Assignment 12

Inheritance

LILLEBAELT ACADEMY OF
PROFESSIONAL HIGHER EDUCATION

Author
Martin Grønholdt
mart80c7@edu.eal.dk

Sunday 12 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 and inheritance is used. The code follows the convention from the Python documentation[1] prepending private members of classes with to underscores, 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/ass12

Error handling

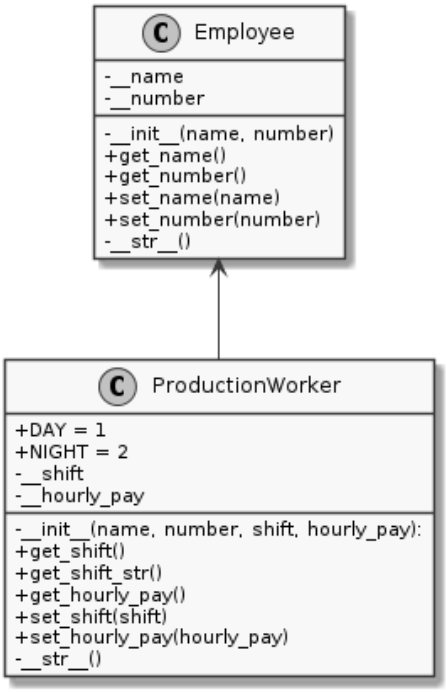
All programs that requests user input, handle bad input by asking the user, to use only the correct data type.

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. Employee and ProductionWorker Classes

This program tests a ProductionWorker class, that is inheriting from an Employee class.



Class and inheritance diagram for program 1.

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 "Employee and ProductionWorker Classes"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-02-12)

Program for entering details in to a ProductionWorker class.
"""
from ass12.productionworker import ProductionWorker


def main():
    """
    Program main entry point.
    """
    # Instantiate the ProductionWorker class.
    my_pw = ProductionWorker()

    try:
        # Use the setters, to set the values from user input.
        my_pw.set_name(input('Input the name of the employee: '))
```

```
            my_pw.set_number(int(input('Input the numer of the employee: ')))
            shift = str(input(
                'Input the shift the employee is assgined to (day, night): '))
            shift = shift.upper()
            # Validate the shift value.
            if shift in ['DAY', 'NIGHT']:
                if shift == 'DAY':
                    my_pw.set_shift(ProductionWorker.DAY)
                else:
                    my_pw.set_shift(ProductionWorker.NIGHT)
            else:
                raise ValueError
            my_pw.set_hourly_pay(
                int(input('Input the hourly pay of the employee: ')))

        except ValueError:
            # Complain when something unexpected was entered.
            print('\nThe input was wrong.')
            exit(1)

        print('\nYou entered: ')
        # There are cleaner ways, but I'm asked to use the setter/getter here.
        print(' Name:\t\t\t{}'.format(my_pw.get_name()))
        print(' Number:\t\t{}'.format(my_pw.get_number()))
        print(' Shift:\t\t\t{}'.format(my_pw.get_shift_str()))
        print(' Hourly pay:\t{}'.format(my_pw.get_hourly_pay()))


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

productionworker.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 "Employee and ProductionWorker Classes"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-02-12)

Class that encapsulates the properties of a production worker.
"""
from ass12.employee import Employee


class ProductionWorker(Employee):
    """
    ProductionWorker class.
    """
    DAY = 1
    NIGHT = 2

    def __init__(self, name='John Doe', number=0, shift=0, hourly_pay=0):
        """
        Constructor.

        :param name: The name of the employee. Defaults to 'John Doe'.
        :param number: The number of the employee.
        :param shift: The shift that the employee is on.
        :param hourly_pay: The hourly pay that the worker earns.
        """
        # Call the parent constructor.
        Employee.__init__(self, name, number)
        # Set to the values passed to the constructor.
        self.__shift = shift
        self.__hourly_pay = hourly_pay

    def get_shift(self):
        """
        Get the shift.

        :return: The shift
        """
        return self.__shift

    def get_shift_str(self):
        """
        Get the shift as a string.

        :return: The shift as a string.
        """
        ret = 'unknown'
        if self.__shift is ProductionWorker.DAY:
            ret = 'day'
        elif self.__shift is ProductionWorker.NIGHT:
            ret = 'night'

        return ret
```

```
    def get_hourly_pay(self):
        """
        Get the hourly pay.

        :return: The hourly pay
        """
        return self.__hourly_pay

    def set_shift(self, shift):
        """
        Set the shift.
        """
        self.__shift = shift

    def set_hourly_pay(self, hourly_pay):
        """
        Set the hourly pay.
        """
        self.__hourly_pay = hourly_pay

    def __str__(self):
        """
        Return a string representing the object.

