Lab 7.1 (Deadline Monday 7 March 23:59)

- Upload your code to Einstein to have it verified.

Element

- In `element_071.py` define an `Element` class to model a chemical element.
- An element has four data attributes: `number`, `name`, `symbol`, and `boiling point`.
- The `Element` class defines the following instance methods:
  - `set_attributes()` : sets the instance's attributes to the specified values
  - `print_attributes()` : prints the instance's attributes

- When your class is correctly implemented, running the following program should produce the given output.

```
from element_071 import Element

def main():

    e1 = Element()
    e2 = Element()
    e3 = Element()
    e4 = Element()
    e5 = Element()

    e1.set_attributes(1, 'Hydrogen', 'H', 20.3)
    e1.print_attributes()

    assert(e1.number == 1)
    assert(e1.name == 'Hydrogen')
    assert(e1.symbol == 'H')
    assert(e1.bp == 20.3)

    e2.set_attributes(3, 'Lithium', 'Li', 1615)
    e2.print_attributes()

    e3.set_attributes(11, 'Sodium', 'Na', 1156)
    e3.print_attributes()

    e4.set_attributes(12, 'Magnesium', 'Mg', 1380)
    e4.print_attributes()

    e5.set_attributes(79, 'Gold', 'Au', 3129)
    e5.print_attributes()


if __name__ == '__main__':
    main()
```

```
Number: 1
Name: Hydrogen
Symbol: H
Boiling point: 20.3 K
Number: 3
```

```
Name: Lithium
Symbol: Li
Boiling point: 1615 K
Number: 11
Name: Sodium
Symbol: Na
Boiling point: 1156 K
Number: 12
Name: Magnesium
Symbol: Mg
Boiling point: 1380 K
Number: 79
Name: Gold
Symbol: Au
Boiling point: 3129 K
```

Bank account

- In *bank_071.py* define a `BankAccount` class to model a bank account.
- An bank account has three data attributes: `name`, `number`, and `balance`.
- The `BankAccount` class defines the following instance methods:
  - `set_attributes()` : sets the instance's attributes to the specified values
  - `print_attributes()` : prints the instance's attributes
  - `deposit()` : increases the balance by a given amount

- Once your class is correctly implemented, running the following program should produce the given output.

```python
from bank_071 import BankAccount

def main():

    b1 = BankAccount()
    b1.set_attributes('Jim', 12343111, 300)

    assert(b1.name == 'Jim')
    assert(b1.number == 12343111)
    assert(b1.balance == 300)

    b1.print_attributes()
    b1.deposit(100)
    b1.print_attributes()

    assert(b1.balance == 400)


if __name__ == '__main__':
    main()
```

```
Name: Jim
Account number: 12343111
Balance: 300.00
Name: Jim
Account number: 12343111
Balance: 400.00
```

Point

- In *point_071.py* define a `Point` class to model a point in a two dimensional space.
- A point has two data attributes: `x` and `y`.
- The `Point` class defines the following instance methods:
  - `set_attributes()` : sets the instance's attributes to the specified values
  - `print_attributes()` : prints the instance's attributes
  - `reflect()` : reflects a point's coordinates through the origin (the effect is to negate the point's `x` and `y` coordinates)

- When your class is correctly implemented, running the following program should produce the given output.

```
from point_071 import Point

def main():
    p1 = Point()
    p2 = Point()

    p1.set_attributes(5, 5)
    p2.set_attributes(4.2, 3.8)

    p1.print_attributes()
    p2.print_attributes()

    assert(p1.x == 5)
    assert(p1.y == 5)

    p1.reflect()
    p1.print_attributes()

    assert(p1.x == -5)
    assert(p1.y == -5)

if __name__ == '__main__':
    main()
```

```
x: 5.00
y: 5.00
x: 4.20
y: 3.80
x: -5.00
y: -5.00
```

Student

- In *student_071.py* define a `Student` class to model a student.
- A student has three data attributes: `sid` (student ID), `name` and `modlist` (the list of modules for which the student is registered).

- The `Student` class defines the following instance methods:
  - `set_attributes()` : sets the instance's attributes to the specified values (see the example below)
  - `print_attributes()` : prints the instance's attributes (see the example below)
  - `add_module()` : adds a module (passed as an argument) to `modlist` (has no effect if `modlist` already contains the module)
  - `del_module()` : removes a module (passed as an argument) from `modlist` (has no effect if the module is not in `modlist`)

- When your class is correctly implemented, running the following program should produce the given output.

```
from student_071 import Student

def main():

    s1 = Student()

    s1.set_attributes(15234654, 'Jimmy Murphy', ['CA116'])
    s1.print_attributes()

    assert(s1.name == 'Jimmy Murphy')
    assert(s1.sid == 15234654)
    assert(s1.modlist == ['CA116'])

    s1.add_module('CA117')
    s1.print_attributes()

    s1.add_module('CA100')
    s1.del_module('CA116')
    s1.print_attributes()

    assert(s1.modlist == ['CA117', 'CA100'])

if __name__ == '__main__':
    main()
```

```
ID: 15234654
Name: Jimmy Murphy
Modules: CA116
ID: 15234654
Name: Jimmy Murphy
Modules: CA116, CA117
ID: 15234654
Name: Jimmy Murphy
Modules: CA117, CA100
```