Python

Python also has arrays, but they are called *lists*. It is generally the same concept, just with a different name.

Like a string, a list is a sequence of values. In a string, the values are characters; in a list, they can be any type. The values in a list are called elements or sometimes items. There are several ways to create a new list; the simplest is to enclose the elements in square brackets, [and], as shown in the following examples:

```
[10, 20, 30, 40]
['crunchy frog', 'ram bladder', 'lark vomit']
```

The first example is a list of four integers. The second is a list of three strings. The elements of a list don't have to be the same type. The following list contains a string, a float, an integer, and another list:

```
['spam', 2.0, 5, [10, 20]]
```

A list within another list is nested. A list that contains no elements is called an empty list; you can create one with empty brackets, []. As you might expect, you can assign list values to variables:

```
>>> cheeses = ['Cheddar', 'Edam', 'Gouda']
>>> numbers = [17, 123]
>>> empty = []
>>> print (cheeses, numbers, empty)
      ['Cheddar', 'Edam', 'Gouda'] [17, 123] []
```

The most common way to traverse the elements of a list is with a "for" loop. We will cover for loops in the next class. The syntax is the same as for strings:

```
for cheese in cheeses:
       print (cheese)
```

This works well if you only need to read the elements of the list. But if you want to write or update the elements, you need the indices. A common way to do that is to combine the functions range and len:

```
for i in range(len(numbers)):
       numbers[i] = numbers[i] * 2
```

This loop traverses the list and updates each element. len (short for "length") returns the number of elements in the list. range returns a list of indices from 0 to n - 1, where n is the length of the list. Each time through the loop, i gets the index of the next element. The assignment statement in the body uses i to read the old value of the element and to assign the new value.

Python provides methods that operate on lists. For example, append adds a new element to the end of a list:

```
>>> t = ['a', 'b', 'c']
>>> t.append('d')
>>> print (t)
['a', 'b', 'c', 'd']
```

extend takes a list as an argument and appends all of the elements:

```
>>> t1 = ['a', 'b', 'c']
>>> t2 = ['d', 'e']
>>> t1.extend(t2)
>>> print (t1)
['a', 'b', 'c', 'd', 'e']
```

The following example combines the two lists and leaves t2 unmodified. sort arranges the elements of the list from low to high:

```
>>> t = ['d', 'c', 'e', 'b', 'a']
>>> t.sort()
>>> print (t)
['a', 'b', 'c', 'd', 'e']
```

List methods are void as statements; they modify the list and return None. If you accidentally write t = t.sort(), you will be disappointed with the result.

There are several ways to delete elements from a list. If you know the index of the element you want, you can use pop:

```
>>> t = ['a', 'b', 'c']
>>> x = t.pop(1)
>>> print (t)
['a', 'c']
>>> print (x)
b
```

pop modifies the list and returns the element that was removed. If you don't provide an index, it deletes and returns the last element. If you don't need the removed value, you can use the del operator:

```
>>> t = ['a', 'b', 'c']
>>> del t[1]
>>> print (t)
['a', 'c']
```

If you know the element you want to remove (but not the index), you can use remove:

```
>>> t = ['a', 'b', 'c']
>>> t.remove('b')
>>> print (t)
['a', 'c']
```

The return value from remove is None. To remove more than one element, you can use del with a slice index:

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> del t[1:5]
>>> print (t)
['a', 'f']
```

Example 1

This coding example will ask user to input a set of numbers for Maryland Lottery Pick 3, and then display the set of number on command console screen. An array is used to store those numbers.

```
# Ask for user input

pick3 = ["", "", ""]
print("Welcome to Maryland Lottery Pick3!")
pick3[0] = input("Enter your first number: ")
pick3[1] = input("Enter your second number: ")
pick3[2] = input("Enter your third number: ")

# Display the result

print("Your Pick3 numbers: " + pick3[0] + pick3[1] + pick3[2])
```

Here is a sample for running the code. ">"refers to user input:

```
Welcome to Maryland Lottery Pick3!
>Enter your first number: 0
>Enter your second number: 8
>Enter your third number: 3
Your Pick3 numbers: 083
```

Example 2

This coding example will use an array to record the final exam scores for a class management system. The scores and average will be displayed on command console after all scores are input from the user.

```
# Assume the student roast is short

students = ["John Smith", "Jane Wong", "Pat Nelson"]

# Get the scores from user by using for loop

scores = [] # Create an empty array

print("Enter the final exam scores: ")
scores.append(eval(input(students[0] + ": ")))
scores.append(eval(input(students[1] + ": ")))
scores.append(eval(input(students[2] + ": ")))
```

```
# Calculate the average

avg = sum(scores) / len(scores) # sum() is array elements sum

# Display student scores

print ()
print ("The class final exam scores: ")

print("Name , Score")
print(students[0], ',', scores[0])
print(students[1], ',', scores[1])
print(students[2], ',', scores[2])

print("Average", ':', round(avg))
```

Here is a sample for running the code. “>”refers to user input:

```
Enter the final exam scores:
>John Smith: 85
>Jane Wong: 90
>Pat Nelson: 78

The class final exam scores:
Name, Score
John Smith, 85
Jane Wong, 90
Pat Nelson, 78
Average: 84
```

Example 3

This coding example is a currency converter which can convert US dollar to a user selected currency. The program will ask user to select the currency and input the amount of US dollar, and then convert the US dollar to the user selected currency according to the current rate and display on command console screen.

```
currencies = ["Euro", "British Pound", "Chinese Yuan"]
rates = [0.93, 0.80, 6.89]

# Get user input information

print("Welcome to US Dollar Converter!")

print("1 <-", currencies[0])

print("2 <-", currencies[1])

print("3 <-", currencies[2])

select = eval(input("Select the currency converted to (1, 2, or 3): "))
```

```
    print()usd = eval(input("US Dollar: "))

    # Convert

    converted = usd * rates[select - 1]

    # Display the result

    print(currencies[select - 1] + ": %.2f" % converted)
```

Here is a sample for running the code. “>”refers to user input:

```
    Welcome to US Dollar Converter!
    1 <- Euro
    2 <-British Pound
    3 <-Chinese Yuan
    >Select te currency converted to (1, 2, or 3) : 3
    >US Dollar: 10
    Chinese Yuan: 68.9
```