**20210203\_인터페이스개발\_수행평가\_김현석**

1. **파이썬 연습문제 1~10번**

**1. 문제:**

**10보다 작은 자연수 중에서 3 또는 5의 배수는 3,5,6,9가 존재해요! 이것들의 합은 23입니다. 1000보다 작은 자연수 중에서 3 또는 5의 배수들을 구해서 모두 합하면 얼마인가요?**

**코드:**

result = 0

for i in range(1000):

if i % 3 == 0:

result += i

elif i % 5 == 0:

result += i

else:

pass

print(result)

**답:**



**2. 문제:**

**피보나치 수열의 각 항은 바로 앞의 항 두 개를 더한 것이 됩니다. 1과 2로 시작하는 경우 이 수열은 아래와 같습니다. 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ... 짝수이면서 4백만 이하인 모든 항을 더하면 얼마가 됩니까?**

**코드:**

i = 1

previous = 1

current = 1

pivo = []

result = 0

while i <= 50:

pivo.append(current)

previous, current = current, previous + current

i += 1

for i in pivo:

if i % 2 == 0 and i <= 4000000:

result += i

print(result)

**답:**



**3. 문제:**

**알파벳 대소문자로 된 문자열이 주어지면, 이 문자열에서 가장 많이 사용된 알파벳이 무엇인지 출력하는 프로그램을 작성하시오. 단, 대소문자는 구별하지 않아요. 만약 동률이 존재하는 경우 알파벳 순으로 제일 앞에 있는 문자를 출력하세요.**

**코드:**

string1 = "This is a sample Program mississippi river"

string2 = "abcdabcdababccddcd"

up\_string1 = string1.upper()

up\_string2 = string2.upper()

result1 = 0

word1 = ''

for i in up\_string1:

if result1 < up\_string1.count(i):

result1 = up\_string1.count(i)

word1 = i

print(f'정답 : "{string1}" => {word1}')

result2 = 0

word2 = ''

for i in up\_string2:

if result2 < up\_string2.count(i):

result2 = up\_string2.count(i)

word2 = i

print(f'정답 : "{string2}" => {word2}')

**답:**



**4. 문제:**

**## 로또 프로그램 작성**

**## 5000원으로 로또복권을 5장 자동으로 구매합니다.**

**## 이번 주 로또 당첨번호를 생성하여 로또 당첨을 확인하세요!**

**## 쉬운버전으로 먼저 작성합니다.**

**## 6숫자가 다 맞으면 1등, 5개 맞으면 2등으로 처리합니다.**

**## 즉, 쉬운버전은 보너스 숫자는 없는 것으로 간주합니다.**

**## 쉬운버전을 해결했다면**

**## 보너스 숫자를 이용하여 로또 당첨을 확인합니다.**

**## 보너스 숫자를 제외한 모든 숫자가 다 맞으면 1등,**

**## 보너스 숫자를 포함하여 6개의 숫자가 맞으면 2등,**

**## 보너스를 제외하고 5개의 숫자가 맞으면 3등으로 처리합니다.**

**## 쉬운버전의 출력은 1등 몇개, 2등 몇개, 3등 몇개,**

**## 4등 몇개, 꽝 몇개로 출력**

**## 어려운버전의 출력은 1등 몇개, 2등 몇개, 3등 몇개,**

**## 4등 몇개, 5등 몇개, 꽝 몇개로 출력**

**코드:**

# 쉬운 버전 로또

import random

my\_lotto = []

lotto = []

for i in range(6): # lotto 당첨 번호 여섯 개 뽑기

a = random.randint(1,45)

while a in lotto: # 중복 없이 수 뽑기

a = random.randint(1,45)

lotto.append(a) # 뽑은 랜덤 숫자 lotto 리스트에 삽입

for i in range(5): # 나의 로또복권 5장 뽑기

my\_number = []

for n in range(6): # 나의 로또 번호 여섯 개 뽑기

a = random.randint(1,45)

while a in my\_number: # 중복 없이 수 뽑기

a = random.randint(1,45)

my\_number.append(a) # 뽑은 랜덤 숫자 나의 로또 번호 리스트에 삽입

my\_lotto.append(my\_number) # 나의 로또 번호 리스트를 나의 로또 복권 리스트에 삽입

one, two, three, four, fail = 0,0,0,0,0

for i in my\_lotto:

result =[]

result = set(lotto) & set(i)

if len(result) == 6:

one += 1

elif len(result) == 5:

two += 1

elif len(result) == 4:

three += 1

elif len(result) == 3:

four += 1

else:

fail += 1

print(f"1등 : {one}개, 2등 : {two}개, 3등 : {three}개, 4등 : {four}개, 꽝 : {fail}개")

