List

What is a list?

Lists are used to store multiple items in a single variable.

List items are:

- Ordered: items take order in which they are written in code

- Changeable: you can use the assignment operator to change the items

- And allow duplicate values

List items are indexed, the first item has index [0], the second item has index [1], the third item has index [2], so and so forth.

Indexing (Part 1)

```
# Accessing list items by using their index
# Note : index start from 0
# Note : Negative index start from -1
thislist = ["apple", "banana", "cherry"]
print(thislist[0])
print(thislist[1])
print(thislist[2])
print(thislist[-1]) # Prints the last element
```

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

print(thislist[0])
print(thislist[1])
print(thislist[2])
print(thislist[-1]) # Prints the last ele
```

```
apple
banana
cherry
cherry
```

Indexing (Part 2)

Using negative numbers as indices to write a slice.

x = ["apple", "banana", "orange", "watermelon", "cherry"]

```
print(thislist[-1]) # Prints the last element
print(thislist[len(thislist) - 1])
# Index “-1” is equivalent to index “len(thislist) – 1”.

print(thislist[-2]) # Prints the second last element
print(thislist[len(thislist) - 2])

print(thislist[-3]) # Prints the third last element
print(thislist[len(thislist) - 3])
```

```
[1]: thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

[2]: thislist

[2]: ['apple', 'banana', 'cherry', 'orange', 'kiwi', 'melon', 'mango']

[3]: thislist[-1]

[3]: 'mango'

[4]: thislist[-2]

[4]: 'melon'

[5]: thislist[-3]

[5]: 'kiwi'

[6]: thislist[-4]

[6]: 'orange'

[7]: thislist[-4:-1]

[7]: ['orange', 'kiwi', 'melon']
```

```
[8]: thislist[::-1]
```

```
[8]: ['mango', 'melon', 'kiwi', 'orange', 'cherry', 'banana', 'apple']
```

```
-1 means traversing from right to left.
That means when step is negative, we need to start from right and provide an
left.
```

```
[9]: thislist[0:4:-1]
```

```
[9]: []
```

```
[13]: thislist
```

```
[13]: ['apple', 'banana', 'cherry', 'orange', 'kiwi', 'melon', 'mango']
```

```
[10]: thislist[4:0:-1] # Exclusive of 0
```

```
[10]: ['kiwi', 'orange', 'cherry', 'banana']
```

```
[12]: thislist[4::-1]
```

```
[12]: ['kiwi', 'orange', 'cherry', 'banana', 'apple']
```

Slicing (Part 1)

Slice syntax = [start index : end index : step (optional)]

Q: You are given that a list has 10 items.

Which all indices are covered by the slice [2:5]?

Ans: Three parts of the slice [2:5] are:

start index = 2

end index = 5

step = 1 (takes the default value as it is not mentioned)

Slicing is “exclusive” of end index, that means index 5 will not be a part of the slice [2:5].

That means indices covered are: 2, 3 and 4.

Slicing (Part 2)

What does this slice expand to - [:4]

Start index is not mentioned. That means start index takes the default value: 0

End index is given to be 4.

Step is not mentioned. Step takes the default value of ‘1’

Slicing (Part 3)

What does this slice expand to... [4:]

Start index is mentioned.

It has the value: 4

End index is not mentioned.

It has the default value of: len()

Step is not mentioned. Step takes the default value of ‘1’

Slicing (Part 4)

You are given a list of 10 elements.

Ques: Which all indices are present in slice... [2:10:2]

Ans:

Start index is 2. Why? Because it is the number mentioned before the first colon.

End index is 10.

‘Step’ is mentioned after the second colon. Step has value 2.

So, 2 will come. After that, 4 will come. After that, 6 will come. After that, 8 will come. That is it, because it is exclusive of end index.

Slicing (Part 3)

A slice has three parts: start index, end index and step.

Default value of start index: 0

Default value of end index: len()

Default value of step: 1

Q: What does this [2:5] expand to?

[2:5:1]

Q: What does this [:4] expand to?

Here, we can see that we have a single colon. This means that we have assume default value for step = 1.

Because the number before colon is missing that has to be start index. And number after colon is: end index.

[0:4:1]

Q: What does this [2:] expand to?

Answer: [2:len():1]

Slicing (Part 4)