        :return: string
        """
        return ('ProductionWorker "{}", number {}'.format(self.get_name(),
                                                          self.get_number()) +
                ', shift {}, hourly pay {}'.format(self.get_shift_str(),
                                                   self.get_hourly_pay()))
```

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 1 "Employee and ProductionWorker Classes"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-02-12)

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


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

    def __init__(self, name='John Doe', number=0):
        """
        Constructor.

        :param name: The name of the employee. Defaults to 'John Doe'.
        :param number: The number of the employee.
        """
```

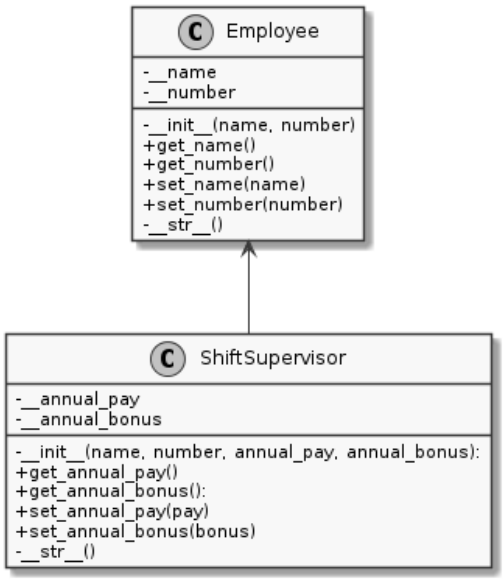
```
        # Set to the values passed to the constructor.
        self.__name = name
        self.__number = number

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

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

    def get_number(self):
        """
        Get the number.

        :return: The number
        """
        return self.__number

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

    def set_number(self, number):
        """
        Set the number.
        """
        self.__number = number

    def __str__(self):
        """
        Return a string representing the object.

        :return: string
        """
        return ('Employee "{}" number {}'.format(self.__name, self.__number))
```

Result

```
Input the name of the employee: Hans Hansen
Input the number of the employee: 500
Input the shift the employee is assigned to (day, night): day
Input the hourly pay of the employee: 500

You entered:
 Name:            Hans Hansen
 Number:          500
 Shift:           day
 Hourly pay:      500
```

2. ShiftSupervisor Class

This programs creates a ShiftSupervisor class from an Employee class using inheritance.



Class and inheritance diagram for program 2.

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 "ShiftSupervisor Class"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-02-12)

Program that tests the ShiftSupervisor class
"""
from ass12.shiftsupervisor import ShiftSupervisor


def main():
    """
    Program main entry point.
    """
    # Instantiate the ShiftSupervisor class.
    my_ss = ShiftSupervisor(name='Hans Hansen')

    #Set the values.
    my_ss.set_number(1)
    my_ss.set_annual_bonus(50000)
    my_ss.set_annual_pay(500000)

    # Print the data.
    print(str(my_ss))
```

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

shiftsupervisor.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 "ShiftSupervisor Class"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-02-12)

Class that encapsulates the properties of a shift supervisor.
"""
from ass12.employee import Employee


class ShiftSupervisor(Employee):
    """
    ShiftSupervsior class.
    """

    def __init__(self, name='John Doe', number=0, annual_pay=0, annual_bonus=0):
        """
        Constructor.

        :param name: The name of the employee. Defaults to 'John Doe'.
        :param number: The number of the employee.
        :param annual_sallery: Annual sallery of the employee.
        :param annual_bonus: Annual bonus of the employee.
        """
        # Call the parent constructor.
        Employee.__init__(self, name, number)
        # Set to the values passed to the constructor.
        self.__annual_pay = annual_pay
        self.__annual_bonus = annual_bonus

    def get_annual_pay(self):
        """
        Get the annual pay.

        :return: The annual pay
        """
        return self.__annual_pay

    def get_annual_bonus(self):
        """
        Get the annual bonus.

        :return: The annual bonus.
        """
        return self.__annual_bonus

    def set_annual_pay(self, pay):
        """
        Set the sannual pay.
```

```
        """
        self.__annual_pay = pay

    def set_annual_bonus(self, bonus):
        """
        Set the annual onus.
        """
        self.__annual_bonus = bonus

    def __str__(self):
        """
        Return a string representing the object.

        :return: string
        """
        return ('Shift supervisor "{}", number {}'.format(self.get_name(),
                                                          self.get_number()) +
                ', annual sallery {}'.format(self.get_annual_pay()) +
                ', annual bonus {}'.format(self.get_annual_bonus()))
```

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 1 "Employee and ProductionWorker Classes"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-02-12)

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


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

    def __init__(self, name='John Doe', number=0):
        """
        Constructor.

