Models in Memory in Python

Roberto A. Bittencourt

List all

Listing all elements from the collection

- Beyond the typical CRUD operations, it is very common to create an operation for listing all elements from the object collection
- This is more typical when the collection is encapsulated in an object and one wants to create a getter method for the collection
- Usually a copy of the collection is created so that it may be operated on later (e.g., sorting) without affecting the original collection
- Typically the collection will be returned as a simpler collection (e.g., a list) or as an Iterator

List all: Program in Python with an ***unordered list*** or with an ***ordered list***

```
def list_all(self):
    students_list = []
    for student in self.students:
        students_list.append(student)
    return students_list

def main():
  school = School('Central High')
  school.create('Peter Smith', 123, 1995)
  school.create('Rika Nakamura', 345, 1987)
  school.create('Peter Parker', 234, 1998)
  lista = school.list_all()
  print("Listing all students:")
  for student in lista:
    print(student)
```

List all: Program in Python with a ***dictionary***

```
def list_all(self):
    students_list = []
    for student in self.students.values():
        students_list.append(student)
    return students_list

def main():
  school = School('Central High')
  school.create('Peter Smith', 123, 1995)
  school.create('Rika Nakamura', 345, 1987)
  school.create('Peter Parker', 234, 1998)
  lista = school.list_all()
  print("Listing all students:")
  for student in lista:
    print(student)
```

Testing School: **list all students** (school_test.py) (1 of 2)

```
def test_list_all(self):
    student1 = Student('Peter', 123, 2010)
    student2 = Student('Ali', 234, 2011)
    student3 = Student('Kala', 345, 2009)
    student4 = Student('Jin', 567, 2007)
    student5 = Student('Marcos', 789, 2008)

    school = School('Central High')
    self.assertEqual(0, len(school.list_all()), "empty student collection")

    # add one student
    school.create('Peter', 123, 2010)

    # List a singleton list
    listed = school.list_all()
    self.assertEqual(1, len(listed), "there should be 1 student in the list")
    self.assertEqual(student1, listed[0], "Listing Peter")

    # add some students
    school.create('Ali', 234, 2011)
    school.create('Kala', 345, 2009)

    # List with three students
    listed = school.list_all()
    self.assertEqual(3, len(listed), "there should be 3 students in the list")
    self.assertEqual(student1, listed[0], "Listing Peter")
    self.assertEqual(student2, listed[1], "Listing Ali")
    self.assertEqual(student3, listed[2], "Listing Kala")
```

Testing School: **list all students** (school_test.py) (2 of 2)

```
# continuing
# delete some students
school.delete(345)
school.delete(123)

# List is a singleton list again
listed = school.list_all()
self.assertEqual(1, len(listed), "there should be 1 student in the list")
self.assertEqual(student2, listed[0], "Listing Ali")

# add some more students
school.create('Jin', 567, 2007)
school.create('Marcos', 789, 2008)

# List is back with three students, different ones
listed = school.list_all()
self.assertEqual(3, len(listed), "there should be 3 students in the list")
self.assertEqual(student2, listed[0], "Listing Ali")
self.assertEqual(student4, listed[1], "Listing Jin")
self.assertEqual(student5, listed[2], "Listing Marcos")

# delete all students
school.delete(567)
school.delete(234)
school.delete(789)

# List is empty again
self.assertEqual(0, len(school.list_all()), "list should be empty")
```

Sorting Object Collections

Sorting an object list

- Sorting an object list is essentially similar to sorting an integer list
- The only differences will be:
  - Comparison will be done against the chosen object sorting field
  - When swapping elements, swap the objects themselves

Recalling **Selection Sort**

Given the list named `lista` and being `size` the list length...

```
size = len(lista)
for i in range(0, size-1):
   min = i
   for j in range(i+1, size):
      //Find the least value
      if (lista[j] < lista [min]):
         min = j
   lista[i], lista[min] = lista[min], lista[i]
```

Selection Sort: Program in Python that sorts an ***unordered list*** by **number**

```
def selection_sort_by_number(self):
    size = len(self.students)
    for i in range(0, size-1):
       min = i
       for j in range(i+1, size):
          # Find the least student number
          if (self.students[j].number < self.students[min].number):
             min = j
       self.students[i], self.students[min] = self.students[min], self.students[i]
def main():
  school = School('Central High')
  school.create('Marcos', 789, 1997)
  school.create('Rika', 345, 1987)
  school.create('Ali', 234, 1998)
  school.create('Latifa', 456, 1999)
  school.create('Peter', 123, 1995)
  school.create('Jin', 567, 2001)
  for student in school.students:
    print(student)
  print("\n\nSorting by student number: \n")
  school.selection_sort_by_number()
  for student in school.students:
    print(student)
```

Selection Sort: Program in Python that sorts an ***unordered list*** by **name**

```
def selection_sort_by_name(self):
    size = len(self.students)
    for i in range(0, size-1):
       min = i
       for j in range(i+1, size):
          # Find the name that comes first
          if (self.students[j].name < self.students[min].name):
             min = j
       self.students[i], self.students[min] = self.students[min], self.students[i]
def main():
  school = School('Central High')
  school.create('Marcos', 789, 1997)
  school.create('Rika', 345, 1987)
  school.create('Ali', 234, 1998)
  school.create('Latifa', 456, 1999)
  school.create('Peter', 123, 1995)
  school.create('Jin', 567, 2001)
  for student in school.students:
    print(student)
  print("\n\nSorting by student name: \n")
  school.selection_sort_by_name()
  for student in school.students:
    print(student)
```

Selection Sort: Program in Python that presents a ***dictionary*** sorted by number

```
def selection_sort_by_number(lista):
    size = len(lista)
    for i in range(0, size-1):
       min = i
       for j in range(i+1, size):
          # Find the least element
          if (lista[j] < lista[min]):
             min = j
       lista[i], lista[min] = lista[min], lista[i]
def keys_sorted_by_number(self):
    keys = list(self.students.keys())
    selection_sort_by_number(keys)
    return keys
def main():
  school = School('Central High')
  school.create('Marcos', 789, 1997)
  school.create('Rika', 345, 1987)
  school.create('Ali', 234, 1998)
  school.create('Latifa', 456, 1999)
  school.create('Peter', 123, 1995)
  school.create('Jin', 567, 2001)
  for key in students:
      print(students[key])
  print("\n\nPresenting students sorted by student number:\n")
  keys = school.keys_sorted_by_number()
  for key in keys:
      print(students[key])
```

Testing School: **selection sort by number** and **selection sort by name** (school_test.py)

- These testing methods are left to you as an exercise
- Which strategies would you use to test them?