```
# Accessing multiple items by using a range of indexes

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(thislist[2:5]) # Inclusive of 2 and exclusive of 5

print(thislist[:4])   # From Begining to specified index

# Note "kiwi" not included in the result

# This slice [:4] -> [0:4:1]

print(thislist[2:])

# This slice [2:] -> [2:7:1]

# 7 because it is the length of the list.

# Using Negative indexes

print(thislist[-4:-1]) # Note  - last item index value is -1 and this not included in result
```

```
['cherry', 'orange', 'kiwi']
['apple', 'banana', 'cherry', 'orange']
['cherry', 'orange', 'kiwi', 'melon', 'mango']
['orange', 'kiwi', 'melon']
```

Assignment of another list on a slice of list

- With assignment operator, you can assign to an index and assign to a slice.
- When you assign a longer list to a shorter slice, it increases the length of the list.
- When you assign a shorter list to a longer slice, it decreases the length of the list.

Adding elements to a list

- Insert(): Used to insert element at a specified index.

  Takes two arguments. First argument is the index.
- Append(): takes one argument. That argument is an element to be appended in the list.
- Extend(): takes one argument but that argument is a list for us.

  Here, usage is: mainlist.extend(listtoappend)

  It is equivalent to: mainlist = mainlist + listtoappend

  Note: They lead to increase in the length of the original list.

Removing items from a list

- Remove(): takes one argument. And argument is the element itself.

  For ex: if it is list of strings, the argument would be a string. If it is a list of integers, the argument would be an int.
- Pop(): also takes one argument (defaults to ‘len()-1’ (which means the last element) if no argument is passed) but that argument is the index.
- Del: using the keyword ‘del’ delete from a specified index.
- Clear(): empties a list

Problem

- Q: Let the input list be: x = ["apple", "banana", "orange", "watermelon", "cherry"]

  Show how you can reverse this list in three ways.

Solution

1) Through slicing:

x[::-1]

2) Using the built-in function:

list(reversed(x))

3) Using method invocation:

x.reverse()

List Comprehension (Part 1)

Creating a new list from an existing one:

```
# Creating a new list from an existing one
l = ["apple", "banana", "cherry", "guava"]
n = []
for i in l:
    n.append(i)
print(n)
```

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

```
# NOW USING LIST COMPREHENSION:

l = ["apple", "banana", "cherry", "guava"]
n = [ str(i) for i in l ]
print(n)
```

List Comprehension (Part 2)

```
# Syntax of list comprehension: [ some_function(i) for i in some_list ]

my_list = ["apple", "banana", "cherry", "guava"]
n = [ str(i) for i in my_list ]
print(n)
```

Output:
['apple', 'banana', 'cherry', 'guava']

Here:
some_function() is your: str()
some_list is your: my_list = ["apple", "banana", "cherry", "guava"]

Problem on List Comprehension

You have a list of 10 numbers.

[5, 2, 9, 1, 8, 3, 6, 0, 4, 7]

Create a new list by adding 5 to each of these numbers.

Solution

```
# Create a new list by adding 5 to each of these numbers.
l = [5, 2, 9, 1, 8, 3, 6, 0, 4, 7]
k = [i + 5 for i in l]
print(k)
```

[10, 7, 14, 6, 13, 8, 11, 5, 9, 12]

Problem 1 on: Assignment to a slice

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
thislist[1:3] = ['potato', 'tomato', 'brinjal']

What will be the output?

Solution 1 on: Assignment to a slice

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

len(thislist)
Out: 7

# What is being replaced?
thislist[1:3]
Out: ['banana', 'cherry']

thislist[1:3] = ['potato', 'tomato', 'brinjal']

thislist
Out: ['apple', 'potato', 'tomato', 'brinjal', 'orange', 'kiwi', 'melon', 'mango']

len(thislist)
8
```