Fundamentals of Programming

Unit 5 Tuples & Dictionaries

Ms. Maggie Lee | CUSCS

More about Tuples

Perform more operations with tuples...

Tuples

- Tuples
  - one of the sequences and a collection of objects which ordered and immutable
  - tuple indices start at 0, and they can be sliced and concatenated
  - the value of the items stored in the tuple cannot be changed

- Create a Tuple

```
tupleA = ('Python', 'Programming')
tupleB = (123, 456, 789)

print(tupleA)
print(tupleB)
```

```
= RESTART:
('Python', 'Programming')
(123, 456, 789)
```

Programming Code

```
tupleA = ('Python', 'Programming')
tupleB = (123, 456, 789)

print(tupleA)
print(tupleB)
```


Tuples

- Access Values in Tuple
  - use the square brackets for slicing along with the index or indices to obtain value available at that index

- Example

```
tupleA = ('Python', 'Programming', 'Learning', 'Happy')

print("First item:", tupleA[0])
print("2nd to 4th items: ", tupleA[1:4])
```

```
= RESTART:
First item: Python
2nd to 4th items:  ('Programming', 'Learning', 'Happy')
```

Programming Code

```
tupleA = ('Python', 'Programming', 'Learning', 'Happy')
print("First item:", tupleA[0])
print("2nd to 4th items:", tupleA[1:4])
```

Tuples

- Add or Update Values in Tuples
  - tuples are immutable → cannot update or change the values of tuple elements
  - take portions of existing tuples to create new tuples
- Example

```
tupleA = ('Python', 'Programming', 'Learning', 'Happy')
tupleA[0] = 'School'
```

```
= RESTART:
Traceback (most recent call last):
  File "                                                    ", l
ine 3, in <module>
    tupleA[0] = 'School'
TypeError: 'tuple' object does not support item assignment
```

Programming Code

```
tupleA = ('Python', 'Programming', 'Learning', 'Happy')
tupleA[0] = 'School'
```


Tuples

- Example

```
tupleA = ('Python', 'Programming', 'Learning', 'Happy')
tupleB = (123, 456, 789)
tupleC = tupleA[0:2] + tupleB[1:3]

print(tupleC)
```

```
= RESTART:
('Python', 'Programming', 456, 789)
```

Programming Code

```
tupleA = ('Python', 'Programming', 'Learning', 'Happy')
tupleB = (123, 456, 789)
tupleC = tupleA[0:2] + tupleB[1:3]
print(tupleC)
```

Tuples

- Delete Values in Tuple
  - cannot remove individual tuple elements
  - put together another tuple with the undesired elements discarded
  - use the ***del*** statement to explicitly remove an entire tuple
- Example

```
tupleA = ('Python', 'Programming', 'Learning', 'Happy')
del tupleA[0]
```

```
= RESTART:
Traceback (most recent call last):
  File "                                                    ", l
ine 3, in <module>
    del tupleA[0]
TypeError: 'tuple' object doesn't support item deletion
```

Programming Code

```
tupleA = ('Python', 'Programming', 'Learning', 'Happy')
del tupleA[0]
```


Tuples

- Example

```
tupleA = ('Python', 'Programming', 'Learning', 'Happy')
print(tupleA)

del tupleA
print(tupleA)
```

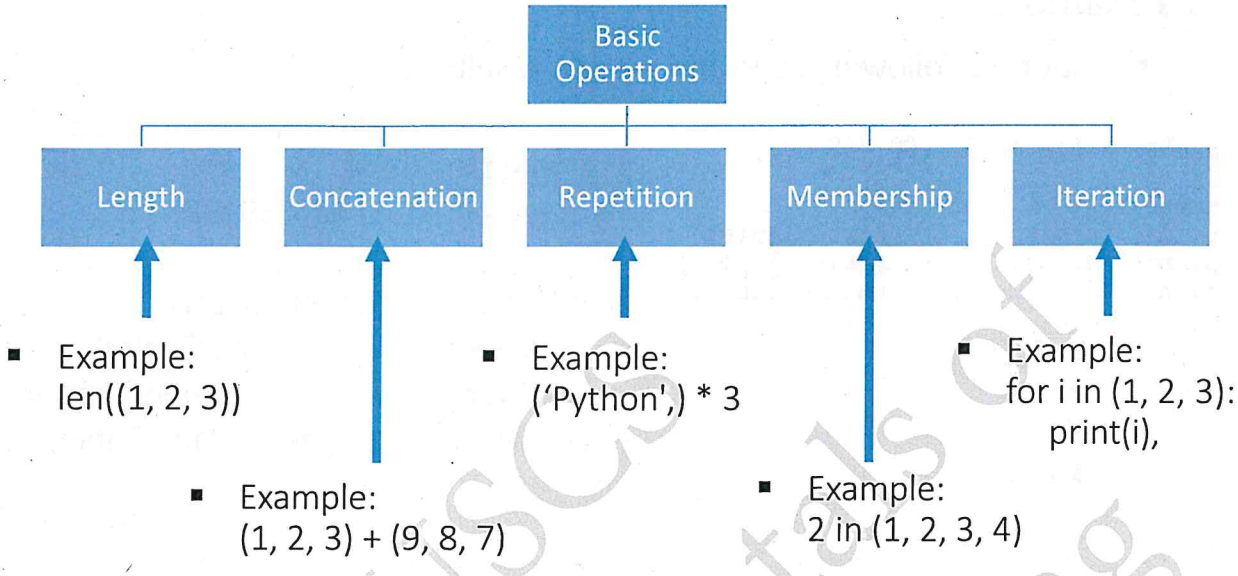
```
= RESTART:
('Python', 'Programming', 'Learning', 'Happy')
Traceback (most recent call last):
  File "                                                    ", l
ine 6, in <module>
    print(tupleA)
NameError: name 'tupleA' is not defined
```

