파이썬 문법 익히기

연습문제

4. 파이썬 리스트

51. 리스트 생성

▶ 2016년 11월 영화 예매 순위 기준 top3는 다음과 같습니다. 영화 제목을 movie_rank 이름의 리스트에 저장해보세요. (순위 정보는 저장하지 않습니다.)

순위	영화
1	닥터 스트레인지
2	스플릿
3	럭키

52. 리스트에 원소 추가

▶ 51의 movie_rank 리스트에 "배트맨"을 추가하라.

53.리스트에 원소 추가

▶ movie_rank 리스트에는 아래와 같이 네 개의 영화 제목이 할당되어 있다. "슈퍼맨"을 "닥터 스트레인지"와 "스플릿" 사이에 추가하라.

```
movie_rank = ['닥터 스트레인지', '스플릿', '럭키', '배트맨']
```

54. 리스트에 원소 삭제

▶ movie_rank 리스트에서 '럭키'를 삭제하라.

```
movie_rank = ['닥터 스트레인지', '슈퍼맨', '스플릿', '럭키', '배트맨']
```

55. 리스트에 원소 삭제

▶ movie_rank 리스트에서 '스플릿' 과 '배트맨'을 를 삭제하라.

```
movie_rank = ['닥터 스트레인지', '슈퍼맨', '스플릿', '배트맨']
```

56. 리스트 병합

▶ lang1과 lang2 리스트가 있을 때 lang1과 lang2의 원소를 모두 갖고 있는 langs 리스트를 만들어라.

```
lang1 = ["C", "C++", "JAVA"]
lang2 = ["Python", "Go", "C#"]
```

실행 예:

```
langs
['C', 'C++', 'JAVA', 'Python', 'Go', 'C#']
```

57. 리스트 min(), max() 함수

▶ 다음 리스트에서 최댓값과 최솟값을 출력하라. (힌트: min(), max() 함수 사용)

```
nums = [1, 2, 3, 4, 5, 6, 7]
```

실행 예:

```
max:  7
min:  1
```

58. 리스트의 sum() 함수

▶ 다음 리스트의 합을 출력하라.

```
nums = [1, 2, 3, 4, 5]
```

실행 예:

```
15
```

59. 리스트 len() 함수

▶ 다음 리스트에 저장된 데이터의 개수를 화면에 구하하라.

```
cook = ["피자", "김밥", "만두", "양념치킨", "족발", "피자", "김치만두", "쫄면", "소시지", "라면", "팥빙수", "김치전"]
```

60. 리스트의 평균

▶ 다음 리스트의 평균을 출력하라.

```
nums = [1, 2, 3, 4, 5]
```

실행 예:

```
3.0
```

61. 리스트 슬라이싱

▶ price 변수에는 날짜와 종가 정보가 저장돼 있다. 날짜 정보를 제외하고 가격 정보만을 출력하라. (힌트 : 슬라이싱)

```
price = ['20180728', 100, 130, 140, 150, 160, 170]
```

출력 예시:

```
[100, 130, 140, 150, 160, 170]
```

62. 리스트 슬라이싱

▶ 슬라이싱을 사용해서 홀수만 출력하라.

```
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

실행 예:

```
[1, 3, 5, 7, 9]
```

63. 리스트 슬라이싱

▶ 슬라이싱을 사용해서 짝수만 출력하라.

```
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

실행 예:
```
[2, 4, 6, 8, 10]
```

64. 리스트 슬라이싱

▶ 슬라이싱을 사용해서 리스트의 숫자를 역 방향으로 출력하라.

```
nums = [1, 2, 3, 4, 5]
```

실행 예:
```
[5, 4, 3, 2, 1]
```

65. 리스트 인덱싱

▶ interest 리스트에는 아래의 데이터가 할당되어 있다.

```
interest = ['삼성전자', 'LG전자', 'Naver']
```
interest 리스트를 사용하여 아래와 같이 화면에 출력하라.

출력 예시:
삼성전자 Naver

66. 리스트 join 메서드

▶ interest 리스트에는 아래의 데이터가 할당되어 있다.

```
interest = ['삼성전자', 'LG전자', 'Naver', 'SK하이닉스', '미래에셋대우']
```
interest 리스트를 사용하여 아래와 같이 화면에 출력하라.

