Python List

01204111 Section 1

What is list

- A list is a datatype that can contains multiple values inside in an ordered sequence.
- Items in list are separated with comma (,) and enclosed within square brackets [ ].

```
my_list = [1, 2, 3, 4]
print(my_list)
```

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

Item in a list

- Item in a list can be any data type, even though another list.

```
prime_number = [2, 3, 5, 7, 11]
animal = ['cat', 'dog', 'mouse']
couple = [['Jack', 'Rose'], ['Edward', 'Bella'], 123]
today = [30, '30 Aug 2017', 'Wednesday']
```

Index

- You can access item in list by using index which enclosed within a pair of square brackets [ ].
- Index is integer number beginning from zero (0).

```
animal = ['cat', 'dog', 'mouse']
print(animal[0])
```

```
cat
```

Index

- Index can be a negative number.
  - It can access list backward.

```
animal = ['cat', 'dog', 'mouse']
print(animal[-1])
print(animal[-2])
print(animal[-3])
print(animal[-4])
```

```
mouse
dog
cat
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

Index

- Item in list can be changed by using index and assignment operator.

```
animal = ['cat', 'dog', 'mouse']
animal[2] = 'rabbit'
print(animal)
```

```
['cat', 'dog', 'rabbit']
```

- Assigning list item on out-of-range index causes an error.

```
animal = ['cat', 'dog', 'mouse']
animal[5] = 'horse'
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in
<module>
IndexError: list assignment index
out of range
```

Slice

- Index can get only single value from list.
- Slice can get several values from list and form into a new list.

```
prime_number = [2, 3, 5, 7, 11]
my_prime = prime_number[1:4]
print(my_prime)
```

```
[3, 5, 7]
```

Slice

- Slice consists of three numbers separated with Colon sign (:) and enclosed within square bracket [ ].
- Last Colon sign and number are optional.
- Each number can be empty because there are default values.

[a : b : c]

a = start index, the first item of new list will be value at a index, *default value = 0*
b = end index, the last item of new list will be value at (b – 1) index, *default value = length of list + 1*
c = step size, each item in new list will be value at (a + nc) index when n is 0, 1, 2, 3, ...,default value = 1

Slice

```
>>> prime_number = [2, 3, 5, 7, 11]
>>> prime_number[1:4]
[3, 5, 7]
>>> prime_number[2:]
[5, 7, 11]
>>> prime_number[:3]
[2, 3, 5]
>>> prime_number[::2]
[2, 5, 11]
>>> print_number[::-1]
[11, 7, 5, 3, 2]
```

Length of list

- *len( )* function will return the number of values inside list.

```
animal = ['cat', 'dog', 'mouse']
print(len(animal))
```

```
3
```

Finding item in list

- You can find item in list by using *index( )* function.
- *index( )* function will return index of the passed value.

```
animal = ['cat', 'dog', 'mouse']
dog_index = animal.index('dog')
print(dog_index)
```

```
1
```

```
animal = ['a', 'b', 'c', 'a']
a_index = animal.index('a')
print(a_index)
```

```
0
```

Return only first index

Finding item in list

- *index( )* will be error if there is no passed value inside list.

```
animal = ['cat', 'dog', 'mouse']
animal.index('horse')
```

```
Traceback (most recent call
last):
  File "<stdin>", line 1, in
<module>
ValueError: "horse" is not in
list
```

Finding item in list

- You can prevent error of *index( )* by using *in* operation.
- *in* operation will return *True* if value on the left side is in list on the right side.

```
prime_number = [2, 3, 5, 7, 9]
print(5 in prime_number)
print(4 in prime_number)
```

```
True
False
```

Finding item in list

- Using *if* statement with *in* operator can prevent error of *index( )*.

```
animal = ['cat', 'dog', 'mouse']
if 'horse' in animal:
    horse_index = animal.index('horse')
    # do something
else:
    print('There is no horse')
```

Add value into list

- *append( )* function is used for add passed value into the end of list.

