(https://learningspoons.com/course/detail/pythonforfinance/)

검색어를 입력하세요.

271 ~ 280

271 Account 클래스

은행에 가서 계좌를 개설하면 은행이름, 예금주, 계좌번호, 잔액이 설정됩니다. Account 클래스를 생성한 후 생성자를 구현해보세요. 생성자에서는 예금주와 초기 잔액만 입력 받습니다. 은행이름은 SC은행으로 계좌번호는 3자리-2자리-6자리 형태로 랜덤하게 생성됩니다.

```
은행이름: SC은행
계좌번호: 111-11-111111
```

▼ 정답확인

```
import random

class Account:
    def __init__(self, name, balance):
        self.name = name
        self.balance = balance
        self.bank = "SC은행"
        num1 = random.randint(0, 999)
        num2 = random.randint(0, 99)
        num3 = random.randint(0, 999999)

        num1 = str(num1).zfill(3)      # 1 -> '1' -> '001'
        num2 = str(num2).zfill(2)      # 1 -> '1' -> '01'
        num3 = str(num3).zfill(6)      # 1 -> '1' -> '0000001'
        self.account_number = num1 + '-' + num2 + '-' + num3  # 001-01-000001

kim = Account("김민수", 100)
print(kim.name)
print(kim.balance)
print(kim.bank)
print(kim.account_number)
```

272 클래스 변수

클래스 변수를 사용해서 Account 클래스로부터 생성된 계좌 객체의 개수를 저장하세요.

▼ 정답확인

```
import random

class Account:
    # class variable
    account_count = 0

    def __init__(self, name, balance):
        self.name = name
        self.balance = balance
        self.bank = "SC은행"

        num1 = random.randint(0, 999)
        num2 = random.randint(0, 99)
        num3 = random.randint(0, 999999)

        num1 = str(num1).zfill(3)      # 1 -> '1' -> '001'
        num2 = str(num2).zfill(2)      # 1 -> '1' -> '01'
        num3 = str(num3).zfill(6)      # 1 -> '1' -> '0000001'
        self.account_number = num1 + '-' + num2 + '-' + num3  # 001-01-000001

        Account.account_count += 1


kim = Account("김민수", 100)
print(Account.account_count)
lee = Account("이민수", 100)
print(Account.account_count)
```

273 클래스 변수 출력

Account 클래스로부터 생성된 계좌의 개수를 출력하는 get_account_num() 메서드를 추가하세요.

▼ 정답확인

```
import random

class Account:
    # class variable
    account_count = 0

    def __init__(self, name, balance):
        self.name = name
        self.balance = balance
        self.bank = "SC은행"

        # 3-2-6
        num1 = random.randint(0, 999)
        num2 = random.randint(0, 99)
        num3 = random.randint(0, 999999)

        num1 = str(num1).zfill(3)      # 1 -> '1' -> '001'
        num2 = str(num2).zfill(2)      # 1 -> '1' -> '01'
        num3 = str(num3).zfill(6)      # 1 -> '1' -> '0000001'
        self.account_number = num1 + '-' + num2 + '-' + num3  # 001-01-000001
        Account.account_count +=1

    @classmethod
    def get_account_num(cls):
        print(cls.account_count)     # Account.account_count


kim = Account("김민수", 100)
lee = Account("이민수", 100)
kim.get_account_num()
```

274 입금 메서드

Account 클래스에 입금을 위한 deposit 메서드를 추가하세요. 입금은 최소 1원 이상만 가능합니다.

▼ 정답확인

```
import random

class Account:
    # class variable
    account_count = 0

    def __init__(self, name, balance):
        self.name = name
        self.balance = balance
        self.bank = "SC은행"

        # 3-2-6
        num1 = random.randint(0, 999)
        num2 = random.randint(0, 99)
        num3 = random.randint(0, 999999)

        num1 = str(num1).zfill(3)      # 1 -> '1' -> '001'
        num2 = str(num2).zfill(2)      # 1 -> '1' -> '01'
        num3 = str(num3).zfill(6)      # 1 -> '1' -> '0000001'
        self.account_number = num1 + '-' + num2 + '-' + num3  # 001-01-000001
        Account.account_count +=1

    @classmethod
    def get_account_num(cls):
        print(cls.account_count)     # Account.account_count

    def deposit(self, amount):
        if amount >= 1:
            self.balance += amount
```

275 출금 메서드

Account 클래스에 출금을 위한 withdraw 메서드를 추가하세요. 출금은 계좌의 잔고 이상으로 출금할 수는 없습니다.

