Name: Steve Hommy

Pair: -

Amount of completed tasks: 10

Which tasks were left undone or incomplete: 0

Self-assessment:

This exercise was easy for me because I have done these in advanced python course. Doing this exercise, I learned about some new methods. I understood everything.

Test report

Task	**Input / action**	**Desired output**	**Actual output (use red color if desired output != actual output)**
3	User inputs integer out of range, e.g. -10 or 121.	Exercise points: -19 Error: exercise points cannot be < 0 or > 120.	insert exercise points: -19 Error: exercise points cannot be < 0 or > 120.
3	User inputs a valid integer, e.g. 78.	Exercise points: 78 Your grade is: 2	Insert exercise points: 78 Grade 2
3	<test every grade 0-5 that the points vs. grade works correctly>	Exercise points: Your grade is:	Insert exercise points: Error: exercise points cannot be < 0 or > 120.
4	User inputs 2 students. <You can also have some other way to ask the amount of entries than the one presented on the right. Just modify the desired output accordingly.>	Give name: Sanna Give grade: 3 Want to input more students (Y/N): Y Give name: Anne Give grade: 5 Want to input more students (Y/N): N Average grade of students is: 4	Give name: Sanna Give grade: 3 Want to input more students (Y/N): y Give name: Anne Give grade: 5 Want to input more students (Y/N): n Sanna grade is 3 Anne grade is 5 Average grade of students is: 4.00
4	User inputs 3 students. Your own test cases here, add rows. Test at least with 0 students, multiple students, grades out of bounds (so error message is given) etc. Add own row for every test case.>	Give name: Santeri Give grade: 5 Want to input more students (Y/N): y Give name: Mikko Give grade: 2 Want to input more students (Y/N): y Give name: Miika Give grade: 3 Want to input more students (Y/N): n Santeri grade is 5 Mikko grade is 2 Miika grade is 3 Average grade of students is: 3.33 Give name: Miika Give grade: -1 Error: grade cannot be < 0 or > 5.	

		Want to input more students (Y/N): y Give name: Pekka Give grade: 3 Want to input more students (Y/N): gds Invalid input Want to input more students (Y/N): y Give name: Matti Give grade: 2 Want to input more students (Y/N): n Pekka grade is 3 Matti grade is 2 Average grade of students is: 2.50 Give name: Give grade: invalid literal for int() with base 10: ''	
6	User runs the program <Run the program a couple of times so that you get each side up at least once>	This side is up: Heads Tossing the coin… Now this side is up: Tails	This side is up: Heads Tossing the coin... Now this side is up: Tails
7	User runs the program <Run the program a couple of times so that you get every side up at least once.>	This side is up: Heads Tossing the coin… Now this side is up: Coin defies gravity and disappeared.	This side is up: Heads Tossing the coin... Coin defies gravity and gets lost on a wormhole in space This side is up: Heads Tossing the coin... Now this side is up: coin drops on the ground and disappears on a rabbit hole
10	User runs the program <Write test case depending on your implementation.>	Enter the hour you want to wake up at: 21 Enter the minute you want to wake up at: 02 Ring! Ring! Ring!	

1. Explain the following terms:

a. Pseudocode

Pseudocode is an informal and contrived way of writing programs in which you represent the sequence of actions and instructions in a form that humans can easily understand

b. Algorithm

With algorithm a programmer can easily solve a problem, because algorithms are expressed using natural verbal but somewhat technical annotations. Algorithm is an organized logical sequence of the actions or the approach towards a particular problem.

c. Data attribute

Data attribute is a single-value descriptor for a data point or data object. It exists most often as a column in a data table, but can also refer to special formatting or functionality for objects

d. Method

A method is somewhat like a function, except it is associated with object/classes. Methods are very similar to functions except for two major differences the method is implicitly used for an object for which it is called, and the method is accessible to data that is contained within the class.

2. Take a look at the course's assessment (points from exercises meaning certain grade). Write pseudocode for a program where user inputs the exercise points and program prints out the grade

This program where user inputs the exercise points and program prints out the grade.