```
member = ['Jack', 'John', 'Jim']
member.append('Rose')
print(member)
```

```
['Jack', 'John', 'Jim', 'Rose']
```

Add value into list

- *insert( )* function is similar to *append( )* but it can add value at any index in list.
- *insert( )* takes 2 arguments. The first is the index, and the second is the new value.

```
member = ['Jack', 'John', 'Jim']
member.insert(1, 'Rose')
print(member)
```

```
['Jack', 'Rose', 'John', 'Jim']
```

Remove item from list

- *remove( )* function will remove passed value from list.

```
member = ['Jack', 'John', 'Jim']
member.remove('Jim')
print(member)
```

```
['Jack', 'John']
```

Remove item from list

- If value appears multiple times, *remove( )* will remove only the first one

```
solar_system = ['Saturn', 'Pluto', 'Earth', 'Mars', 'Pluto']
solar_system.remove('Pluto')
print(solar_system)
```

```
['Saturn', 'Earth', 'Mars', 'Pluto']
```

Remove item from list

- You can remove item at any index by using *del* operator.

```python
member = ['Jack', 'John', 'Jim']
del member[0]
print(member)
del member[-1]
print(member)
```

```
['John', 'Jim']

['John']
```

Concatenation

- The operation + can use for concatenate lists.

```
a = [1, 2, 3] + [4, 7, 9]
print(a)
```

```
[1, 2, 3, 4, 7, 9]
```

```
a = [1, 2, 3]
b = [4, 7, 9]
print(a+b+a)
```

```
[1, 2, 3, 4, 7, 9, 1, 2, 3]
```

```
a = [1, 2, 3]
print(sum(a))
```

```
6
```

Loops with Lists

- A **for** loop repeats the code block once for each value in a list or list-like value.

```
a_list = [1, 2, 3, 4, 7, 9]
for a_value in a_list :
        print(a_value, end=' ')
```

```
1 2 3 4 7 9
```

```
for a_value in range(0,20,3) :
        print(a_value, end=' ')
```

```
0 3 6 9 12 15 18
```

Task: *multiply until Zero*

- Write a program that add the value in a list until input zero then show the multiplication of every value in a list.

```
Input: 7
Input: 2
Input: 18
Input: 4
Input: 10
Input: 0
7*2*18*4*10=10080
```

Task: *multiply until Zero*

```python
vals = []
val = int(input('Input: '))
while val != 0:
      vals.append(val)
      val = int(input('Input: '))
mul = 1; first = True
for tmp in vals:
      mul *= tmp
      if first:
            print(tmp, end='')
            first = False
      else:
            print('*%d'%tmp, end='')
print('=%d'%mul)
```

Task: *unique and find frequency*

- Write a program that remove the duplicate items in a list and show the frequency of every item in a list.

```
Input: [2,5,3,0,2,4,4,4,3,7,5]
2:2
5:2
3:2
0:1
4:3
7:1
```

Task: *unique and find frequncy*

```
def make_uniq(vals):
	uniq_vals = []
	for tmp in vals:
		if not tmp in uniq_vals :
			uniq_vals.append(tmp)
	return uniq_vals
```

```
vals = [2,5,3,0,2,4,4,4,3,7,5]
uniq_vals = make_uniq(vals)
freqs = []
for tmp1 in uniq_vals:
	cnt = 0
	for tmp2 in vals:
		if tmp1 == tmp2:
			cnt += 1
	freqs.append(cnt)
for i in range(0,len(uniq_vals)):
	print('%d:%d'%(uniq_vals[i],freqs[i]))
```

Task: *selection sort*

- The selection sort divides the input list into two parts: the sublist of items already sorted, which is built up from left to right at the front of the list, and the sublist of items remaining to be sorted that occupy the rest of the list.