**답:**



**코드:**

# 보너스 번호 추가 버전

import random

my\_lotto = []

lotto = []

for i in range(6): # lotto 당첨 번호 다섯 개 뽑기

a = random.randint(1,45)

while a in lotto: # 중복 없이 수 뽑기

a = random.randint(1,45)

lotto.append(a) # 뽑은 랜덤 숫자 lotto 리스트에 삽입

for i in range(5): # 나의 로또복권 5장 뽑기

my\_number = []

for n in range(6): # 나의 로또 번호 여섯 개 뽑기

a = random.randint(1,45)

while a in my\_number: # 중복 없이 수 뽑기

a = random.randint(1,45)

my\_number.append(a) # 뽑은 랜덤 숫자 나의 로또 번호 리스트에 삽입

my\_bonus = random.randint(1,45) # 보너스 번호 추가

while my\_bonus in my\_number:

my\_bonus = random.randint(1,45)

my\_number.append(my\_bonus) # 보너스 번호를 나의 로또 번호 리스트에 삽입

my\_lotto.append(my\_number) # 나의 로또 번호 리스트를 나의 로또 복권 리스트에 삽입

one, two, thee, four, five, fail = 0,0,0,0,0,0

for i in my\_lotto:

result = []

result = set(lotto) & set(i[:6])

bonus = 0

if i[-1] in lotto:

bonus = 1

if len(result) == 6:

one += 1

elif len(result) + bonus == 6:

two += 1

elif len(result) == 5:

three += 1

elif len(result) == 4:

four += 1

elif len(result) == 3:

five += 1

else:

fail += 1

print(f"1등 : {one}개, 2등 : {two}개, 3등 : {three}개, 4등 : {four}개, 5등 : {five}개, 꽝 : {fail}개")

**답:**



**5. 문제:**

**## 어떤 수를 소수의 곱으로만 나타내는 것을 소인수분해라 하고,**

**## 이 소수들을 그 수의 소인수라고 합니다.**

**## 예를 들면 13195의 소인수는 5, 7, 13, 29 입니다.**

**## 600851475143의 소인수 중에서 가장 큰 수를 구하세요.**

**코드:**

n\_list = []

for i in range(1, 10000):

n = 0

for j in range(1, i + 1):

if i % j == 0:

n += 1

if n == 2:

n\_list.append(i)

result = []

for i in n\_list:

if 600851475143 % i == 0:

result.append(i)

print(max(result))

**답:**



**6. 문제:**

**## 앞에서부터 읽을 때나 뒤에서부터 읽을 때나 모양이**

**## 같은 수를 대칭수(palindrome)라고 부릅니다.**

**## 두 자리 수를 곱해 만들 수 있는 대칭수 중**

**## 가장 큰 수는 9009 (= 91 × 99) 입니다.**

**## 세 자리 수를 곱해 만들 수 있는 가장 큰 대칭수를 구하세요**

**코드:**

def is\_palindrome(number):

return number == number[::-1]

maxim = 0

for i in range(999, 1, -1):

for j in range(999, 1, -1):

n = j\*i

if is\_palindrome(str(n)) == True:

maxim = max(maxim, n)

print(maxim)

**답:**



**7. 문제:**

**## 1 ~ 10 사이의 어떤 수로도 나누어 떨어지는 가장 작은 수는 2520입니다.**

**## 그러면 1 ~ 20 사이의 어떤 수로도 나누어 떨어지는 가장 작은 수는 얼마입니까?**

**코드:**

n\_list = [i for i in range(1,21)]

n = 232792500

count = 0

while count < len(n\_list):

n += 1

count = 0

for i in n\_list:

if n % i == 0:

count += 1

print(n)

**답:**



**8. 문제:**

**## 1부터 10까지 자연수를 각각 제곱해 더하면 다음과 같습니다 (제곱의 합).**

**## 1\*\*2 + 2\*\*2 + ... + 10\*\*2 = 385**

**## 1부터 10을 먼저 더한 다음에 그 결과를 제곱하면 다음과 같습니다 (합의 제곱).**

**## (1 + 2 + ... + 10)\*\*2 = 552 = 3025**

**## 따라서 1부터 10까지 자연수에 대해 "합의 제곱"과 "제곱의 합" 의**

**## 차이는 3025 - 385 = 2640 이 됩니다.**

**## 그러면 1부터 100까지 자연수에 대해 "합의 제곱"과 "제곱의 합"의 차이는**

**## 얼마입니까?**

**코드:**

def square\_sum(x):

square\_sum = 0

square\_list = [i\*\*2 for i in range(1, x+1)]

for i in square\_list:

square\_sum += i

return square\_sum

def sum\_square(x):

sum\_square = 0

sum\_list = [i for i in range(1,x+1)]

for i in sum\_list:

sum\_square += i

sum\_square = sum\_square \*\* 2

return sum\_square

print(sum\_square(100) - square\_sum(100))

**답:**



**9. 문제:**

**## 소수를 크기 순으로 나열하면 2, 3, 5, 7, 11, 13, ... 과 같이 됩니다.**

**## 이 때 10,001번째의 소수를 구하세요.**

**코드:**

primes = [2]

number = 1

def IsPrimeNumber(number):

for i in range(len(primes)):

if number % primes[i] == 0:

return False

return True

while len(primes) < 10001:

number += 2

if IsPrimeNumber(number):

primes.append(number)

print(primes[10000])