Programming Code

```
tupleA = ('Python', 'Programming', 'Learning', 'Happy')
print(tupleA)
del tupleA
print(tupleA)
```

Tuples

- Basic Operations



Tuple Common Methods

- len()
  - return the number of elements in the tuple
- tuple()
  - convert a list of items into tuples
- max()
  - return the elements from the tuple with maximum value
- min()
  - return the elements from the tuple with minimum value

Tuple Operations

- Example
  - Input the following code and see the results

```
number = (1, 4, 5, 100, 59, 98)

numberTuple = tuple(number)
print("Length: ", len(numberTuple))
print("Maximum: ", max(numberTuple))
print("Minimum: ", min(numberTuple))
```

```
= RESTART:
Length:  6
Maximum:  100
Minimum:  1
```

Programming Code

```
number = (1, 4, 5, 100, 59, 98)

numberTuple = tuple(number)
print("Length: ", len(numberTuple))
print("Maximum: ", max(numberTuple))
print("Minimum: ", min(numberTuple))
```


Tuple Operations

- Example
  - Input the following code and see the results

```
items = ("Bread", "Milk", "Market")

itemsTuple = tuple(items)
print("Length: ", len(itemsTuple))
print("Maximum: ", max(itemsTuple))
print("Minimum: ", min(itemsTuple))
```

```
Length:  3
Maximum:  Milk
Minimum:  Bread
```

Programming Code

```
items = ("Bread", "Milk", "Market")

itemsTuple = tuple(items)
print("Length: ", len(itemsTuple))
print("Maximum: ", max(itemsTuple))
print("Minimum: ", min(itemsTuple))
```

Tuple Operations

- Example
  - Input the following code and see the results

```
items = (10, 33, "Bread", "Milk", "Market")

itemsTuple = tuple(items)
print("Length: ", len(itemsTuple))
print("Maximum: ", max(itemsTuple))
print("Minimum: ", min(itemsTuple))
```

Programming Code

```
items = (10, 33, "Bread", "Milk", "Market")

itemsTuple = tuple(items)
print("Length: ", len(itemsTuple))
print("Maximum: ", max(itemsTuple))
print("Minimum: ", min(itemsTuple))
```

```
= RESTART:
Length:  5
Traceback (most recent call last):
  File "                                                    ", l
ine 5, in <module>
    print("Maximum: ", max(itemsTuple))
TypeError: '>' not supported between instances of 'str' and 'int'
```


Tuple Operations

- Class Activity
  - Ask the user to input the product list in the supermarket. Before the confirmation, the user can still update the existing items in the product list.
  - If the user wants to update the list, user should input the item number, say '1' for the first item, '2' for the second item in the list and so on. The program should ask the user to confirm the list until the confirmation from users.
  - Once the user confirmed the product list, the user can only check the number of items in the product list and show those items.
  - Display the results as shown below.

Tuple Operations

- Class Activity

```
= RESTART:
Enter the item (NA to end the input): Milk
Enter the item (NA to end the input): Juice
Enter the item (NA to end the input): Water
Enter the item (NA to end the input): NA
Existing Product List:  ['Milk', 'Juice', 'Water']
---------------------------------------
Confirm the product list? (Y/N) n
Existing Product List:  ['Milk', 'Juice', 'Water']
Please indicate the item no. to be updated: 2
Please input the new item: Coffee
Updated Product List:  ['Milk', 'Coffee', 'Water']
---------------------------------------
Confirm the product list? (Y/N) Y
('Milk', 'Coffee', 'Water')
Number of items in product list:  3
---------------------------------------
```


More about Dictionaries

Perform more operations with tuples...