```
Input: [8,5,2,6,3,1,4,7]
Output: [1,2,3,4,5,6,7,8]
```

8	5	2	6	3	1	4	7
7	5	2	6	3	1	4	8
4	5	2	6	3	1	7	8
4	5	2	1	3	6	7	8
4	3	2	1	5	6	7	8
1	3	2	4	5	6	7	8
1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8

Task: *selection sort*

```
vals = [8,5,2,6,3,1,4,7]
print('Input:',vals)
scope = len(vals)
while scope > 0 :
      maxIndex = getMaxIndex(vals[:scope])
      tmp = vals[maxIndex]
      vals[maxIndex] = vals[scope-1]
      vals[scope-1] = tmp
      scope -= 1
print('Output:',vals)
```

```
def getMaxIndex(vals):
      i = j = 0
      while j < len(vals):
            if vals[i] < vals[j]:
                  i = j
            j += 1
      return i
```

Sort

- You can sort item at any index by using *sort* operator.

```
member = [4,6,3,2,0]
print(sorted(member))
member.sort()
print(member)
```

```
[0, 2, 3, 4, 6]
[0, 2, 3, 4, 6]
```

Sort

- You can also reverse sort item at any index too.

```
member = [4,6,3,2,0]
print(sorted(member,reverse=True))
member.sort(reverse=True)
print(member)
```

```
[6, 4, 3, 2, 0]
[6, 4, 3, 2, 0]
```

Sorting String

- You can sort item at any index by using *sort* operator.

```python
member = ['Jack', 'Tom', 'Jim']
print(sorted(member))
```

```
['Jack', 'Jim', 'Tom']
```

```python
member = ['a', 'z', 'A', 'Z']
print(sorted(member))
```

```
['A', 'Z', 'a', 'z']
```

```python
member = ['a', 'z', 'A', 'Z']
member.sort(key=str.lower)
print(member)
```

```
['a', 'A', 'z', 'Z']
```

Multiple assignment Trick

- **The multiple assignment trick** is a shortcut that let you assign multiple variables with the values in a list in one line of code.

```
x,y,z = ['Jack', 'Tom', 'Jim']
print(z, x, y)
```

```
Jim Jack Tom
```

Expression in List

- List comprehensions can contain complex expressions and nested functions.

```
print([i for i in range(0,10)])
from math import pi
print([str(round(pi,i)) for i in range(1, 6)])
vec = [-4, -2, 0, 2, 4]
print([abs(x) for x in vec])
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
['3.1', '3.14', '3.142', '3.1416', '3.14159']
[4, 2, 0, 2, 4]
```

String

- Strings and lists are actually similar. If you consider a string to be a list of single text characters.

```python
text = 'Computer and Programing'
print(text[0])
print(text[-2])
print(text[0:4])
print(len(text))
print('Program' in text)
print('ComPro' not in text)
for i in text[13:] :
        print(i, end='|')
```