**답:**



**10. 문제:**

**## 세 자연수 a, b, c 가 피타고라스 정리 a\*\*2 + b\*\*2 = c\*\*2 를 만족하면**

**## 피타고라스 수라고 부릅니다 (여기서 a < b < c ).**

**## 예를 들면 3\*\*2 + 4\*\*2 = 9 + 16 = 25 = 5\*\*2이므로**

**## 3, 4, 5는 피타고라스 수입니다.**

**## a + b + c = 1000 인 피타고라스 수 a, b, c는 한 가지 뿐입니다.**

**## 이 때, a × b × c 는 얼마입니까?**

**코드:**

for a in range(1, 1000):

for b in range(1, 1000):

c = 1000 - a - b

if a \* a + b \* b == c \* c and a < b < c:

print(a \* b \* c)

**답:**



1. **블로그에 있는 클래스 연습문제 구현**

**Data load:**

class Mpg:

def \_\_init\_\_(self, car\_data):

self.manufacturer = car\_data[0] # 제조업체 이름

self.model = car\_data[1] # 모델명

self.displ = car\_data[2] # 엔진 배기량(리터)

self.year = car\_data[3] # 제조년도

self.cyl = car\_data[4] # 실린더 수

self.trans = car\_data[5] # 전송 유형

self.drv = car\_data[6] # 드라이브 트레인의 유형

# f: 전륜 구동, r: 후륜 구동, 4: 4wd

self.cty = car\_data[7] # 도시 연비

self.hwy = car\_data[8] # 고속도로 연비

self.fl = car\_data[9] # 연료 종류

self.car\_class = car\_data[10] # 자동차의 유형

def \_\_lt\_\_(self, other): # little then

return self.hwy < other.hwy

def \_\_str\_\_(self):

return f"{self.manufacturer}, {self.model}, {self.displ}, "\

f"{self.year}, {self.cyl}, {self.trans}, {self.drv}, "\

f"{self.cty}, {self.hwy}, {self.fl}, {self.car\_class}"

my\_file = open('mpg.txt', 'r')

my\_file.readline()

car\_list = []

while True:

line = my\_file.readline()

if len(line) < 1:

pass

else:

line2 = line.strip('\n')

split\_list = line2.split(',')

car\_list.append(Mpg(split\_list))

if not line:

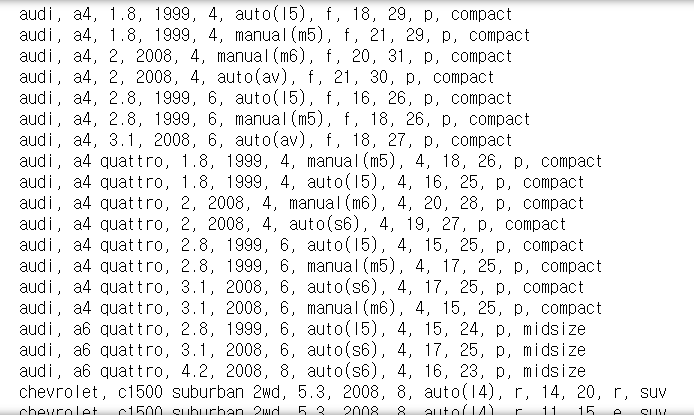
break

my\_file.close()

for i in car\_list:

print(i)

**결과:**



**1. 문제:**

**# 1. displ(배기량)이 4 이하인 자동차와 5 이상인 자동차 중**

**# 어떤 자동차의 hwy(고속도로 연비)가 평균적으로 더 높은지 확인하세요.**

**코드:**

low\_count, sum\_low\_displ, high\_count, sum\_high\_displ = 0,0,0,0

for i in range(len(car\_list)):

if float(car\_list[i].displ) <= 4:

sum\_low\_displ += int(car\_list[i].hwy)

low\_count += 1

elif float(car\_list[i].displ) >= 5:

sum\_high\_displ += int(car\_list[i].hwy)

high\_count += 1

low\_mean, high\_mean = sum\_low\_displ / low\_count, sum\_high\_displ / high\_count

if low\_mean > high\_mean:

print('배기량이 4 이하인 자동차의 고속도로 연비가 더 높습니다')

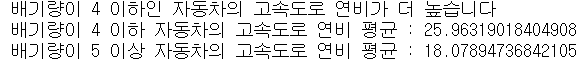
else:

print('배기량이 5 이상인 자동차의 고속도로 연비가 더 높습니다')

print(f"배기량이 4 이하 자동차의 고속도로 연비 평균 : {low\_mean}")

print(f"배기량이 5 이상 자동차의 고속도로 연비 평균 : {high\_mean}")ㄴ

**답:**



**2. 문제:**

**# 2. 자동차 제조 회사에 따라 도시 연비가 다른지 알아보려고 한다.**

**# "audi"와 "toyota" 중 어느 manufacturer(제조회사)의 cty(도시 연비)가**

**# 평균적으로 더 높은지 확인하세요.**

**코드:**

audi\_count, sum\_audi\_cty, toyota\_count, sum\_toyota\_cty =0, 0, 0, 0

for i in range(len(car\_list)):

if car\_list[i].manufacturer == "audi":

sum\_audi\_cty += int(car\_list[i].cty)

audi\_count += 1

elif car\_list[i].manufacturer == "toyota":

sum\_toyota\_cty += int(car\_list[i].cty)

toyota\_count += 1

audi\_mean, toyota\_mean = sum\_audi\_cty/audi\_count, sum\_toyota\_cty/toyota\_count

if audi\_mean > toyota\_mean:

print("audi의 도시 연비가 더 높습니다")

