파이썬 문법 익히기

연습문제

11. 파이썬 클래스

251. 클래스, 객체, 인스턴스

▶ 클래스, 객체, 인스턴스에 대해 설명해봅시다.

252. 클래스 정의

▶ 비어있는 사람 (Human) 클래스를 "정의" 해보세요.

253. 인스턴스 생성

▶ 사람 (Human) 클래스의 인스턴스를 "생성" 하고 이를 areum 변수로 할당해보세요.

254. 클래스 생성자-1

▶ 사람 (Human) 클래스에 "응애응애"를 출력하는 생성자를 추가하세요.

```
>>> areum = Human()
응애응애
```

255. 클래스 생성자-2

▶ 사람 (Human) 클래스에 (이름, 나이, 성별)을 받는 생성자를 추가하세요.

```
>>> areum = Human("아름", 25, "여자")
```

256. 인스턴스 속성에 접근

▶ 255에서 생성한 인스턴스의 이름, 나이, 성별을 출력하세요. 인스턴스 변수에 접근하여

값을 출력하면 됩니다.

이름: 조아름, 나이: 25, 성별: 여자
인스턴스 변수에 접근하여 값을 가져오는 예

```
>>> areum.age
25
```

257. 클래스 메소드 - 1

▶ 사람 (Human) 클래스에서 이름, 나이, 성별을 출력하는 who() 메소드를 추가하세요.

```
>>> areum.who()
이름: 조아름, 나이: 25, 성별: 여자
```

258. 클래스 메소드 - 2

▶ 사람 (Human) 클래스에 (이름, 나이, 성별)을 받는 setInfo 메소드를 추가하세요.

```
>>> areum = Human("모름", 0, "모름")
>>> areum.setInfo("아름", 25, "여자")
```

259. 클래스 소멸자

▶ 사람 (human) 클래스에 "나의 죽음을 알리지 말라"를 출력하는 소멸자를 추가하세요.

```
>>> areum = Human("아름", 25, "여자")
>>> del areum
나의 죽음을 알리지 말라
```

260. 에러의 원인

▶ 아래와 같은 에러가 발생한 원인에 대해 설명하세요.

```
class OMG :
```

```
    def print() :
        print("Oh my god")

>>> myStock = OMG()
>>> myStock.print()
TypeError Traceback (most recent call last)
<ipython-input-233-c85c04535b22> in <module>()
----> myStock.print()

TypeError: print() takes 0 positional arguments but 1 was given
```

▷ 정답확인

```
class OMG :
    def print() :
        print("Oh my god")



mystock = OMG()
mystock.print()      # OMG.print(mystock)
```

261. Stock 클래스 생성

▶ 주식 종목에 대한 정보를 저장하는 **Stock** 클래스를 정의해보세요. 클래스는 속성과 메서드를 갖고 있지 않습니다.

262. 생성자

▶ **Stock** 클래스의 객체가 생성될 때 종목명과 종목코드를 입력 받을 수 있도록 생성자를 정의해보세요.

```
삼성 = Stock("삼성전자", "005930")
```

263. 메서드

▶ 객체에 종목명을 입력할 수 있는 **set_name** 메서드를 추가해보세요.

```
a = Stock(None, None)
a.set_name("삼성전자")
```

264. 메서드

▶ 객체에 종목코드를 입력할 수 있는 **set_code** 메서드를 추가해보세요.

```
a = Stock(None, None)
a.set_code("005930")
```

265. 메서드

▶ 종목명과 종목코드를 리턴하는 **get_name, get_code** 메서드를 추가하세요. 해당 메서드를 사용하여 종목명과 종목코드를 얻고 이를 출력해보세요.

```
삼성 = Stock("삼성전자", "005930")
```

266. 객체의 속성값 업데이트

▶ 생성자에서 종목명, 종목코드, **PER, PBR,** 배당수익률을 입력 받을 수 있도록 생성자를 수정하세요. **PER, PBR,** 배당수익률은 **float** 타입입니다.