출력 예시:
삼성전자 LG전자 Naver SK하이닉스 미래에셋대우

67. 리스트 join 메서드

▶ interest 리스트에는 아래의 데이터가 할당되어 있다.

interest = ['삼성전자', 'LG전자', 'Naver', 'SK하이닉스', '미래에셋대우']
interest 리스트를 사용하여 아래와 같이 화면에 출력하라.

출력 예시:
삼성전자/LG전자/Naver/SK하이닉스/미래에셋대우

68. 리스트 join 메서드

▶ interest 리스트에는 아래의 데이터가 할당되어 있다.

interest = ['삼성전자', 'LG전자', 'Naver', 'SK하이닉스', '미래에셋대우']
join() 메서드를 사용해서 interest 리스트를 아래와 같이 화면에 출력하라.

출력 예시:
삼성전자
LG전자
Naver
SK하이닉스
미래에셋대우

69. 문자열 split 메서드

▶ 회사 이름이 슬래시 ('/')로 구분되어 하나의 문자열로 저장되어 있다.

string = "삼성전자/LG전자/Naver"
이를 interest 이름의 리스트로 분리 저장하라.

실행 예시

```
>> print(interest)
['삼성전자', 'LG전자', 'Naver']
```

70. 리스트 정렬

▶ 리스트에 있는 값을 오름차순으로 정렬하세요.

```
data = [2, 4, 3, 1, 5, 10, 9]
```

5. 파이썬 튜플

71.

▶ my_variable 이름의 비어있는 튜플을 만들고, 변수가 튜플인지를 확인하라.

72.

▶ 2016년 11월 영화 예매 순위 기준 top3는 다음과 같다. 영화 제목을 movie_rank 이름의 튜플에 저장하라. (순위 정보는 저장하지 않는다.)

순위	영화
1	닥터 스트레인지
2	스플릿
3	럭키

73.

▶ 숫자 1 이 저장된 튜플을 생성하라.

74.

▶ 다음 코드를 실행해보고 오류가 발생하는 원인을 설명하라.

```
>> t = (1, 2, 3)
>> t[0] = 'a'
Traceback (most recent call last):
  File "<pyshell#46>", line 1, in <module>
    t[0] = 'a'
TypeError: 'tuple' object does not support item assignment
```

75.

▶ 아래와 같이 t에는 1, 2, 3, 4 데이터가 할당되어 있다. t가 할당하는 데이터 타입은 무엇인가?

```
t = 1, 2, 3, 4
```

76.

▶ 변수 t에는 아래와 같은 값이 저장되어 있다. 변수 t가 ('A', 'b', 'c') 튜플을 가리키도록 수정 하라.

```
t = ('a', 'b', 'c')
```

77.

▶ 다음 튜플을 리스트로 변환하라.

```
interest = ('삼성전자', 'LG전자', 'SK Hynix')
```

78.

▶ 다음 리스트를 튜플로 변경하라.

```
interest = ['삼성전자', 'LG전자', 'SK Hynix']
```

79. range 함수

▶ 1 부터 99까지의 정수 중 짝수만 저장된 튜플을 생성하라.

```
(2, 4, 6, 8 ... 98)
```

80. 튜플 언팩킹

▶ 다음 튜플 temp의 각 요소 값을 한번에 변수 a, b, c에 할당하라.

```
temp = ('apple', 'banana', 'cake')
```

예시)
```
a = 'apple'
b = 'banana'
c = 'cake'
```

81. 별 표현식

▶ 기본적으로 데이터 언패킹은 좌변의 변수와 우변 데이터 개수가 같아야 합니다. 하지만 star expression을 사용하면 변수의 개수가 달라도 데이터 언패킹을 할 수 있습니다. 튜플에 저장된 데이터 중에서 앞에 있는 두 개의 데이터만 필요할 경우 나머지 데이터의 언패킹 코드를 작성할 필요가 없습니다.

```
>> a, b, *c = (0, 1, 2, 3, 4, 5)
>> a
0
>> b
1
>> c
[2, 3, 4, 5]
```

다음과 같이 10개의 값이 저장된 scores 리스트가 있을 때, start expression을 사용하여 좌측 8개의 값을 valid_score 변수에 할당하여라.

```
scores = [8.8, 8.9, 8.7, 9.2, 9.3, 9.7, 9.9, 9.5, 7.8, 9.4]
```

82.