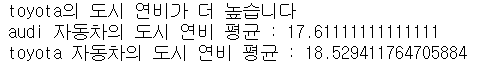
else:

print("toyota의 도시 연비가 더 높습니다")

print(f"audi 자동차의 도시 연비 평균 : {audi\_mean}")

print(f"toyota 자동차의 도시 연비 평균 : {toyota\_mean}")

**답:**



**3. 문제:**

**# 3. "chevrolet", "ford", "honda" 자동차의 고속도로 연비 평균을 알아보려고 한다.**

**# 이 회사들의 데이터를 추출한 후 hwy(고속도로 연비) 평균을 구하세요.**

**코드:**

count, sum\_hwy = 0, 0

manufacturer\_list = ["chevrolet", "ford", "honda"]

for i in range(len(car\_list)):

if car\_list[i].manufacturer in manufacturer\_list:

sum\_hwy += int(car\_list[i].hwy)

count += 1

mean = sum\_hwy / count

print(f'"chevrolet", "ford", "honda"의 고속도로 연비 평균은 {mean}입니다')

**답:**



**4. 문제:**

**# 4. "audi"에서 생산한 자동차 중에 어떤 자동차 모델의 hwy(고속도로 연비)가**

**# 높은지 알아보려고 한다. "audi"에서 생산한 자동차 중 hwy가 1~5위에 해당하는**

**# 자동차의 데이터를 출력하세요.**

**코드:**

audi\_list = []

for i in range(len(car\_list)):

if car\_list[i].manufacturer == "audi":

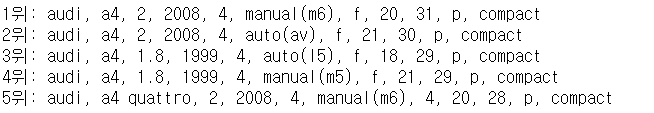
audi\_list.append(car\_list[i])

audi\_hwy\_list = sorted(audi\_list, reverse = True)

for i in range(5):

print(f"{i+1}위: {audi\_hwy\_list[i]}")

**답:**



**5. 문제:**

**# 5. mpg 데이터는 연비를 나타내는 변수가 2개입니다.**

**# 두 변수를 각각 활용하는 대신 하나의 통합 연비 변수를 만들어 사용하려 합니다.**

**# 평균 연비 변수는 두 연비(고속도로와 도시)의 평균을 이용합니다.**

**# 회사별로 "suv" 자동차의 평균 연비를 구한후 내림차순으로 정렬한 후 1~5위까지 데이터를 출력하세요.**

**코드:**

# suv 자동차의 리스트 생성

suv\_list = []

for i in range(len(car\_list)):

if car\_list[i].car\_class == "suv":

suv\_list.append(car\_list[i])

# suv 자동차의 리스트에서 제조회사명을 중복없이 추출

company\_list = []

for i in range(len(suv\_list)):

if suv\_list[i].manufacturer in company\_list:

pass

else:

company\_list.append(suv\_list[i].manufacturer)

# 회사별로 suv 자동차의 평균 연비를 추출

sum\_list = []

for i in range(len(company\_list)):

temp\_sum = 0

count = 0

for j in range(len(suv\_list)):

if suv\_list[j].manufacturer == company\_list[i]:

temp\_sum += (int(suv\_list[j].cty) + int(suv\_list[j].hwy)) / 2

count += 1

sum\_list.append(temp\_sum / count)

# 회사명과 suv 자동차의 평균 연비를 연결하는 dict 생성

mean\_dict = {}

for i in range(len(sum\_list)):

mean\_dict[company\_list[i]] = sum\_list[i]

# 평균 연비 순위를 구해 mean\_desc 리스트에 삽입

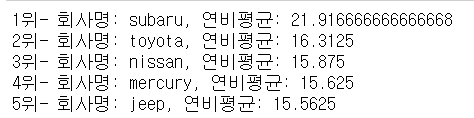
mean\_desc = sorted(mean\_dict.items(), key = (lambda x:x[1]), reverse = True)

# 1위부터 5위까지의 회사명과 연비 평균 데이터 출력

for i in range(5):

print(f"{i+1}위- 회사명: {mean\_desc[i][0]}, 연비평균: {mean\_desc[i][1]}")

**답:**



**6. 문제:**

**# 6. mpg 데이터의 class는 "suv", "compact" 등 자동차의 특징에 따라**

**# 일곱 종류로 분류한 변수입니다. 어떤 차종의 도시 연비가 높은지 비교하려 합니다.**

**# class별 cty 평균을 구하고 cty 평균이 높은 순으로 정렬해 출력하세요.**

**코드:**

# 자동차의 유형을 중복없이 추출

class\_list = []

for i in range(len(car\_list)):

if car\_list[i].car\_class in class\_list:

pass

else:

class\_list.append(car\_list[i].car\_class)

