Assignment 11

Classes and objects

LILLEBAELT ACADEMY OF
PROFESSIONAL HIGHER EDUCATION

Author
Martin Grønholdt
mart80c7@edu.eal.dk

Sunday 5 February 2017

Table of Contents

Introduction

The programs in this hand-in is an example of using classes in Python, the getter/setter patterns is implemented throughout. The code follows the convention from the Python documentation[1] prepending private members of classes with and underscore, even though the members are as public as ever.

All files for this hand in are available at:
https://github.com/deadbok/eal_programming/tree/master/ass11

Error handling

All programs that requests user input, handle bad input by asking the user, to use only the correct data type.

Generally all error conditions lead to an error message where after the program exits or continues depending on the severity of the error.

1 https://docs.python.org/3.6/tutorial/classes.html#tut-private

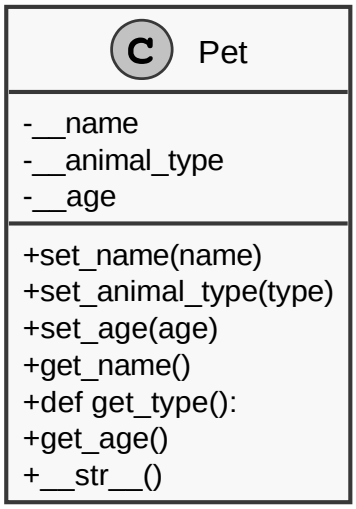
1. Pet Class

This programs uses a class to store information about a pet. A user enters the information, that is stored in an instance of the single class in this program, Pet. Pet stores this information and prints it back using the overloaded __str__ method.

prog1.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 1 "Pet Class"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-02-01)

Program for entering details in to a pet class.
"""
import pet


def main():
    """
    Program main entry point.
    """
    # Instantiate the pet class.
    my_pet = pet.Pet()

    try:
        # Use the setters, to set the values from user input.
        my_pet.set_name(input('Input the name of the pet: '))
        my_pet.set_animal_type(input('Input the type of the pet: '))
        my_pet.set_age(int(input('Input the age of the pet: ')))
    except ValueError:
        # Complain when something unexpected was entered.
        print('\nThe input was wrong.')
        exit(1)

    # Use the __str__ method to print the pet object data.
    print('\n{}'.format(my_pet))


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

pet.py



Pet class diagram.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 1 "Pet Class"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-03-01)

Program to organise name and email addresses.
"""


class Pet:
    """
    Pet class, holds the name, type, and age of a pet.
    """

    def __init__(self, name='', animal_type=None, age=0):
        """
        Constructor.

        :param name: The pets name.
        :param animal_type: The type of pet.
        :param age: The age of the pet. Default is 0.
        """
        self.__name = name
        self.__animal_type = animal_type
        self.__age = age

    def set_name(self, name):
        """
        Set the name of the pet.
        :param name: The name
        """
        self.__name = name
```

```
    def set_animal_type(self, type):
        """
        Set the type of the pet.

        :param type: The type of pet.
        """
        self.__animal_type = type

    def set_age(self, age):
        """
        Set the age of the pet.

        :param age: Age of the pet.
        """
        self.__age = age

    def get_name(self):
        """
        Get the name of the pet.

        :return: The name of the pet.
        """
        return self.__name

    def get_animal_type(self):
        """
        Get the type of the pet.

        :return: The type of the pet.
        """
        return self.__animal_type

    def get_age(self):
        """
        Get the age of the animal.

        :return: The age of the pet.
        """
        return self.__age

    def __str__(self):
        """
        Return a string representation of the pet.

        :return: The string.
        """
        return (
            'Name:\t{}\nType:\t{}\nAge:\t{}'.format(self.__name,
                                                   self.__animal_type,
                                                   self.__age))
```

Result

```
Input the name of the pet: Jonni
Input the type of the pet: Rabbit
Input the age of the pet: 9+

The input was wrong.
```

```
Input the name of the pet: Jonni
Input the type of the pet: Rabbit
Input the age of the pet: 10