Function grade_calculator() {

While True:

Prompt input Insert points. Check value error if points is less than 0 or more than 120 and when error occurs print out error message

If points is larger or 108 print Grade 5

Elif points is larger or 96 print Grade 4

Elif points is larger or 84 print Grade 3

Elif points is larger or 72 print Grade 2

Elif points is larger or 60 print Grade 1

Else:

Print Grade 0

Grade_calculator()

3. After writing the pseudocode, code task 2. Simple code is enough, no objects needed. Use informative and readable output prints.

```
def grade_calculator():
    while True:
        try:
            points = int(input("Insert exercise points: "))
            if points < 0 or points > 120:
                raise ValueError
            break
        except ValueError:
            print("Error: exercise points cannot be < 0 or > 120.")
    if points >= 108:
        print("Grade 5")
    elif points >= 96:
        print("Grade 4")
    elif points >= 84:
        print("Grade 3")
    elif points >= 72:
        print("Grade 2")
    elif points >= 60:
        print("Grade 1")
    else:
        print("Grade 0")

grade_calculator()
```

4. Write pseudocode for a program that accepts student's name and grade as input and counts the average of grades of all students. If you have difficulties, you can fix the number of students to e.g. 5. Print out the average.

This program accepts student's name and grade as input and counts the average of grades of all students.

Function students():

List for student name

List for student grade

While True:

Input if you want to add more student Y or N

If given Wrong input exception as error

If input is Y

Input student name

Input student grade

If grade is lower than 0 or larger than 5

Raise value error

Append name into name list

Append grade into grade list

Elif input is N

Calculate average grade

Print average grade

Else

Print invalid input

Students()

5. After writing the pseudocode, code task 4. Simple code is enough, no objects needed. Use informative and readable output prints.

```
def students():
    student_list = []
    grade_list = []
    while True:
        # The lower() method returns a string where all characters are lower case
        # the strip() method remove spaces at the beginning and at the end of the string
        add_student = str(input("Want to input more students (Y/N): ")).lower().strip()
        try:
            if add_student[0] == "y":
                student_name = input("Give name: ")
                student_grade = int(input("Give grade: "))
                try:       You, seconds ago • Uncommitted changes
                    if student_grade < 0 or student_grade > 5:
                        raise ValueError
                    student_list.append(student_name)
                    grade_list.append(student_grade)
                except ValueError:
                    print("Error: grade cannot be < 0 or > 5.")
            elif add_student[0] == "n":
                average = sum(grade_list) / len(student_list)
                # the zip() brings elements of the same index from multiple iterable
                # objects together as elements of the same tuples
                for name, grade in zip(student_list, grade_list):
                    print(name + " grade is", grade)
                    # Formating average grade into 2 decimal
                print("Average grade of students is: " + " {:.2f}".format(average))
                break
            else:
                print("Invalid input")
        except Exception as error:
            print("Please input Y or N: ")
            print(error)

students()
```

8. Imagine you would have to code a simple alarm clock (shows time and alarms you at certain time you can set). Which data attributes will you have? Do the attributes have some value restrictions? You should find at least 5 data attributes. Which methods would you need? Which methods should be public and which ones should be private?

Data attributes would be hour, minute, second, current time and alarm. Restrictions for time would be that they must be integers. Methods would be set alarm, show current time and alarm sound. Methods that would be public is setting alarm and shutting the alarm. Methods that would be private is current time.

9. Write pseudocode for the alarm clock (see task 6). Notice, this is a simple alarm clock. This is a simple task, do not make it unnecessarily complicated! For example, you can start with a simple clock without the alarm. If this gets too complicated, write pseudocode for a timer instead of an alarm clock (each round in a loop increases the seconds with 1 etc.).

Object that works as a simple alarm clock

Import time

Class Alarm():

Function __init__(self, hours, minutes):

Self.hours = value for hours

Self.minutes = value for minutes

Self.keep_running = True

Function for setting up the timer set_timer

While the alarm is running

Get localtime using time module

If hours and minutes matches localtime print Ring!

Return

Main function

input for user in what hour time the alarm will go off

input for user in what minute time the alarm will go off

insert these two inputs into Alarm object

call the set_timer function

main()

10. Code the alarm clock, use objects. This is also supposed to be a simple task, so do not make it unnecessarily complicated. Alarm can simply be a text output on screen. No sounds are needed (can be added if you wish). If this is too difficult, code a timer instead.

```
# Importing time module for localtime
import time

# Defining class
class Alarm():
    # __init__ allow the class to initialize the attributes of a class
    # Using the “self” keyword we can access the attributes and methods of the class in python
    def __init__(self, hours, minutes):
        self.hours = int(hours)
        self.minutes = int(minutes)
        self.keep_running = True

    # Checks if localtime equals the input time
    def set_timer(self):
        while self.keep_running:
            current_time = time.localtime()
            if (current_time.tm_hour == self.hours and current_time.tm_min == self.minutes):
                print("Ring! Ring! Ring!")
                return


# Defining main function
def main():
    set_hour = input("Enter the hour you want to wake up at: ")
    set_minute = input("Enter the minute you want to wake up at: ")
    alarm = Alarm(set_hour, set_minute)
    alarm.set_timer()
main()
```