        :param name: The name of the employee. Defaults to 'John Doe'.
        :param number: The number of the employee.
        """
        # Set to the values passed to the constructor.
        self.__name = name
        self.__number = number

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

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

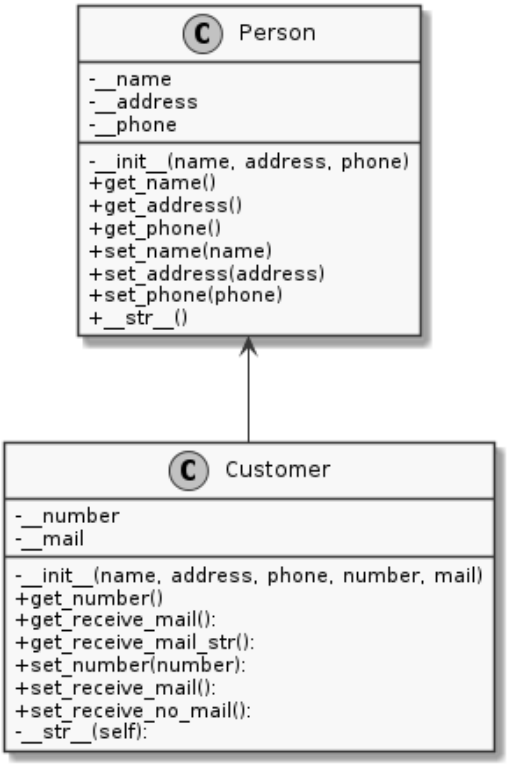
```
    def get_number(self):
        """
        Get the number.

        :return: The number
        """
        return self.__number

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

    def set_number(self, number):
        """
        Set the number.
        """
        self.__number = number

    def __str__(self):
        """
        Return a string representing the object.

        :return: string
        """
        return ('Employee "{}" number {}'.format(self.__name, self.__number))
```

Result

```
Shift supervisor "Hans Hansen", number 1, annual sallery 500000, annual bonus
50000
```

3. Person and Customer Classes

This program uses a base Person class to create a Customer class using inheritance. The Customer class is tested in the main program.



Class an inheritance diagram for program 3.

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 "Person and Customer Classes"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-02-12)

Program that uses the Customer class.
"""
from ass12.customer import Customer

def main():
    """
    Program main entry point.
    """
    # Create a list of people.
    new_cus = Customer()
```

```
    #Create a customer.
    new_cus.set_name('Hans Hansen')
    new_cus.set_address('Hans Hansensvej 555, 8210 Aarhus V')
    new_cus.set_phone('1231456')
    new_cus.set_number(500)
    new_cus.set_receive_mail()

    #Print.
    print(new_cus)

    print('\nRemoving customer form mailing list\n')

    new_cus.set_receive_no_mail()
    print(new_cus)

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

customer.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 (2017-02-12)

Class that encapsulates the properties of a customer.
"""
from ass12.person import Person


class Customer(Person):
    """
    Customer class.
    """
    def __init__(self, name='John Doe', address='', phone='', number=0,
                 mail=False):
        """
        Constructor.

        :param name: The name of the customer. Default is 'John Doe'.
        :param address: The address of the customer.
        :param phone: The phone number of the customer.
        :param number: The customer number.
        :param mail: True is the customer is on the mailing list.
        """
        # Call the parent constructor.
        Person.__init__(self, name, address, phone)
        # Set to the values passed to the constructor.
        self.__number = number
        self.__mail = mail

    def get_number(self):
        """
        Get the customer number.
```

```
        :return: The scustomer number.
        """
        return self.__number

    def get_receive_mail(self):
        """
        True if the customer is on the mailing list.

        :return: The mailing list status.
        """
        return self.__mail

    def get_receive_mail_str(self):
        """
        Return the customer the mailing list status. 'yes' if the customer
        receives mail, 'no' otherwise.

        :return: The mailing list status.
        """
        ret = False
        if self.__mail:
            ret = True
        return ret

    def set_number(self, number):
        """
        Set customer number.
        """
        self.__number = number

    def set_receive_mail(self):
        """
        Set the customer to receive mail.
        """
        self.__mail = True

    def set_receive_no_mail(self):
        """
        Set the customer to not receive mail.
        """
        self.__mail = False

    def __str__(self):
        """
        Return a string that is directly usable when printing the entry.

        :return: string
        """
        return('Name:\t\t\t{}\n'.format(self.get_name()) +
               'Address:\t\t{}\n'.format(self.get_address()) +
               'Phone:\t\t\t{}\n'.format(self.get_phone()) +
               'Number:\t\t\t{}\n'.format(self.get_number()) +
               'Mailing list:\t{}'.format(self.get_receive_mail_str())
               )
```

person.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 Person:
    """
    Class that holds personal information.
    """
    def __init__(self, name='John Doe', address='', phone=''):
        """
        Constuctor.

        :param name: The name of the person. Default is 'John Doe'.
        :param address: The address of the person.
        :param phone: The phone number of the person.
        """
        # Set to the values passed to the constructor.
        self.__name = name
        self.__address = address
        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_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_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) +
               'Address:\t{}\n'.format(self.__address) +
               'Phone:\t\t{}\n'.format(self.__phone))
```

Result

```
Name:               Hans Hansen
Address:            Hans Hansensvej 555, 8210 Aarhus V
Phone:              1231456
Number:             500
Mailing list:       True

Removing customer form mailing list

Name:               Hans Hansen
Address:            Hans Hansensvej 555, 8210 Aarhus V
Phone:              1231456
Number:             500
Mailing list:       False
```

Conclusion

Classes can be nicer and more powerful with inheritance!