Lists and Dictionaries in Python

What are the Function of List and Dictionary

- A list allows the programmer to manipulate a sequence of data values of any types
- A dictionary organizes data values by association with other data values rather than by sequential position.

Lists

- A list is a sequence of data values called items or elements.
- An item can be of any type.
- Here are some real-world examples of lists:
  - A shopping list for the grocery store, a to do list, a roster for an athletic team, A guest list for a wedding, A recipe, which is a list of instructions, etc

Lists

- One dimension List

```
[1951, 1969, 1984]                  # A list of integers
['apples', 'oranges', 'cherries']   # A list of strings
[]                                  # An empty list
```

- Multi dimension List

```
[[5, 9], [541, 78]]
```

Lists

- Range function can be used to generate an integer list
- The function len and the subscript operator [] work just as they do for strings:

```
>>> first = [1, 2, 3, 4]
>>> second = range(1, 5)
>>> first
[1, 2, 3, 4]
>>> second
[1, 2, 3, 4]
>>>
```

```
>>> len(first)
4
>>> first[0]
1
>>> first[2:4]
[3, 4]
>>>
```

Lists

- Concatenation and equality operator can be used in lists
- In operator can be used in list too

```
>>> first + [5, 6]
[1, 2, 3, 4, 5, 6]
>>> first == second
True
>>>
```

```
>>> 3 in [1, 2, 3]
True
>>> 0 in [1, 2, 3]
False
>>>
```

Operator and function in list

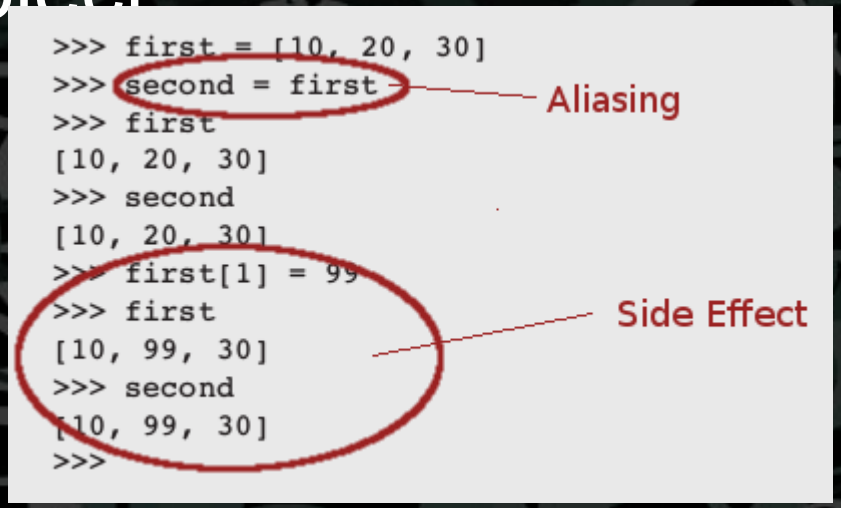
OPERATOR OR FUNCTION	WHAT IT DOES
`L[<an integer expression>]`	Subscript used to access an element at the given index position.
`L[<start>:<end>]`	Slices for a sublist. Returns a new list.
`L + L`	List concatenation. Returns a new list consisting of the elements of the two operands.
`print L`	Prints the literal representation of the list.
`len(L)`	Returns the number of elements in the list.
`range(<upper>)`	Returns a list containing the integers in the range `0` through `upper - 1`.
`==, !=, <, >, <=, >=`	Compares the elements at the corresponding positions in the operand lists. Returns `True` if all the results are true, or `False` otherwise.
`for <variable> in L:` `  <statement>`	Iterates through the list, binding the variable to each element.
`<any value> in L`	Returns `True` if the value is in the list or `False` otherwise.

List Methods for Inserting and Removing Elements

LIST METHOD	WHAT IT DOES
`L.append(element)`	Adds **`element`** to the end of **`L`**.
`L.extend(aList)`	Adds the elements of **`aList`** to the end of **`L`**.
`L.insert(index, element)`	Inserts **`element`** at **`index`** if **`index`** is less than the length of **`L`**. Otherwise, inserts **`element`** at the end of **`L`**.
`L.pop()`	Removes and returns the element at the end of **`L`**.
`L.pop(index)`	Removes and returns the element at **`index`**.

