Homework 2: Object-oriented Programming

Robert Litschko[*]
Symbolische Programmiersprache

Due: Wednesday, October 29, 2025, 12:00 (noon)

In this exercise you will:

- Practice creating simple classes and objects with Python.
- For working on this week's homework please look into your group's GitLab repository (`git pull`).

Exercise 1: Bank account class [5 points]

1. Using the slides & the script, put together a file containing the complete Account class. Each method must have a documentation string at the beginning which describes what the method is doing.
2. Create a main application where you create a number of accounts. Play around with depositing/withdrawing money. Change the account holder of an account using a setter method.
3. Change the withdraw function to ensure the account balance never exceeds **-500**. If a withdrawal attempt would breach this limit, return an appropriate error message to the user.
4. Write a function `apply_interest(self)` which allows account holders to choose between 3 different types of accounts:
   Standard Account: **1.2%** interest rate
   Gold Account: **1.7%** interest rate
   Platinum Account: **2.2%** interest rate
   Implement the `apply_interest(self)` function accordingly to apply the respective interest rates.
5. Implement the `__str__` magic method. The method should return a string with the account holder's name, current balance, and account type.

[*]Credit: Exercises are based on previous iterations from Katerina Kalouli.

Exercise 2: Employee class [6 points]

1. Write the complete code for the Employee class (including constructor, `__str__`,...)
2. Create a main application, create a few employee objects and show how you can manipulate them using the methods.
3. Create a department dictionary (dictionary of string to lists or sets of employees) with at least two departments (e.g. *HR*, *Engineering*,...) with each at least two employees. Print for each employee in the dictionary:
   <department> <employee name>.
4. Extend the Employee class by adding an attribute for salary and a method `give_raise(self, amount)`, which increases the employee's salary by the given amount. In your main application, demonstrate giving a raise to a selected employee and then printing out their updated salary.
5. Extend the Employee class by adding a method `calculate_bonus(self)` which calculates an annual bonus as 10% of the employee's salary. In your main application, calculate and display the bonus for each employee.
6. Extend the Employee class by adding an attribute "best_employee" with the default value False and a method `make_employee_of_the_month(self)` that changes the value for "best_employee" to True for an employee object.