```
C
n
Comp
23
True
True
P|r|o|g|r|a|m|i|n|g|
```

Task: *How many Strings?*

- Write a program to count the number of strings where the string length is 2 or more and the first and the last character are same from a given list of strings.

Task: *How many Strings?*

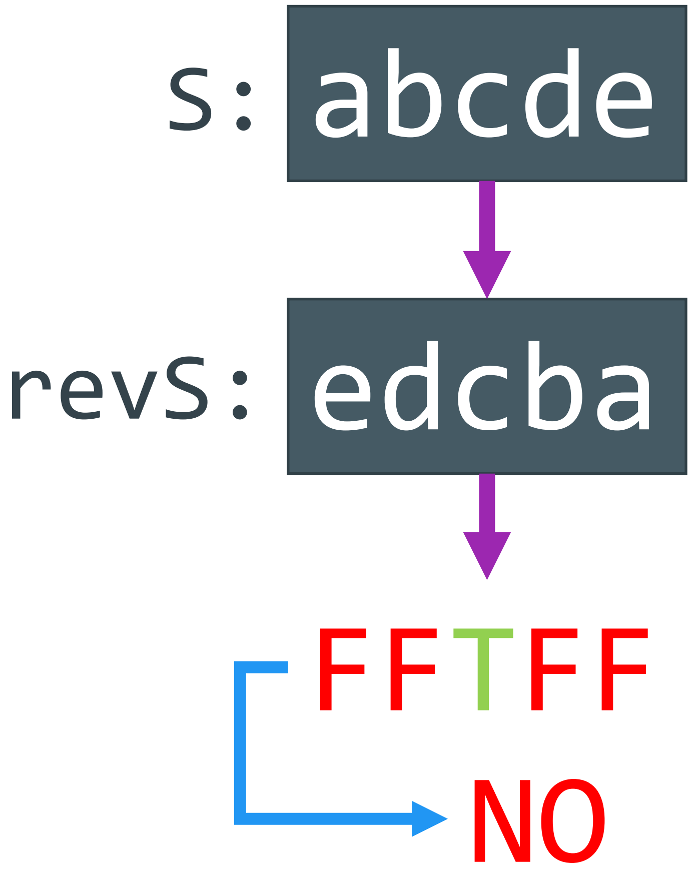
```
samples = ['abc', 'xyz', 'aba', '1221']
cnt = 0
for tmp in samples:
        if len(tmp) :
                if tmp[0] == tmp[len(tmp)-1] :
                        cnt += 1
print(cnt)
```

Task: *palindrome*

- **Palindrome** is a word or other sequence of characters which reads the same backward as forward such as '1331' 'noon‘.
- Write a program to tell that string S is palindrome or not.

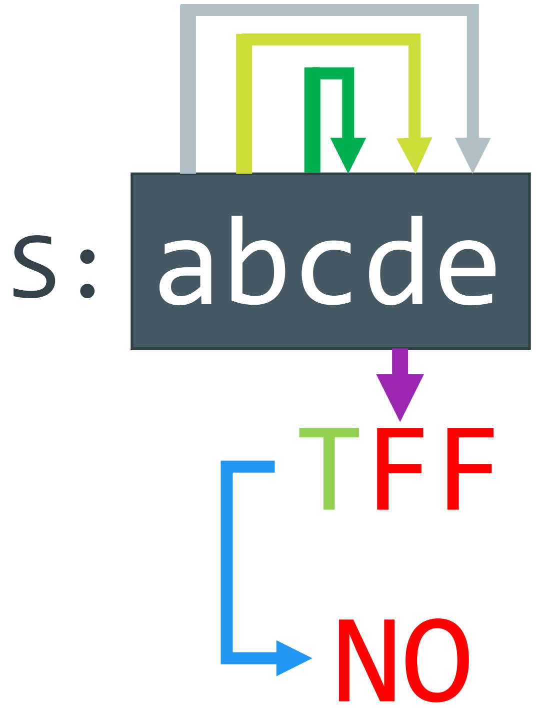
Task: *palindrome Ver.1*

```python
S = input('Enter S: ')
revS = ''
for tmp in S:
        revS = tmp + revS
check = True
for i in range(0,len(S)):
        if S[i] != revS[i]:
                check = False
if check:
        print('Yes')
else:
        print('No')
```



Task: *palindrome Ver.2*

```
S = input('Enter S: ')
check = True
for i in range(0,len(S)//2):
        if S[i] != S[len(S) - i - 1] :
                check = False
                break
if check :
        print('Yes')
else :
        print('No')
```



Making List of String

- You can separate a string by using **split()**.

```
text = 'Computer and Programing‘
print(text.split(' '))
print(text.split('m'))
```

```
['Computer', 'and', 'Programing']
['Co', 'puter and Progra', 'ing']
```

Tuple

- Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

```
tup1 = ('CPE', 'SKE', 31, 15)
tup2 = 'a', 'b', 'c'
print(())
print((50,))
print(len(tup1))
print(tup2[0])
print(tup1+tup2[:2])
for i in tup1[2:4] :
      print(i, end=' ')
```

```
()
(50, )
4
A
('CPE', 'SKE', 31, 15, 'a', 'b')
31 15
```

References

- Python Slides 2017 – **Department of Computer Engineering Kasetsart University**
- Think Python – **Allen B. Downey**
- https://automatetheboringstuff.com/chapter4/