Aliasing and Side Effect

- Aliasing is two objects that refer to each other
- Side Effect is when a value in one of the list of two object that mutual refer change, then automatically a value in the other object will change and have the same value as the other object



Equality: Object Identity and Structural Equivalence

- Object identity is if the two objects has alias for the same object
- Structural Equivalence is if the two different object have the same contents
- Python's is operator can be used to test for object identity. It returns True if the two operands refer to the exact same object, and it returns False if the operands refer to distinct objects (even if they are structurally equivalent)

Equality: Object Identity and Structural Equivalence

```
>>> first = [20, 30, 40]
>>> second = first
>>> third = [20, 30, 40]
>>> first == second
True
>>> first == third
True
>>> first is second
True
>>> first is third
False
>>>
```

Tuple

- Use a pair of ”()” to declare a tuple

```
>>> fruits = ("apple", "banana")
>>> fruits
('apple', 'banana')
>>> meats = ("fish", "poultry")
>>> meats
('fish', 'poultry')
>>> food = meats + fruits
>>> food
('fish', 'poultry', 'apple', 'banana')
>>> veggies = ["celery", "beans"]
>>> tuple(veggies)
('celery', 'beans')
```

Dictionary

- A dictionary organizes information by association, not position.
- In computer science, data structures organized by association are also called tables or association lists.
- In Python, a dictionary associates a set of keys with data values.
-

```
{'Savannah':'476-3321', 'Nathaniel':'351-7743'}
```
A Phone book

```
{'Name':'Molly', 'Age':18}
```
Personal information

-

Dictionary

- The keys in a dictionary can be data of any immutable types, including other data structures, although keys normally are strings or integers.
- The associated values can be of any types.
-
-

Adding Keys and Replacing Values in Dictionary

- You add a new key/value pair to a dictionary by using the subscript operator [] .The form of this operation is the following:

```
<a dictionary>[<a key>] = <a value>
```

- Example :

```
>>> info = {}
>>> info["name"] = "Sandy"
>>> info["occupation"] = "hacker"
>>> info
{'name': 'Sandy', 'occupation': 'hacker'}
>>>
```

To Check Key in Dictionary

- Use get method to check if the key is exist in the dictionary
- This method expects two arguments, a possible key and a default value.
- If the key is in the dictionary, the associated value is returned. If the otherwise happens, the default value is returned
- Example :
  - info[“name”] = “Sandy”
  - info[“occupation”] = “hacker”
  - print info.get(“job”,None)

Remove Key in Dictionary

- Pop is used to remove the key in dictionary
- This method expects a key and an optional default value as arguments
- If the key is in the dictionary, it is removed and its associated value is returned. Otherwise, the default value is returned
- Example :
  - info[“name”] = “Sandy”
  - info[“occupation”] = “hacker”
  - print info.pop(“job”,None)
  - print info.pop(“occupation”)

Use the for loop for printing items in Dictionary

- Example to print the items in info dictionary

```
for key in info:
    print key, info[key]
```

```
for (key, value) in grades.items():
   print key, value
```

- You can also use dictionary method items to print the content of the dictionary

```
>>> grades = {90:"A", 80:"B", 70:"C"}
>>> grades.items()
[(80, 'B'), (90, 'A'), (70, 'C')]
```

Dictionary Operation

DICTIONARY OPERATION	WHAT IT DOES
`len(d)`	Returns the number of entries in **d**.
`aDict[key]`	Used for inserting a new key, replacing a value, or obtaining a value at an existing key.
`d.get(key [, default])`	Returns the value if the key exists or returns the default if the key does not exist. Raises an error if the default is omitted and the key does not exist.
`d.pop(key [, default])`	Removes the key and returns the value if the key exists or returns the default if the key does not exist. Raises an error if the default is omitted and the key does not exist.
`d.keys()`	Returns a list of the keys.
`d.values()`	Returns a list of the values.

Dictionary Operation

DICTIONARY OPERATION	WHAT IT DOES
`d.items()`	Returns a list of tuples containing the keys and values for each entry.
`d.has_key(key)`	Returns **`True`** if the key exists or **`False`** otherwise.
`d.clear()`	Removes all the keys.
`for key in d:`	**`key`** is bound to each key in **`d`** in an unspecified order.

Home Work

- Write the application to convert hexadecimal and octal into binary number system

Reference

Fundamentals of Python from First Program Through Data Structures Chapter 5