ADVANCED COURSE IN PROGRAMMING, AUTUMN 2022, ONLINE EXAM 1

STARTED: 08:41 ENDS: 12:41 3:59

Exercise 1

Complete this in exercise template `exercise1.py`

a)

Write a class `Question`, which should contain the following members:

- Hidden attributes `question` (string) and `maximum_points` (integer)
- A constructor, which takes question and maximum points as its parameters, in this given order and sets the values of the attributes according to these
- Getter and setter methods for accessing the attributes
- Method `__str__`, which returns question data as in the sample output

b)

Write a class `Exam`, which should contain the following members:

- Public attributes `subject` (string) and `date` (string)
- Method `add_question(question: Question)`, which adds new question to the exam
- Method `print_questions()`, which prints the exam information, and all the questions one below the other
- Method `total_points()`, which returns the total number of the points (the maximum points for all questions combined) for the whole exam as an integer

An example of the usage of the classes:

```
q1 = Question("When was the Olympics held in Helsinki", 10)
q2 = Question("when did Finland become independent", 5)

print(q1)

exam = Exam("History", "1.12.2021")
exam.add_question(q1)
exam.add_question(q2)
```

```
exam.print_questions()
print("Maximum points of the exam:", exam.total_points())
```

Sample output:

```
When was the Olympics held in Helsinki, 10 points
Exam on History, questions:
When was the Olympics held in Helsinki, 10 points
when did Finland become independent, 5 points
Maximum points of the exam: 15
```

Exercise 2

Complete this in exercise template `exercise2.py`

Copy and paste the following class to the exercise template:

```
class BankAccount:
    def __init__(self, customer: str, account_number: str,
saldo: float, credit: float):
        self.customer = customer
        self.account_number = account_number
        self.saldo = saldo
        self.credit = credit

    def __str__(self):
        return (f"BankAccount(customer={self.customer}, " +
            f"account_number={self.account_number}, " +
            f"saldo={self.saldo}, credit={self.credit})")
```

Add the following new methods to the class:

- Implementation for the operator ==: two accounts are equal, if account numbers are the same
- Implementation for the operator >: account a is greater than account b, if the total saldo and credit of account a is greater than the total saldo and credit of account b
- Implementation for the operator +: method returns new BankAccount-object, whose customer and account number are retrieved from the current (self) object. The saldo of new account is sum of the saldo of both accounts. Credit is calculated in the same way.

Example of usage of the class after additions:

```
account1 = BankAccount("Peter Python", "12345", 200.0,
300.0)
account2 = BankAccount("Paula Python", "12345", 100.0,
500.0)
print(account1 == account2)
print(account1 > account2)
print(account2 > account1)
```

```
pt3 = account1 + account2
print(pt3)
```

Sample output:

```
True
False
True
BankAccount(customer=Peter Python, account_number=12345,
saldo=300.0, credit=800.0)
```

Exercise 3

Complete this in exercise template `exercise3.py`

It is allowed to add import-statements to the exercise template. Modules and functions imported with import statements can be used inside the function requested to be written.

Write a generator function named `draw`, which:

- Takes a list as its parameter
- Returns random item from the list given as parameter

The function must not return the same item twice. When requesting the next item from the generator, the function must raise the `StopIteration` exception, if the list given as a parameter is empty or all items in the list have been returned once.

An example run of the function:

```
animal_generator = draw(['Chicken', 'Bear', 'Gorilla',
'Fox'])
for i in range(4):
    print(next(animal_generator))
```

Sample output:

```
Bear
Fox
Chicken
Gorilla
```

ABOUT MOOC CENTER

The MOOC Center creates custom online courses for the University of Helsinki. It's behind the highly popular courses that have been available in mooc.fi from 2012. The platform for the courses has been developed in-house by teams comprising both university employees and students.

RESOURCES

Privacy