267. 객체 생성

▶ **266**번에서 정의한 생성자를 통해 다음 정보를 갖는 객체를 생성해보세요.

```
항목   정보
종목명 삼성전자
종목코드      005930
PER   15.79
PBR   1.33
```

배당수익률 2.83

268. 객체의 속성 수정

▶ PER, PBR, 배당수익률은 변경될 수 있는 값입니다. 이 값을 변경할 때 사용하는 set_per, set_pbr, set_dividend 메서드를 추가하세요.

269. 객체의 속성 수정

▶ 267번에서 생성한 객체에 set_per 메서드를 호출하여 per 값을 12.75로 수정해보세요.

270. 여러 종목의 객체 생성

▶ 아래의 표를 참조하여 3종목에 대해 객체를 생성하고 이를 파이썬 리스트에 저장하세요. 파이썬 리스트에 저장된 각 종목에 대해 for 루프를 통해 종목코드와 PER을 출력해보세요.

종목명	종목코드	PER	PBR	배당수익률
삼성전자	005930	15.79	1.33	2.83
현대차	005380	8.70	0.35	4.27
LG전자	066570	317.34	0.69	1.37

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

272. 클래스 변수

▶ 클래스 변수를 사용해서 **Account** 클래스로부터 생성된 계좌 객체의 개수를 저장하세요.

273. 클래스 변수 출력

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

274. 입금 메서드

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

275. 출금 메서드

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

276. 정보 출력 메서드

▶ **Account** 인스턴스에 저장된 정보를 출력하는 **display_info()** 메서드를 추가하세요. 잔고는 세자리마다 쉼표를 출력하세요.

은행이름: **SC**은행
예금주: 파이썬
계좌번호: **111-11-111111**
잔고: **10,000**원

277. 이자 지급하기

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

278. 여러 객체 생성

▶ Account 클래스로부터 3개 이상 인스턴스를 생성하고 생성된 인스턴스를 리스트에 저장해 보세요.

279. 객체 순회

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

280. 입출금 내역

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

281. 클래스 정의

▶ 다음 코드가 동작하도록 차 클래스를 정의하세요.

```
>> car = 차(2, 1000)
>> car.바퀴
2
>> car.가격
1000
```

▷ 정답확인

282. 클래스 상속

▶ 차 클래스를 상속받은 자전차 클래스를 정의하세요.

283. 클래스 상속

▶ 다음 코드가 동작하도록 자전차 클래스를 정의하세요. 단 자전차 클래스는 차 클래스를 상속받습니다.

```
>> bicycle = 자전차(2, 100)
>> bicycle.가격
100
```

284. 클래스 상속

▶ 다음 코드가 동작하도록 자전차 클래스를 정의하세요. 단 자전차 클래스는 차 클래스를 상속받습니다.

```
>> bicycle = 자전차(2, 100, "시마노")
>> bicycle.구동계
시마노
```

285. 클래스 상속

▶ 다음 코드가 동작하도록 차 클래스를 상속받는 자동차 클래스를 정의하세요.

```
>> car = 자동차(4, 1000)
>> car.정보()
바퀴수 4
가격 1000
```

286. 부모 클래스 생성자 호출

▶ 다음 코드가 동작하도록 차 클래스를 수정하세요.

```
>> bicycle = 자전차(2, 100, "시마노")
>> bicycle.정보()

바퀴수 2
가격 100
```

287. 부모 클래스 메서드 호출

▶ 자전차의 정보() 메서드로 구동계 정보까지 출력하도록 수정해보세요.

```
>> bicycle = 자전차(2, 100, "시마노")
>> bicycle.정보()
바퀴수 2
가격 100
구동계 시마노
```

288. 메서드 오버라이딩

▶ 다음 코드의 실행 결과를 예상해보세요.

```
class 부모:
  def 호출(self):
    print("부모호출")

class 자식(부모):
  def 호출(self):
    print("자식호출")
나 = 자식()
나.호출()
```