# 자동차 유형별 도시연비 평균을 추출

sum\_list = []

for i in range(len(class\_list)):

temp\_sum = 0

count = 0

for j in range(len(car\_list)):

if car\_list[j].car\_class == class\_list[i]:

temp\_sum += int(car\_list[j].cty)

count += 1

sum\_list.append(temp\_sum / count)

# 자동차 유형과 도시연비 평균을 연결하는 dict 생성

mean\_dict = {}

for i in range(len(sum\_list)):

mean\_dict[class\_list[i]] = sum\_list[i]

# 도시연비 평균의 순위를 구해 mean\_desc 리스트에 삽입

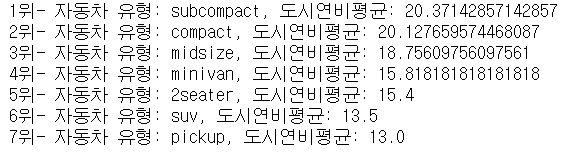
mean\_desc = sorted(mean\_dict.items(), key = (lambda x:x[1]), reverse = True)

# 도시연비 평균이 높은 순으로 자동차 유형과 도시연비 평균 데이터 출력

for i in range(len(class\_list)):

print(f"{i+1}위- 자동차 유형: {mean\_desc[i][0]}, 도시연비평균: {mean\_desc[i][1]}")

**답:**



**7. 문제:**

**# 7. 어떤 회사 자동차의 hwy(고속도로 연비)가 가장 높은지 알아보려 합니다.**

**# hwy(고속도로 연비) 평균이 가장 높은 회사 세 곳을 출력하세요.**

**코드:**

# 제조 회사명을 중복없이 추출

company\_list = []

for i in range(len(car\_list)):

if car\_list[i].manufacturer in company\_list:

pass

else:

company\_list.append(car\_list[i].manufacturer)

# 회사별 고속도로연비 평균을 추출

sum\_list = []

for i in range(len(company\_list)):

temp\_sum = 0

count = 0

for j in range(len(car\_list)):

if car\_list[j].manufacturer == company\_list[i]:

temp\_sum += int(car\_list[j].hwy)

count += 1

sum\_list.append(temp\_sum / count)

# 회사와 고속도로연비 평균을 연결하는 dict 생성

mean\_dict = {}

for i in range(len(sum\_list)):

mean\_dict[company\_list[i]] = sum\_list[i]

# 고속도로연비 평균의 순위를 구해 mean\_desc 리스트에 삽입

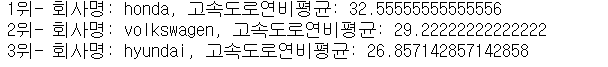
mean\_desc = sorted(mean\_dict.items(), key = (lambda x:x[1]), reverse = True)

# 고속도로연비 평균이 높은 순으로 회사명과 고속도로연비 평균 데이터 출력

for i in range(3):

print(f"{i+1}위- 회사명: {mean\_desc[i][0]}, 고속도로연비평균: {mean\_desc[i][1]}")

**답:**



**8. 문제:**

**# 8. 어떤 회사에서 "compact" 차종을 가장 많이 생산하는지 알아보려고 합니다.**

**# 각 회사별 "compact" 차종 수를 내림차순으로 정렬해 출력하세요.**

**코드:**

# compact 자동차의 리스트 생성

compact\_list = []

for i in range(len(car\_list)):

if car\_list[i].car\_class == "compact":

compact\_list.append(car\_list[i])

# compact 자동차의 리스트에서 제조회사명을 중복없이 추출

company\_list = []

for i in range(len(compact\_list)):

if compact\_list[i].manufacturer in company\_list:

pass

else:

company\_list.append(compact\_list[i].manufacturer)

# 회사별로 compact 자동차의 개수를 추출

sum\_list = []

for i in range(len(company\_list)):

count = 0

for j in range(len(compact\_list)):

if compact\_list[j].manufacturer == company\_list[i]:

count += 1

sum\_list.append(count)

# 회사명과 compact 자동차의 개수를 연결하는 dict 생성

cnt\_dict = {}

for i in range(len(sum\_list)):

cnt\_dict[company\_list[i]] = sum\_list[i]

# 차종 수를 구해 mean\_desc 리스트에 삽입

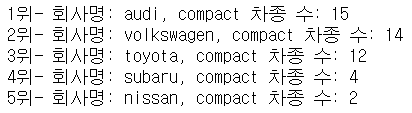
cnt\_desc = sorted(cnt\_dict.items(), key = (lambda x:x[1]), reverse = True)

# compact 차종 개수가 높은 순으로 회사명과 compact 차종 개수 데이터 출력

for i in range(len(company\_list)):

print(f"{i+1}위- 회사명: {cnt\_desc[i][0]}, compact 차종 수: {cnt\_desc[i][1]}")