▼ 정답확인

```
import random


class Account:
    # class variable
    account_count = 0

    def __init__(self, name, balance):
        self.name = name
        self.balance = balance
        self.bank = "SC은행"

        # 3-2-6
        num1 = random.randint(0, 999)
        num2 = random.randint(0, 99)
        num3 = random.randint(0, 999999)

        num1 = str(num1).zfill(3)  # 1 -> '1' -> '001'
        num2 = str(num2).zfill(2)  # 1 -> '1' -> '01'
        num3 = str(num3).zfill(6)  # 1 -> '1' -> '0000001'
        self.account_number = num1 + '-' + num2 + '-' + num3  # 001-01-000001
        Account.account_count += 1

    @classmethod
    def get_account_num(cls):
        print(cls.account_count)  # Account.account_count

    def deposit(self, amount):
        if amount >= 1:
            self.balance += amount

    def withdraw(self, amount):
        if self.balance > amount:
            self.balance -= amount

k = Account("kim", 100)
k.deposit(100)
k.withdraw(90)
print(k.balance)
```

276 정보 출력 메서드

Account 인스턴스에 저장된 정보를 출력하는 display_info() 메서드를 추가하세요. 잔고는 세자리 마다 쉼표를 출력하세요.

```
은행이름: SC은행
예금주: 파이썬
계좌번호: 111-11-111111
잔고: 10,000원
```

▼ 정답확인

```
import random


class Account:
    # class variable
    account_count = 0

    def __init__(self, name, balance):
        self.name = name
        self.balance = balance
        self.bank = "SC은행"

        # 3-2-6
        num1 = random.randint(0, 999)
        num2 = random.randint(0, 99)
        num3 = random.randint(0, 999999)

        num1 = str(num1).zfill(3)  # 1 -> '1' -> '001'
        num2 = str(num2).zfill(2)  # 1 -> '1' -> '01'
        num3 = str(num3).zfill(6)  # 1 -> '1' -> '0000001'
        self.account_number = num1 + '-' + num2 + '-' + num3  # 001-01-000001
        Account.account_count += 1

    @classmethod
    def get_account_num(cls):
        print(cls.account_count)  # Account.account_count

    def deposit(self, amount):
        if amount >= 1:
            self.balance += amount

    def withdraw(self, amount):
        if self.balance > amount:
            self.balance -= amount

    def display_info(self):
        print("은행이름: ", self.bank)
        print("예금주: ", self.name)
        print("계좌번호: ", self.account_number)
        print("잔고: ", f"{self.balance:,}")


p = Account("파이썬", 10000)
p.display_info()
```

277 이자 지급하기

입금 횟수가 5회가 될 때 잔고를 기준으로 1%의 이자가 잔고에 추가되도록 코드를 변경해보세요.

▼ 정답확인

```
import random


class Account:
    # class variable
    account_count = 0

    def __init__(self, name, balance):
        self.deposit_count = 0

        self.name = name
        self.balance = balance
        self.bank = "SC은행"

        # 3-2-6
        num1 = random.randint(0, 999)
        num2 = random.randint(0, 99)
        num3 = random.randint(0, 999999)

        num1 = str(num1).zfill(3)  # 1 -> '1' -> '001'
        num2 = str(num2).zfill(2)  # 1 -> '1' -> '01'
        num3 = str(num3).zfill(6)  # 1 -> '1' -> '0000001'
        self.account_number = num1 + '-' + num2 + '-' + num3  # 001-01-000001
        Account.account_count += 1

    @classmethod
    def get_account_num(cls):
        print(cls.account_count)  # Account.account_count

    def deposit(self, amount):
        if amount >= 1:
            self.balance += amount

            self.deposit_count += 1
            if self.deposit_count % 5 == 0:         # 5, 10, 15
                # 이자 지금
                self.balance = (self.balance * 1.01)


    def withdraw(self, amount):
        if self.balance > amount:
            self.balance -= amount

    def display_info(self):
        print("은행이름: ", self.bank)
        print("예금주: ", self.name)
        print("계좌번호: ", self.account_number)
        print("잔고: ", self.balance)

p = Account("파이썬", 10000)
p.deposit(10000)
```

```
p.deposit(10000)
p.deposit(10000)
p.deposit(5000)
p.deposit(5000)
print(p.balance)
```

---

278 여러 객체 생성

Account 클래스로부터 3개 이상 인스턴스를 생성하고 생성된 인스턴스를 리스트에 저장해보세요.