Dictionaries

- Dictionaries
  - store the data in a key-value pair format
  - keys are unique within a dictionary while values may not be
  - the values of a dictionary can be of any type
  - the keys must be of an immutable data type such as strings, numbers, or tuples
- Create a Dictionary
  - Review Example

```
dictA = {'Name':'Peter', 'Class':'1A'}
print(dictA)
```

```
= RESTART:
{'Name': 'Peter', 'Class': '1A'}
```

Programming Code

```
dictA = {'Name':'Peter', 'Class':'1A'}

print(dictA)
```


Dictionaries

- Access Values in Dictionary
  - use the familiar square brackets along with the key to obtain its value
- Example

```
dictA = {'Name':'Peter', 'Class':'1A'}

print("dict[\"Name\"]: ", dictA["Name"])
print("dict[\"Class\"]: ", dictA["Class"])
```

```
= RESTART:
dict["Name"]:  Peter
dict["Class"]:  1A
```

Programming Code

```
dictA = {'Name':'Peter', 'Class':'1A'}
print("dict[\"Name\"]: ", dictA["Name"])
print("dict[\"Class\"]: ", dictA["Class"])
```

Dictionaries

- Add or Update Values in Dictionaries
  - update a dictionary by adding a new entry or a key-value pair, modifying an existing entry

- Example

```
= RESTART:
{'Name': 'Peter', 'Class': '1B', 'Program': 'Python'}
```

```
dictA = {'Name':'Peter', 'Class':'1A'}

dictA['Program'] = 'Python'
dictA['Class'] = '1B'

print(dictA)
```

Programming Code

```
dictA = {'Name':'Peter', 'Class':'1A'}
dictA['Program'] = 'Python'
dictA['Class'] = '1B'
print(dictA)
```


Dictionaries

- Delete Values in Dictionary
  - can either remove individual dictionary elements or clear the entire contents of a dictionary
  - use the ***del*** statement to explicitly remove an item or entire dictionary

- Example

```
dictA = {'Name':'Peter', 'Class':'1A'}

del dictA['Class']
print(dictA)
del dictA
print(dictA)
```

```
= RESTART:
{'Name': 'Peter'}
Traceback (most recent call last):
  File
line 6, in <module>
    print(dictA)
NameError: name 'dictA' is not defined
```

Programming Code

```
dictA = {'Name':'Peter', 'Class':'1A'}
del dictA['Class']
print(dictA)
del dictA
print(dictA)
```

Dictionaries

- Dictionary Values
  - have no restrictions
  - can be any arbitrary Python object, either standard objects or user-defined objects

- Dictionary Keys
  1. more than one entry per key not allowed. Which means no duplicate key is allowed
     - when duplicate keys encountered during assignment, the last assignment wins
  2. keys must be immutable
     - can use strings, numbers or tuples as dictionary keys

Dictionaries

- Example

```
dictB = {'Program': 'Java', 'School':'AAA School', 'Program': 'Python'}

print(dictB)

dictC = {['Program']: 'Python', 'School':'AAA School'}
print(dictC)
```

```
>>>
= RESTART:
{'Program': 'Python', 'School': 'AAA School'}
Traceback (most recent call last):
  File
line 5, in <module>
    dictC = {['Program']: 'Python', 'School':'AAA School'}
TypeError: unhashable type: 'list'
```

Programming Code

```
dictB = {'Program': 'Java', 'School':'AAA School', 'Program': 'Python'}
print(dictB)
dictC = {['Program']: 'Python', 'School':'AAA School'}
print(dictC)
```

Dictionary Common Methods

- len()
  - return the total length of the dictionary
  - equal to the number of items in the dictionary
- update()
  - add the another dictionary's key-values pairs in to the original dictionary
  - do not return anything

Dictionary Operations

- Example
  - Input the following code and see the results

```
dictA = {'School': 'ABC School'}
dictB = {'Type':'Secondary' , 'Class' : 5}

print(dictA)
print("Length of dictA: ", len(dictA))
dictA.update(dictB)
print(dictA)
print("Length of dictA: ", len(dictA))
```

```
= RESTART:
{'School': 'ABC School'}
Length of dictA:  1
{'School': 'ABC School', 'Type': 'Secondary', 'Class': 5}
Length of dictA:  3
```

Programming Code

```
dictA = {'School': 'ABC School'}
dictB = {'Type':'Secondary' , 'Class' : 5}

print(dictA)
print("Length of dictA: ", len(dictA))
dictA.update(dictB)
print(dictA)
print("Length of dictA: ", len(dictA))
```

Dictionary Common Methods

- values()
  - return a list of all the values available in a given dictionary
- keys()
  - return a list of all the available keys in the dictionary
- Example
  - Input the following code and see the results

```
dictA = {'Program' :'Python', 'Mode':'Part-time'}

print("Values: ", dictA.values())
print("Keys: ", dictA.keys())
```

```
= RESTART:
Values:  dict_values(['Python', 'Part-time'])
Keys:  dict_keys(['Program', 'Mode'])
```

Programming Code

```
dictA = {'Program' :'Python',
'Mode':'Part-time'}
print("Values: ", dictA.values())
print("Keys: ", dictA.keys())
```


Dictionary Operations

- Class Activity
  - Ask the user to input the names of the students in the class. Each student will be assigned with a class number, starting from 1, according to the input order.
  - If the user inputs "End" which is not case sensitive, the input progress will be ended.
  - The program will show the total number of students and show the student list.
  - Display the results as shown below.

```
Enter the student's name (Type 'End' to quit): Peter
Enter the student's name (Type 'End' to quit): Mary
Enter the student's name (Type 'End' to quit): John
Enter the student's name (Type 'End' to quit): end
There are  3 students.
Current Student List (with Class Number):  {1: 'Peter', 2: 'Mary', 3: 'John'}
```

Hope you enjoy the class

Thank you