Course Introduction

Advanced Course in Programming

3.11.2023

Outline

- 5 ECTs.
- 3.11. - 16.12.2021
- One "lecture" per week at Thursdays, 10.15
  - **Exception**: no lecture next Thursday (9.11.); instead lecture on Monday, Nov. 13th at 12.15)!
- Additional workshop times, list in MOOC.fi

- Focus on independent work

Lectures

Lectures

Date	Time
Th 2/11/2023	10:15 AM–12:00 PM
Mo 13/11/2023	12:15 PM–2:00 PM
Th 16/11/2023	10:15 AM–12:00 PM
Th 23/11/2023	10:15 AM–12:00 PM
Th 30/11/2023	10:15 AM–12:00 PM
Th 7/12/2023	10:15 AM–12:00 PM
Th 14/12/2023	10:15 AM–12:00 PM

Teachers

Erkki
Timo
Senior and junior instructors

- erkki.kaila@helsinki.fi - use email primarily
- ohjelmoinnin-mooc@helsinki.fi - concerning material (GIT issues or PR's work also nicely!)
- C213, Exactum (rarely)
- +358 50 3454 325 (if I remember to switch it on occasionally)

Contents

Week	Topic
8	Objects, references, writing own classes
9	More than one class, getters and setters, visibility, static members
10	Inheritance and class hierarchies
11	List comprehension, recursion
12	Generator functions, lambda, reduce, map, filter
13 to 14	Game programming with Pygame

Initial Exam

If you know programming already and haven't completed similar course with MOOC.fi or in any other Finnish higher institute, you can pass the course with initial exam

Exam takes place at Friday 3.11. between 12 and 14 online

Register via email erkki.kaila@helsinki.fi **before tomorrow morning** 08.00! I send the instructions by email **before the exam starts.**

What Do We Do in Course?

Two components:

1. Thursday (except one Monday) shared sessions (such as this)
2. Completing exercises in MOOC.fi

Finally, an exam

Workload

Learning to program requires a lot of work

Workload is around 10 to 20 h per week

...although with previous experience the first weeks may be completed faster

Course Material

All course material is found here

https://programming-23.mooc.fi/

Workshops

Exercises are completed independently, but you don't need to complete them alone

Help can be asked in workshops and in Discord, detailed information is found in MOOC.fi

Registrations

Go to https://programming-23.mooc.fi

If you're a student in University of Helsinki, use the @helsinki.fi email address. Remember to add your student number!

If you have already registered for Introduction to Programming, no need to do it again

Registration (cont.)

Remember to register to course in SISU

Course credits can't be assigned without registration

Deadlines

All rounds are already published

There are no weekly deadlines, all material closes at **January 4th 2024**

At least 25 % of points is required from each round to attend the exam

Passing the Course

Minimum requirements:

- 25 % of all points from each round
- 50 % of exam score

Tasks get harder during the course, so it is a good idea to try to complete as much as possible from the very beginning

Passing the Course (2)

Grade includes MOOC points (50 %) and exam score (50 %)

Total points	**Grade**
90%	5
80%	4
70%	3
60%	2
50%	1
<50%	Fail

Exams

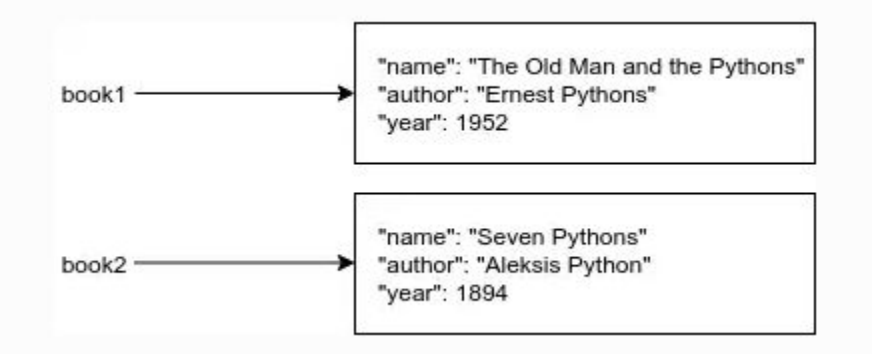
Three exams, dates are in the material. First exam at Saturday 16.12.2023

Done electronically with own computer; open between 10.00 and 22.00

You can attend as many exams as you like

Introduction to
Object-Oriented Programming

Objects are independent



Objects and methods

```
# this creates an object of type dictionary with the name book
book = {"name": "The Old Man and the Pythons", "author": "Ernest Pythons", "year": 1952}

# Print out all the values
# The method call values() is written after the name of the variable
# Remember the dot notation!
for value in book.values():
    print(value)
```

Creating objects

```
# Lists are declared with square brackets
my_list = [1,2,3]

# Strings are declared with quotation marks
my_string = "Hi there!"

# Dictionaries are declared with curly brackets
my_dict = {"one": 1, "two:": 2}

# Tuples are declared with parentheses
my_tuple = (1,2,3)
```

Constructor

```
# we are using the class Fraction from the module fractions
from fractions import Fraction

# create some new fraactions
half = Fraction(1,2)

third = Fraction(1,3)

another = Fraction(3,11)

# print these out
print(half)
print(third)
print(another)

# Fractions can be added together, for example
print(half + third)
```

Class and object

Class is the blueprint of the object

Classes define the **structure** and **functionality** of objects

It is (usually) possible to create several objects out of a single class

Defining own classes

Keyword **class:**

```
class NameOfClass:
    # class defition goes here
```

Naming classes

Usually named in *camel case* :

- Weekday
- BankAccount
- LibraryDatabase
- PythonCourseGrades

Adding a constructor

```python
class BankAccount:

    # The constructor
    def __init__(self, balance: float, owner: str):
        self.balance = balance
        self.owner = owner
```

Methods in own classes

```python
class BankAccount:

    def __init__(self, account_number: str, owner: str, balance: float, annual_interest:
        self.account_number = account_number
        self.owner = owner
        self.balance = balance
        self.annual_interest = annual_interest

    # This method adds the annual interest to the balance of the account
    def add_interest(self):
        self.balance += self.balance * self.annual_interest
```

Terminology

Client is the program code utilizing the class

Encapsulation means hiding the implementation from the client

Ensuring the **internal integrity** of the class

Outputting objects

Method __str__

The print function (for example) calls the method automatically

There's another method __repr__, which we will discuss later

Next week

More classes and objects

Visibility, better encapsulation

Properties, getters and setters

Static members