▼ 정답확인

```
import random


class Account:
    # class variable
    account_count = 0

    def __init__(self, name, balance):
        self.deposit_count = 0

        self.name = name
        self.balance = balance
        self.bank = "SC은행"

        # 3-2-6
        num1 = random.randint(0, 999)
        num2 = random.randint(0, 99)
        num3 = random.randint(0, 999999)

        num1 = str(num1).zfill(3)  # 1 -> '1' -> '001'
        num2 = str(num2).zfill(2)  # 1 -> '1' -> '01'
        num3 = str(num3).zfill(6)  # 1 -> '1' -> '0000001'
        self.account_number = num1 + '-' + num2 + '-' + num3  # 001-01-000001
        Account.account_count += 1

    @classmethod
    def get_account_num(cls):
        print(cls.account_count)  # Account.account_count

    def deposit(self, amount):
        if amount >= 1:
            self.balance += amount

            self.deposit_count += 1
            if self.deposit_count % 5 == 0:         # 5, 10, 15
                # 이자 지금
                self.balance = (self.balance * 1.01)


    def withdraw(self, amount):
        if self.balance > amount:
            self.balance -= amount

    def display_info(self):
        print("은행이름: ", self.bank)
        print("예금주: ", self.name)
        print("계좌번호: ", self.account_number)
        print("잔고: ", self.balance)

data = []
k = Account("KIM", 10000000)
```

```
l = Account("LEE", 10000)
p = Account("PARK", 10000)

data.append(k)
data.append(l)
data.append(p)

print(data)
```

---

279 객체 순회

반복문을 통해 리스트에 있는 객체를 순회하면서 잔고가 100만원 이상인 고객의 정보만 출력하세요.

▼ 정답확인

```
import random


class Account:
    # class variable
    account_count = 0

    def __init__(self, name, balance):
        self.deposit_count = 0

        self.name = name
        self.balance = balance
        self.bank = "SC은행"

        # 3-2-6
        num1 = random.randint(0, 999)
        num2 = random.randint(0, 99)
        num3 = random.randint(0, 999999)

        num1 = str(num1).zfill(3)  # 1 -> '1' -> '001'
        num2 = str(num2).zfill(2)  # 1 -> '1' -> '01'
        num3 = str(num3).zfill(6)  # 1 -> '1' -> '0000001'
        self.account_number = num1 + '-' + num2 + '-' + num3  # 001-01-000001
        Account.account_count += 1

    @classmethod
    def get_account_num(cls):
        print(cls.account_count)  # Account.account_count

    def deposit(self, amount):
        if amount >= 1:
            self.balance += amount

            self.deposit_count += 1
            if self.deposit_count % 5 == 0:         # 5, 10, 15
                # 이자 지금
                self.balance = (self.balance * 1.01)


    def withdraw(self, amount):
        if self.balance > amount:
            self.balance -= amount

    def display_info(self):
        print("은행이름: ", self.bank)
        print("예금주: ", self.name)
        print("계좌번호: ", self.account_number)
        print("잔고: ", self.balance)

data = []
k = Account("KIM", 10000000)
```

```
l = Account("LEE", 10000)
p = Account("PARK", 10000)
data.append(k)
data.append(l)
data.append(p)

for c in data:
    if c.balance >= 1000000:
        c.display_info()
```

