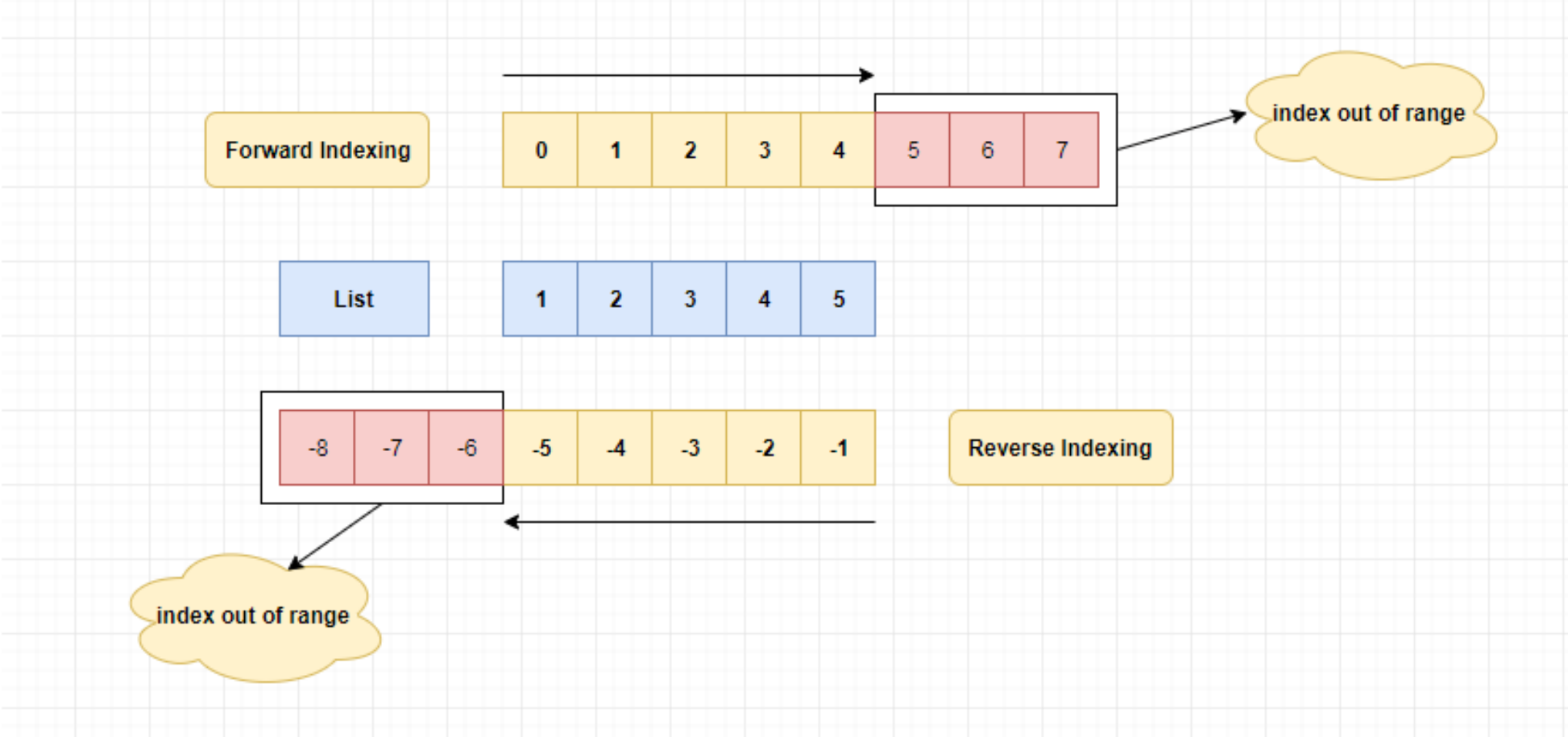
- Lists are used to store multiple items in a single variable.
- Lists are created using square brackets.

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

List – index

- List can be accessed via index.



List – access List items

- List items are indexed and you can access them by referring to the index number.
  - [0] refers to the first item, [1] refers to the second item.
  - [-1] refers to the last item, [-2] refers to the second last item etc.
  - You can specify a range of indexes by specifying where to start and where to end the range.

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[5:])
```

```
['melon', 'mango']
```

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[:4])
```

```
['apple', 'banana', 'cherry', 'orange']
```

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

```
['cherry', 'orange', 'kiwi']
```

List – List item as a variable

- Each item of List can be used as an independent variable.

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
email = "You are currently viewing bicycles of " + bicycles[2].title() + "."
print(email)
```

```
You are currently viewing bicycles of Redline.
```

List – duplicable, ordered List items

- Duplicate items can be exist in the same list.