Name: Jonni
Type: Rabbit
Age:  10
```

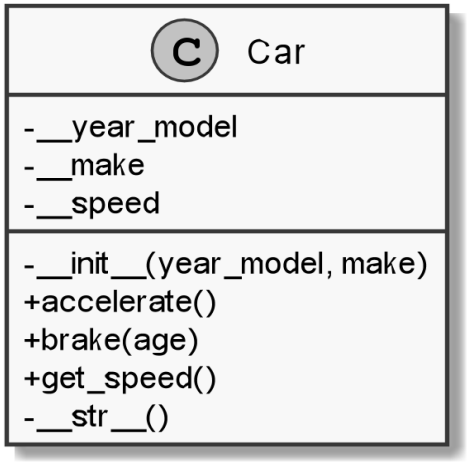
2. Car Class

This programs uses methods in a Car class to simulate a speeding and breaking car.

prog2.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 2 "Car Class"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-02-01)

Program that drives a car.
"""
import car


def main():
    """
    Program main entry point.
    """
    # Instantiate the car class.
    my_car = car.Car(1989)

    # Print the static car data.
    print('The car:\n{}'.format(my_car))

    # Print the initial speed.
    print('Current speed: {}'.format(my_car.get_speed()))

    # Accelerate 5 times and print the speed.
    for counter in range(5):
        my_car.accelerate()
        print('Current speed: {}'.format(my_car.get_speed()))

    # De-accelerate 5 times and print the speed.
    for counter in range(5):
        my_car.brake()
        print('Current speed: {}'.format(my_car.get_speed()))


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

car.py



Car class diagram.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 2 "Car Class"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-03-01)

Class that encapsulates the properties of a car.
"""


class Car:
    """
    Car class.
    """

    def __init__(self, year_model=0, make='Trabant'):
        """
        Constructor.

        :param year_model: Year of the model. Default is 0.
        :param make: Make of the cat. Default is Trabant.
        """
        # Set to the values passed to the constructor.
        self.__year_model = year_model
        self.__make = make
        # We're going no where.
        self.__speed = 0

    def accelerate(self):
        """
        Add some speed.
        """
        self.__speed += 5

    def brake(self):
```

```
        """
        Subtract some speed.
        """
        self.__speed -= 5

    def get_speed(self):
        """
        Get the current speed of the car

        :return: The speed of the car.
        """
        return self.__speed

    def __str__(self):
        """
        Return a string with the value of the year model and make.
        :return:
        """
        return (' Year model:\t{}\n'.format(self.__year_model) +
                ' Make:\t{}\n'.format(self.__make))
```

Result

```
The car:
 Year model:      1989
 Make:            Trabant

Current speed: 0
Current speed: 5
Current speed: 10
Current speed: 15
Current speed: 20
Current speed: 25
Current speed: 20
Current speed: 15
Current speed: 10
Current speed: 5
Current speed: 0
```

3. Personal Information Class

This programs uses a list of PersonalInfo instances to store and print personal information.

prog3.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 3 "Personal Information Class"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-02-04)

Program that uses a class to hold the personal information of a few people.
"""
import persinfo

def main():
    """
    Program main entry point.
    """
    # Create a list of people.
    people = list()
    # Populate 3 instances.
    people.append(
        persinfo.PersonalInfo('Martin B. K. Grønholdt', 'World', '500',
                              '123456789'))
    people.append(
        persinfo.PersonalInfo('Jonni Rabbit', 'Cage', '9+',
                              'none'))
    people.append(
        persinfo.PersonalInfo('Erling', 'Rat paradise', '3 (deceased)',
                              'none'))
    #Print all people.
    for person in people:
        print(person)


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

persinfo.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 3 "Personal Information Class"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-03-01)

Class that encapsulates personal information.
"""
```

```
class PersonalInfo:
    """
    Class that holds personal information.
    """
    def __init__(self, name='John Doe', address='', age=0, phone=''):
        """
        Constructor.

        :param name: The name of the person. Default is 'John Doe'.
        :param address: The address of the person.
        :param age: The age of the person. Default is 0.
        :param phone: The phone number of the person.
        """
        # Set to the values passed to the constructor.
        self.__name = name
        self.__address = address
        self.__age = age
        self.__phone = phone

    def get_name(self):
        """
        Get the name.

        :return: The name.
        """
        return self.__name

    def get_address(self):
        """
        Get the address

        :return: The address
        """
        return self.__address

    def get_age(self):
        """
        Get the age

        :return: The age
        """
        return self.__age

    def get_phone(self):
        """
        Get the phone number.