289. 생성자

▶ 다음 코드의 실행 결과를 예상해보세요.

```
class 부모:
  def __init__(self):
    print("부모생성")

class 자식(부모):
  def __init__(self):
    print("자식생성")
나 = 자식()
```

290. 부모클래스 생성자 호출

▶ 다음 코드의 실행 결과를 예상해보세요.

```
class 부모:
  def __init__(self):
    print("부모생성")

class 자식(부모):
  def __init__(self):
    print("자식생성")
    super().__init__()

나 = 자식()
```

12. 파일 입출력과 예외처리

291. 파일 쓰기

▶ 바탕화면에 '매수종목1.txt' 파일을 생성한 후 다음과 같이 종목코드를 파일에 써보세요.

005930
005380
035420

292. 파일 쓰기

▶ 바탕화면에 '매수종목2.txt' 파일을 생성한 후 다음과 같이 종목코드와 종목명을 파일에 써보세요.

005930 삼성전자
005380 현대차
035420 NAVER

293. CSV 파일 쓰기

▶ 바탕화면에 '매수종목.csv' 파일을 생성한 후 다음과 같이 종목코드와 종목명을 파일에 써보세요. 인코딩은 'cp949'를 사용해야합니다.

	A	B	C
1	종목명	종목코드	PER
2	삼성전자	005930	15.79
3	NAVER	035420	55.82

294. 파일 읽기

▶ 바탕화면에 생성한 '매수종목1.txt' 파일을 읽은 후 종목코드를 리스트에 저장해보세요.

005930

005380
035420

295. 파일 읽기

▶ 바탕화면에 생성한 '매수종목2.txt' 파일을 읽은 후 종목코드와 종목명을 딕셔너리로 저장해보세요. 종목명을 key로 종목명을 value로 저장합니다.

005930 삼성전자
005380 현대차
035420 NAVER

296. 예외처리

▶ 문자열 PER (Price to Earning Ratio) 값을 실수로 변환할 때 에러가 발생합니다. 예외처리를 통해 에러가 발생하는 PER은 0으로 출력하세요.

```
per = ["10.31", "", "8.00"]

for i in per:
    print(float(i))
```

297. 예외처리 및 리스트에 저장

▶ 문자열로 표현된 PER 값을 실수로 변환한 후 이를 새로운 리스트에 저장해보세요.

```
per = ["10.31", "", "8.00"]

for i in per:
    print(float(per))
```

298. 특정 예외만 처리하기

▶ 어떤 값을 0으로 나누면 ZeroDivisionError 에러가 발생합니다. try ~ except로 모든

에러에 대해 예외처리하지 말고 **ZeroDivisionError** 에러만 예외처리해보세요.

299. 예외의 메시지 출력하기

▶ 다음과 같은 코드 구조를 사용하면 예외 발생 시 에러 메시지를 변수로 할당할 수 있습니다.

```
try:
    실행코드
except 예외 as 변수:
    예외처리코드
```

리스트의 인덱싱에 대해 에러를 출력해보세요.

```
data = [1, 2, 3]

for i in range(5)
    print(data[i])
```

300. try, except, else, finally 구조 사용해보기

▶ 파이썬 예외처리는 다음과 같은 구조를 가질 수 있습니다.

```
try:
    실행 코드
except:
    예외가 발생했을 때 수행할 코드
else:
    예외가 발생하지 않았을 때 수행할 코드
finally:
    예외 발생 여부와 상관없이 항상 수행할 코드
```

아래의 코드에 대해서 예외처리를 사용하고 **try, except, else, finally**에 적당한 코드를 작성해봅시다. **else**와 **finally**는 적당한 문구를 **print**하시면 됩니다.

```
per = ["10.31", "", "8.00"]
```

```
for i in per:
    print(float(per))
```