MIDTERM

EXAM 2

FACULTY	DEPARTMENT
COMPUTER AND INFORMATICS	ARTIFICIAL INTELLIGENCE AND DATA ENGINEERING
COURSE	**INSTRUCTOR**
YZV104E – INTRO. TO PROG. FOR DATA SCI. (PYTHON)	A. CÜNEYD TANTUĞ
DATE	**HOURS**
02.05.2024	13:00 – 15:15

Exam Rules

1. This is a computer-based exam. You are requested to develop programs satisfying the question requirements.
2. You can only access Ninova website content during the exam.
3. Using external sources from any other device (such as cellular phones or tablets) with internet connectivity is strictly prohibited. Such an attempt will be treated as a cheating incident and disciplinary actions will be taken.
4. An official offline Python documentation in HTML format will be available during the exam. You should download the zipped documentation file from the Course Files section on Ninova.
5. You have to submit your solutions by using Ninova. Four assignments named "**Midterm 2**", "**Midterm 2 Question 1**", "**Midterm 2 Question 2**", and "**Midterm 2 Question 3**" will be activated at the beginning of the exam. **These assignments will be deactivated when the exam time is up, so you will NOT be able to submit your results anymore**.
6. To avoid possible timing problems during submission, please upload your solutions to Ninova homework when it is 10 minutes before the deadline of the exam (even though it is not your final version).
7. Your submissions will be evaluated in two phases. In the first phase, your programs will be automatically tested and scored by a script with a private set of test cases. In the second phase, a human review will be conducted to avoid any falsifying attempts that aim to trick the script.
8. Codes that cannot be executed **will not be graded**. The evaluation will be conducted with the same environment you use in the laboratory computer, so do not change the default environment while working. No further changes (even very minor) are allowed after midterm time is up. Your score will be proportional to the number of positive test cases divided by the number of all test cases. Bonus points (if there exists any) will be added to your basic score.
9. As a general programming principle, do not start coding immediately. Take a moment to understand and grasp the problem, think about possible solution steps (which is the algorithm), and then start your algorithm implementation.

QUESTIONS

1) [X points] In the realm of finance management, savings accounts play a pivotal role in personal financial planning. Your task in this question is to develop a Python program that manages savings accounts. The program should consist of two classes: Account and SavingsAccount.

- The Account class is responsible for representing a generic bank account. It should have the following attributes:
  - account_number: A unique identifier for the account.
  - balance: The current balance of the account.
  - account_holder_name: The name of the account holder.
- The class should implement the following methods:
  - deposit(amount): Add the specified amount to the account balance.
    - You should print the message if the deposit is successfully made: "Deposited $X into account account_number."
    - You should print "Invalid deposit amount." if the amount is not a number.
  - withdraw(amount): Deduct the specified amount from the account balance, if sufficient funds are available.
    - You should print the message if the withdrawal is successfully done: "Withdrew $X from account account_number."
    - You should print the message "Insufficient funds." if the amount is bigger than the balance.
    - You should print "Invalid deposit amount." if the amount is not a number.
  - __str__(): Return a string representation of the account, including the account number, account holder's name, and current balance with the following format. (each piece of information is on the new line)
    -
```
Account Number: Account12345
Account Holder: Customer1
Balance: $1000.00
```

- The SavingsAccount class inherits from the Account class and adds an additional attribute:
    - interest_rate: The annual interest rate for the account.
- It should implement the following methods:
    - calculate_interest(): Calculate and add interest to the account balance based on the interest rate with the following formula:
        - $Balance = Balance + \frac{Balance \times interest\ rate}{100}$
    - __str__(): Override the __str__() method of the Account class to include the interest rate. You should add the below information:
    -
```
Interest Rate: 30.0%
```

WARNING

When printing the account balances, make sure to **print with 2 floating points** to ensure consistency with the test cases.
[i.e. 150 as 150.00 12400.12345 as 12400.12]

WARNING

Please use the template *bank_account.py* provided with the exam package and modify this file to construct your solution. Note that you should NOT change the lines below
`if __name__ == "__main__":` line in the template.