```
motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati']
print(motorcycles)

motorcycles[0] = 'ducati'
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki', 'ducati']
['ducati', 'yamaha', 'suzuki', 'ducati']
```

- The items have a defined order, but here are some list methods that will change the order.
  - Temporary

```
motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati']
print(sorted(motorcycles))
print(sorted(motorcycles, reverse=True))
print(motorcycles)
```

```
['ducati', 'honda', 'suzuki', 'yamaha']
['yamaha', 'suzuki', 'honda', 'ducati']
['honda', 'yamaha', 'suzuki', 'ducati']
```

Permanent

```
motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati']
print(motorcycles)
motorcycles.sort()
print(motorcycles)
motorcycles.reverse()
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki', 'ducati']
['ducati', 'honda', 'suzuki', 'yamaha']
['yamaha', 'suzuki', 'honda', 'ducati']
```

List – add, insert List items

- To insert a list item at a specified index, use the insert() method.
- To add an item to the end of the list, use the append() method:

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

```
['apple', 'banana', 'cherry', 'orange']
```

```
thislist.insert(1, "orange")
print(thislist)
```

```
['apple', 'orange', 'banana', 'cherry', 'orange']
```

List – change List items

- To change the value of a specific item, refer to the index number.

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

```
['apple', 'blackcurrant', 'cherry']
```

- To change the value of items within a specific range, define a list with the new values, and refer to the range of index numbers where you want to insert the new values:

```
thislist = ["apple", "banana", "cherry"]
thislist[1:2] = ["blackcurrant", "watermelon"]
print(thislist)
```

```
thislist = ["apple", "banana", "cherry"]

thislist[1:2] = ["blackcurrant", "watermelon"]
print(thislist)
```

```
['apple', 'blackcurrant', 'watermelon', 'cherry']
```

List – remove List items 1/2

- The remove() method removes the specified item.

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

```
['apple', 'cherry']
```

- The pop() method removes the specified index. If you do not specify the index, the pop() method removes the last item.

```
thislist = ["apple", "banana", "cherry"]
print(thislist.pop(1))
print(thislist)
```

```
banana
['apple', 'cherry']
```

```
thislist = ["apple", "banana", "cherry"]
print(thislist.pop())
print(thislist)
```

```
cherry
['apple', 'banana']
```

List – remove List items 2/2

- The del keyword also removes the specified index. If you do not specify the index,it will delete the list completely.

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

```
['banana', 'cherry']
```

```
thislist = ["apple", "banana", "cherry"]
del thislist
print(thislist)
```

```
Traceback (most recent call last):
  File "./prog.py", line 3, in <module>
NameError: name 'thislist' is not defined
```

- The clear() method empties the list. The list still remains, but it has no content.

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

```
[]
```

List – loop the list

- If you need to do something for every list items, you can loop through the list items by using a for loop using FOR statement.

```
students = ["scott", "john", "alice"]

for student in students:
        print(student)
```

```
scott
john
alice
```

List – loop the list with extra tasks

- You can add any task you may want to execute within each execution of FOR statement.

```
students = ["scott", "john", "alice"]

i = 1

for student in students:
        print(str(i) +". " + student)
        i += 1
```

```
1. scott
2. john
3. alice
```

```
colors = ["Red", "Green", "Blue"]
nums = [1, 2, 3]

for color in colors:
   for num in nums:
         print(color, num)
```

```
Red 1
Red 2
Red 3
Green 1
Green 2
Green 3
Blue 1
Blue 2
Blue 3
```

List – loop the list, after its completion

- The first line of the for loop must end with a colon, and the body must be indented.
- The colon at the end of the first line signals the start of a block of statements.

```
students = ["scott", "john", "alice"]
i = 1

for student in students:
        print(str(i) +". " + student)
        i += 1

print("As of today, we have " + str(len(students)) + " student(s).")
```

```
1. scott
2. john
3. alice
As of today, we have 3 student(s).
```

List – loop the list

- The shape of indentation makes logical difference without generating an error.

```
students = ["scott", "john", "alice"]
i = 1

for student in students:
        print(str(i) +". " + student)
        i += 1
        print(str(i) +". " + student)
```

```
1. scott
2. scott
2. john
3. john
3. alice
4. alice
>
```

```
students = ["scott", "john", "alice"]
i = 1

for student in students:
        print(str(i) +". " + student)
        i += 1
print(str(i) +". " + student)
```

```
1. scott
2. john
3. alice
4. alice
>
```

List - Concatenation of Lists

- Using + operator, we can concatenate two or more list.

```
numbers = [1, 2, 3, 4, 5]
print("Original numbers are ", numbers)

numbers = numbers + [6, 7, 8] + [9, 10]
print("After concatenation the numbers are", numbers)# Online Python compiler
(interpreter) to run Python online.

Original numbers are  [1, 2, 3, 4, 5]
After concatenation the numbers are [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>
```

Tuple

Tuple - definition

- A tuple is a collection which is ordered and unchangeable.
- Tuples are written with round brackets.

```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
```

```
('apple', 'banana', 'cherry', 'apple', 'cherry')>
```

- You can access tuple items by referring to the index number, and navigating with -1 for the last item, -2 for the second last item etc.

```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple[3])
```

```
apple
```

Tuple – How to change the immutable

- Add, change, and remove items: You can convert the tuple into a list, manipulate the list, and convert the list back into a tuple.

```
thistuple = ("apple", "banana", "cherry")
thattuple = list(thistuple)
thattuple.append("orange")
thistuple = tuple(thattuple)
print(thistuple)
```

```
('apple', 'banana', 'cherry', 'orange')
```

Tuple – packing and unpacking

- When we create a tuple, we normally assign values to it. This is called "packing" a tuple.
- we are also allowed to extract the values back into variables. This is called "unpacking“

```
fruits = ("apple", "banana", "cherry")

(container1, container2, container3) = fruits

print(container1)
print(container2)
print(container3)
```

```
apple
banana
cherry
```