**답:**



1. **블로그에 있는 Pandas 연습문제 구현**
2. **Pandas Exercise – 1**

**Data load:**

# movie 수행평가

# movieId로 merge

# timestamp => 날짜를 숫자로 표현하는 표준

# 1970년 1월1일0시0분0초 => 0

# 1970년 1월1일0시0분1초 => 1

# 1970년 1월1일0시0분2초 => 2

import numpy as np

import pandas as pd

# D:\python\_ML\ml-latest-small

movie\_df = pd.read\_csv('D:\python\_ML\ml-latest-small/movies.csv')

rating\_df = pd.read\_csv('D:\python\_ML\ml-latest-small/ratings.csv')

display(movie\_df)

display(rating\_df)

**결과:**



**1. 사용자가 평가한 모든 영화의 전체 평균 평점을 출력하세요.**

**코드:**

# 사용자가 평가한 모든 영화이니 movieId를 기준으로 right merge 후 rating 평균 출력

print( pd.merge(movie\_df, rating\_df, on='movieId', how='right')['rating'].mean() )

**답:**



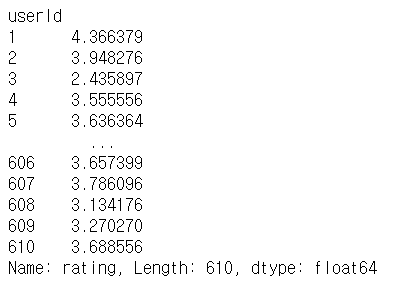
**2. 각 사용자별 평균 평점을 구하세요. 출력시 정렬은 userId로 오름차순 정렬합니다.**

**코드:**

# userId별로 그룹화한 후 평균 rating 출력

print( rating\_df.groupby(rating\_df['userId'])['rating'].mean() )

**답:**



**3. 각 영화별 평균 평점을 구하세요. 출력시 정렬은 movieId로 오름차순 정렬합니다.**

**코드:**

# rating\_df의 rating을 movieId 그룹화

result = rating\_df['rating'].groupby(rating\_df['movieId']).mean()

# result를 movie\_df와 movieId 기준으로 merge

# 누락되는 영화가 없도록 left merge 이용

df = pd.merge(movie\_df, result,

left\_on = 'movieId',

right\_index = True,

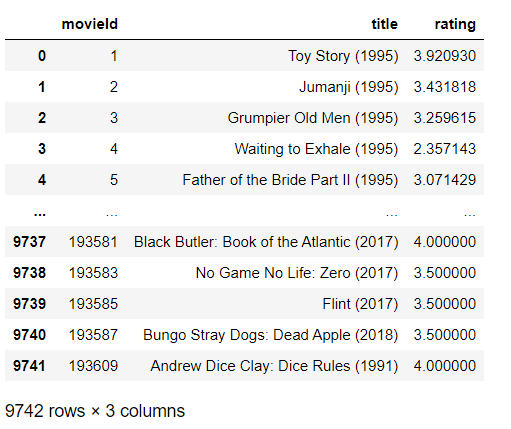
how = 'left')

# df의 movieId, title, rating을 요소로 갖는 DataFrame(result) 생성

result = df[['movieId','title','rating']]

display(result)

**답:**



**4. 평균 평점이 가장 높은 영화의 제목을 출력하세요.**

**단, 동률이 있을 경우 모두 출력하고 title을 기준으로 오름차순 정렬하세요.**

**코드:**

# rating\_df의 rating을 movieId 그룹화

result = rating\_df['rating'].groupby(rating\_df['movieId']).mean()

# result를 movie\_df와 movieId 기준으로 merge

# 누락되는 영화가 없도록 left merge 이용

df = pd.merge(movie\_df, result,

left\_on = 'movieId',

right\_index = True,

how = 'left')

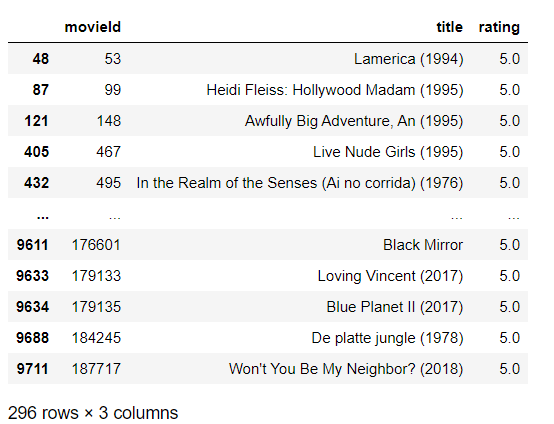
# df의 movieId, title, rating을 요소로 갖는 DataFrame(result) 생성

result = df[['movieId','title','rating']]

# loc[]를 이용하여 rating 값이 rating.max()값과 동일한 record만 출력

display( result.loc[result['rating'] == result['rating'].max(),:] )

**답:**



**5. Comedy영화 중 가장 평점이 낮은 영화의 제목을 출력하세요.**

**단, 동률이 있을 경우 모두 출력하고 title을 기준으로 오름차순 정렬하세요.**

**코드:**

# rating\_df의 rating을 movieId 그룹화

result = rating\_df['rating'].groupby(rating\_df['movieId']).mean()

# result를 movie\_df와 movieId 기준으로 merge

# 평점이 없는 영화를 제외하기 위해 inner merge 사용

df = pd.merge(movie\_df, result,

left\_on = 'movieId',

right\_index = True,

how = 'inner')