---

280 입출금 내역

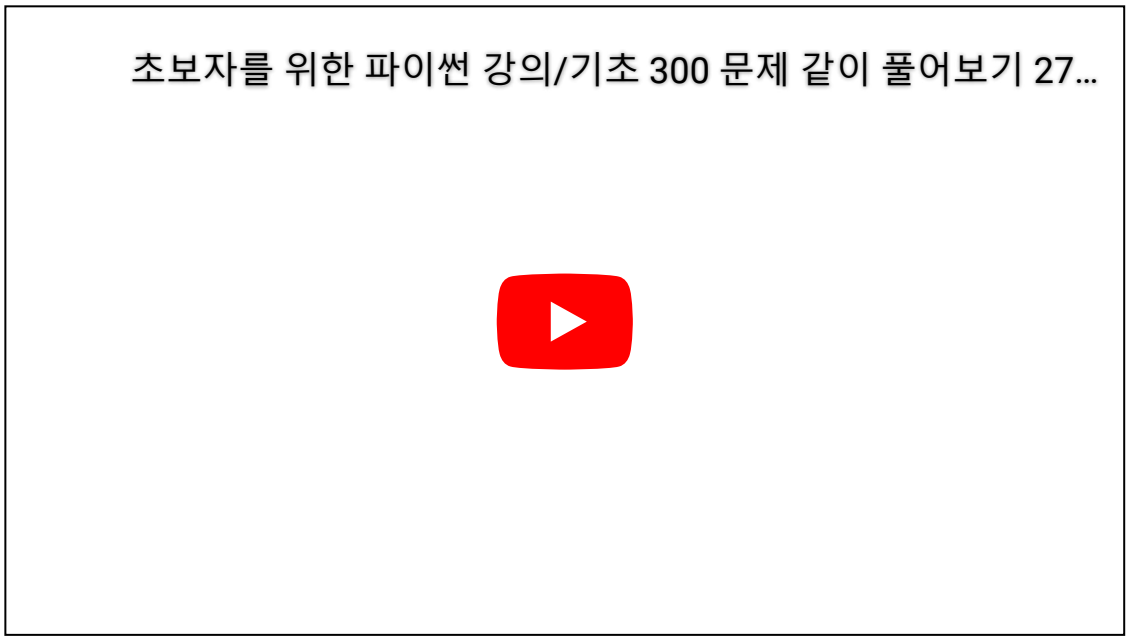
입금과 출금 내역이 기록되도록 코드를 업데이트 하세요. 입금 내역과 출금 내역을 출력하는 deposit_history와 withdraw_history 메서드를 추가하세요.

▼ 정답확인

```
import random


class Account:
    # class variable
    account_count = 0

    def __init__(self, name, balance):
        self.deposit_count = 0
        self.deposit_log = []
        self.withdraw_log = []

        self.name = name
        self.balance = balance
        self.bank = "SC은행"

        # 3-2-6
        num1 = random.randint(0, 999)
        num2 = random.randint(0, 99)
        num3 = random.randint(0, 999999)

        num1 = str(num1).zfill(3)  # 1 -> '1' -> '001'
        num2 = str(num2).zfill(2)  # 1 -> '1' -> '01'
        num3 = str(num3).zfill(6)  # 1 -> '1' -> '0000001'
        self.account_number = num1 + '-' + num2 + '-' + num3  # 001-01-000001
        Account.account_count += 1

    @classmethod
    def get_account_num(cls):
        print(cls.account_count)  # Account.account_count

    def deposit(self, amount):
        if amount >= 1:
            self.deposit_log.append(amount)
            self.balance += amount

            self.deposit_count += 1
            if self.deposit_count % 5 == 0:         # 5, 10, 15
                # 이자 지금
                self.balance = (self.balance * 1.01)


    def withdraw(self, amount):
        if self.balance > amount:
            self.withdraw_log.append(amount)
            self.balance -= amount

    def display_info(self):
        print("은행이름: ", self.bank)
        print("예금주: ", self.name)
        print("계좌번호: ", self.account_number)
```

```
            print("잔고: ", self.balance)

        def withdraw_history(self):
            for amount in self.withdraw_log:
                print(amount)

        def deposit_history(self):
            for amount in self.deposit_log:
                print(amount)


    k = Account("Kim", 1000)
    k.deposit(100)
    k.deposit(200)
    k.deposit(300)
    k.deposit_history()

    k.withdraw(100)
    k.withdraw(200)
    k.withdraw_history()
```

풀이 영상



마지막 편집일시 : 2022년 11월 15일 1:43 오후

(https://pyquant.co.kr/product/system-trading-with-tick-data/)

댓글 4 피드백

- **이전글** : 261 ~ 290
- **다음글** : 281 ~ 290

TOP