        :return: The phone number
        """
        return self.__phone

    def set_name(self, name):
        """
        Set the name.
        """
        self.__name = name

    def set_address(self, address):
        """
        Set the address
```

```
        """
        self.__address = address

    def set_age(self, age):
        """
        Set the age
        """
        self.__age = age

    def set_phone(self, phone):
        """
        Set the phone number.
        """
        self.__phone = phone

    def __str__(self):
        """
        Return a string that is directly usable when printing the entry.
        :return:
        """
        return('Name:\t\t{}\n'.format(self.__name) +
               'Age:\t\t{}\n'.format(self.__age) +
               'Address:\t{}\n'.format(self.__address) +
               'Phone:\t\t{}\n'.format(self.__phone))
```

Result

```
Name:       Martin B. K. Grønholdt
Age:        500
Address:    World
Phone:      123456789

Name:       Jonni Rabbit
Age:        9+
Address:    Cage
Phone:      none

Name:       Erling
Age:        3 (deceased)
Address:    Rat paradise
Phone:      none
```

4. Employee Class

This programs uses a list of Employee instances to store and print employee information. Getters are used when printing the data.

prog4.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 4 "Employee Class"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-02-04)

Program that uses a class to hold the personal information fo a few people.
"""
import employee


def print_employees(employees):
    """
    Print all employees in a nice tables.

    :param employees: List of employees.
    """
    print('| {:<30} '.format('Name') +
          '| {:^10}'.format('ID number') +
          '| {:^20} '.format('Department') +
          '| {:^20} |'.format('Job title'))
    print('|{:-^32}'.format('') +
          '|{:-^11}'.format('') +
          '|{:-^22}'.format('') +
          '|{:-^22}|'.format(''))

    # Print all employees.
    for single_emp in employees:
        print('| {:<30} '.format(single_emp.get_name()) +
              '| {:^10}'.format(single_emp.get_id()) +
              '| {:>20} '.format(single_emp.get_department()) +
              '| {:>20} |'.format(single_emp.get_title()))


def main():
    """
    Program main entry point.
    """
    # Create a list of employees.
    employees = list()
    # Populate 3 instances.
    employees.append(
        employee.Employee('Susan Meyers', '47899', 'Accounting',
                          'Vice President'))
    employees.append(
        employee.Employee('Mark Jones', '39119', 'IT', 'Programmer'))
    employees.append(
        employee.Employee('Joy Rogers', '81774', 'Manufacturing', 'Engineer'))
```

```
    # Print all employees.
    print_employees(employees)


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

employee.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 4 "Employee Class"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-03-01)

Class that encapsulates the properties of an employee.
"""


class Employee:
    """
    Employee class.
    """

    def __init__(self, name='John Doe', id='', department='', title=''):
        """
        Constructor.

        :param name: The name of the employee. Defaults to 'John Doe'.
        :param id: The ID number of the employee.
        :param department: The department that the employee is working in.
        :param title: The job title of the employee.
        """
        # Set to the values passed to the constructor.
        self.__name = name
        self.__id = id
        self.__department = department
        self.__title = title

    def get_name(self):
        """
        Get the name.

        :return: The name
        """
        return self.__name

    def get_id(self):
        """
        Get the id.

        :return: The id
        """
        return self.__id
```

```
    def get_department(self):
        """
        Get the department.

        :return: The department
        """
        return self.__department

    def get_title(self):
        """
        Get the title.

        :return: The title
        """
        return self.__title

    def set_name(self, name):
        """
        Set the name.
        """
        self.__name = name

    def set_id(self, id):
        """
        Set the id.
        """
        self.__id = id

    def set_department(self, department):
        """
        Set the department.
        """
        self.__department = department

    def set_title(self, title):
        """
        Set the title.
        """
        self.__title = title

    def __str__(self):
        """
        Return a string with the value of the year model and make.
        :return:
        """
        return ('Employee')
```

Result

```
| Name                      | ID number |    Department    |    Job title     |
|---------------------------|-----------|------------------|------------------|
| Susan Meyers              |   47899   |       Accounting |   Vice President |
| Mark Jones                |   39119   |               IT |       Programmer |
| Joy Rogers                |   81774   |    Manufacturing |         Engineer |
```

Conclusion

Classes are nice!