# df의 movieId, title, genres, rating을 요소로 갖는 DataFrame(result) 생성

result = df[['movieId', 'title', 'genres','rating']]

# .str.contains()를 사용하여 genres 안에 Comedy가 포함되는 record만 추출한

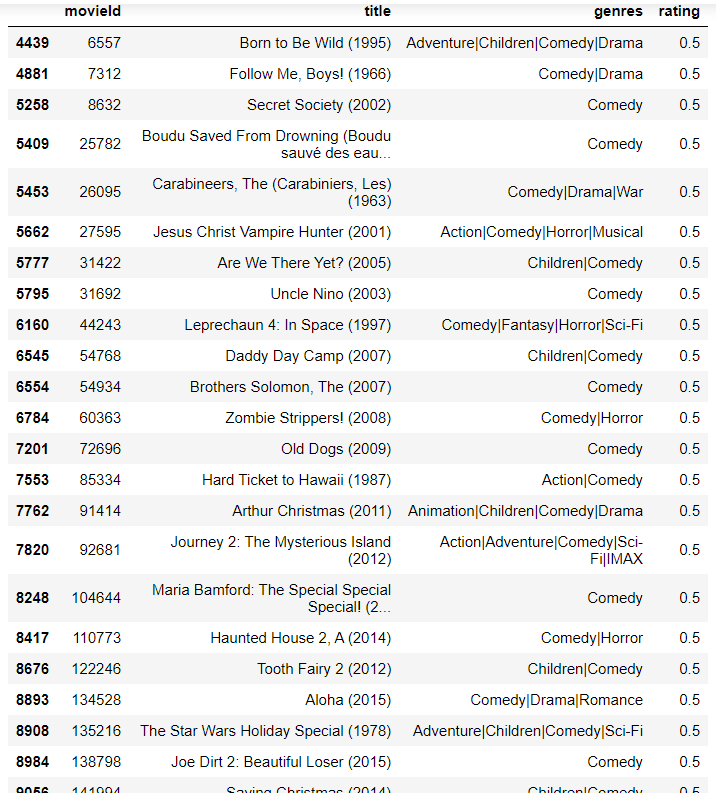
# DataFrame(comedy\_df) 생성

comedy\_df = result[result.genres.str.contains('Comedy')]

# loc[]를 이용하여 rating 값이 rating.min()값과 동일한 record만 출력

display( comedy\_df.loc[comedy\_df['rating'] == comedy\_df['rating'].min(),:] )

**답:**



**6. 2015년도에 평가된 모든 Romance 영화의 평균 평점은?**

**코드:**

# movie\_df와 rating\_df를 movieId를 기준으로 merge

# user에 의해 평가된 영화만 구하기 위해 right merge 사용

df = pd.merge(movie\_df, rating\_df, on='movieId', how='right')

# to\_datetime()을 이용하여 timestamp를 datetime으로 변환

df['date'] = pd.to\_datetime(df['timestamp'], unit='s')

# .strftime()을 이용하여 datetime에서 년도만 추출

df['year'] = df['date'].dt.strftime('%Y')

# df의 movieId, genres, rating, year를 요소로 갖는 DataFrame(result) 생성

result = df[['movieId', 'genres','rating', 'year']]

# .str.contains()를 사용하여 genres 안에 Romance가 포함되는 record만 추출한

# DataFrame(romance\_df) 생성

romance\_df = result[result.genres.str.contains('Romance')]

# loc[]를 이용하여 year가 2015년도인 record들의 rating 평균을 출력

display( romance\_df.loc[romance\_df['year'] == '2015',:]['rating'].mean() )

**답:**



1. **Pandas Exercise – 2**

**Data load:**

import numpy as np

import pandas as pd

df = pd.read\_csv('D:\python\_ML/mpg.txt')

display(df)

# 주요컬럼 설명

# manufacturer : 제조회사

# displ : 배기량

# cyl : 실린더 개수

# drv : 구동 방식

# hwy : 고속도로 연비

# class : 자동차 종류

# model : 자동차 모델명

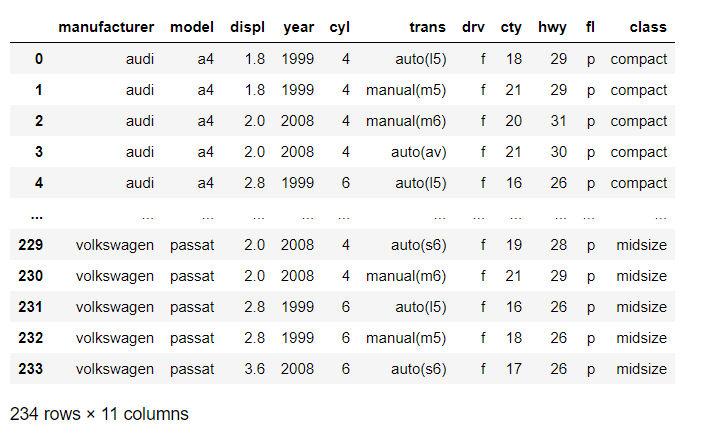
# year : 생산연도

# trans : 변속기 종류

# cty : 도시 연비

# fl : 연료 종류

**결과:**



**1. displ(배기량)이 4 이하인 자동차와 5 이상인 자동차 중**

**어떤 자동차의 hwy(고속도로 연비)가 평균적으로 더 높은지 확인하세요.**

**코드:**

print( f"배기량 4이하인 자동차 : {df.loc[df['displ'] <= 4,:]['hwy'].mean()}" )

print( f"배기량 5이상인 자동차 : {df.loc[df['displ'] >= 5,:]['hwy'].mean()}" )

**답:**



**2. 자동차 제조 회사에 따라 도시 연비가 다른지 알아보려고 한다.**

**"audi"와 "toyota" 중 어느 manufacturer(제조회사)의 cty(도시 연비)가**

**평균적으로 더 높은지 확인하세요.**

**코드:**

print( f"audi의 도시연비평균 : {df.loc[df['manufacturer'] == 'audi',:]['cty'].mean()}" )

print( f"audi의 도시연비평균 : {df.loc[df['manufacturer'] == 'toyota',:]['cty'].mean()}" )

**답:**



**3. "chevrolet", "ford", "honda" 자동차의 고속도로 연비 평균을 알아보려고 한다.**

**이 회사들의 데이터를 추출한 후 hwy(고속도로 연비) 평균을 구하세요.**

**코드:**

print(f"hwy 평균 : {df.loc[df['manufacturer'].isin(['chevrolet', 'ford', 'honda']),:]['hwy'].mean()}")

**답:**



**4. "audi"에서 생산한 자동차 중에 어떤 자동차 모델의 hwy(고속도로 연비)가**

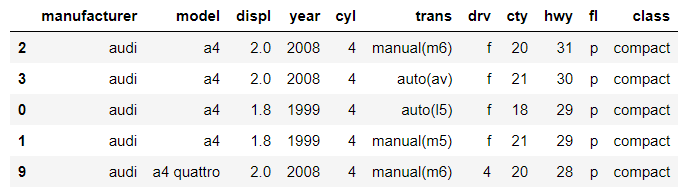
**높은지 알아보려고 한다.**

**"audi"에서 생산한 자동차 중 hwy가 1~5위에 해당하는 자동차의 데이터를 출력하세요.**

**코드:**

display( df.loc[df['manufacturer'] == 'audi',:].sort\_values(by='hwy', ascending=False).head() )

**답:**



**5. mpg 데이터는 연비를 나타내는 변수가 2개입니다.**

**두 변수를 각각 활용하는 대신 하나의 통합 연비 변수를 만들어 사용하려 합니다.**

**평균 연비 변수는 두 연비(고속도로와 도시)의 평균을 이용합니다.**

**회사별로 "suv" 자동차의 평균 연비를 구한후 내림차순으로 정렬한 후 1~5위까지 데이터를 출력하세요.**

**코드:**

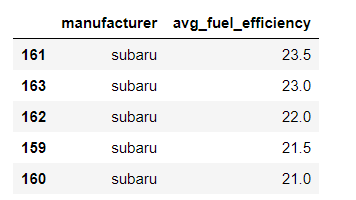
df['avg\_fuel\_efficiency'] = (df['cty'] + df['hwy']) / 2

suv\_df = df.loc[df['class'] == 'suv',:]

result = suv\_df[['manufacturer', 'avg\_fuel\_efficiency']]

display( result.sort\_values(by='avg\_fuel\_efficiency', ascending=False).head() )

**답:**



**6. mpg 데이터의 class는 "suv", "compact" 등 자동차의 특징에 따라**

**일곱 종류로 분류한 변수입니다. 어떤 차종의 도시 연비가 높은지 비교하려 합니다.**

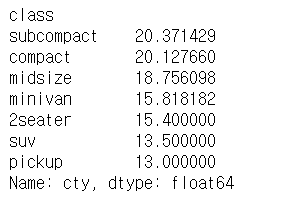
**class별 cty 평균을 구하고 cty 평균이 높은 순으로 정렬해 출력하세요.**

**코드:**

result = df['cty'].groupby(df['class']).mean()

display( result.sort\_values(ascending=False) )

**답:**



**7. 어떤 회사 자동차의 hwy(고속도로 연비)가 가장 높은지 알아보려 합니다.**

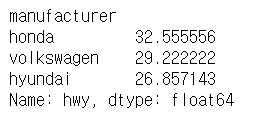
**hwy(고속도로 연비) 평균이 가장 높은 회사 세 곳을 출력하세요.**

**코드:**

result = df['hwy'].groupby(df['manufacturer']).mean()

display( result.sort\_values(ascending=False).head(3) )

**답:**



**8. 어떤 회사에서 "compact" 차종을 가장 많이 생산하는지 알아보려고 합니다.**

**각 회사별 "compact" 차종 수를 내림차순으로 정렬해 출력하세요.**

**코드:**

# compact 차종만 들어있는 compact\_df 생성

compact\_df = df.loc[df['class'] == 'compact',:]

# compact\_df를 manufacturer로 그룹화한 뒤 개수를 출력

result = compact\_df['class'].groupby(compact\_df['manufacturer']).count()

display( result.sort\_values(ascending=False) )

**답:**