▶ 다음과 같이 10개의 값이 저장된 scores 리스트가 있을 때, star expression을 사용하여 우측 8개의 값을 valid_score 변수에 할당하여라.

scores = [8.8, 8.9, 8.7, 9.2, 9.3, 9.7, 9.9, 9.5, 7.8, 9.4]

83.

▶ 다음과 같이 **10**개의 값이 저장된 **scores** 리스트가 있을 때**, star expression**을 사용하여 가운데 있는 **8**개의 값을 **valid_score** 변수에 할당하여라**.**

scores = [8.8, 8.9, 8.7, 9.2, 9.3, 9.7, 9.9, 9.5, 7.8, 9.4]

6. 파이썬 딕셔너리

84. 비어있는 딕셔너리

▶ temp 이름의 비어있는 딕셔너리를 만들라.

85.

▶ 다음 아이스크림 이름과 희망 가격을 딕셔너리로 구성하라.

이름 희망 가격
메로나 1000
폴라포 1200
빵빠레 1800

86.

▶ 85 번의 딕셔너리에 아래 아이스크림 가격정보를 추가하라.

이름 희망 가격
죠스바 1200
월드콘 1500

87.

▶ 다음 딕셔너리를 사용하여 메로나 가격을 출력하라.

```
ice = {'메로나': 1000,
       '폴로포': 1200,
       '빵빠레': 1800,
       '죠스바': 1200,
       '월드콘': 1500}
```

실행 예:
메로나 가격: 1000

88.

▶ 다음 딕셔너리에서 메로나의 가격을 1300으로 수정하라.

```
ice = {'메로나': 1000,
       '폴로포': 1200,
       '빵빠레': 1800,
       '죠스바': 1200,
       '월드콘': 1500}
```

89.

▶ 다음 딕셔너리에서 메로나를 삭제하라.

```
ice = {'메로나': 1000,
       '폴로포': 1200,
       '빵빠레': 1800,
       '죠스바': 1200,
       '월드콘': 1500}
```

90.

▶ 다음 코드에서 에러가 발생한 원인을 설명하라.

```
>> icecream = {'폴라포': 1200, '빵빠레': 1800, '월드콘': 1500, '메로나': 1000}
>> icecream['누가바']
Traceback (most recent call last):
  File "<pyshell#69>", line 1, in <module>
    icecream['누가바']
KeyError: '누가바'
```

91. 딕셔너리 생성

▶ 아래의 표에서, 아이스크림 이름을 키값으로, (가격, 재고) 리스트를 딕셔너리의 값으로 저장하라. 딕셔너리의 이름은 **inventory**로 한다.

이름	가격	재고
메로나	300	20
비비빅	400	3
죠스바	250	100

92. 딕셔너리 인덱싱

▶ **inventory** 딕셔너리에서 메로나의 가격을 화면에 출력하라.

```
inventory = {"메로나": [300, 20],
             "비비빅": [400, 3],
             "죠스바": [250, 100]}
```

실행 예시:
300 원

93. 딕셔너리 인덱싱

▶ **inventory** 딕셔너리에서 메로나의 재고를 화면에 출력하라.

```
inventory = {"메로나": [300, 20],
             "비비빅": [400, 3],
             "죠스바": [250, 100]}
```

실행 예시:
20 개

94. 딕셔너리 추가

▶ **inventory** 딕셔너리에 아래 데이터를 추가하라.

```
inventory = {"메로나": [300, 20],
             "비비빅": [400, 3],
             "죠스바": [250, 100]}

이름   가격   재고
월드콘 500   7
```

실행 예시:

```
>> print(inventory)
{'메로나': [300, 20], '비비빅': [400, 3], '죠스바': [250, 100], '월드콘': [500, 7]}
```

95. 딕셔너리 keys() 메서드

▶ 다음의 딕셔너리로부터 key 값으로만 구성된 리스트를 생성하라.

```
icecream = {'탱크보이': 1200, '폴라포': 1200, '빵빠레': 1800, '월드콘': 1500, '메로나': 1000}
```

96. 딕셔너리 values() 메서드

▶ 다음의 딕셔너리에서 values 값으로만 구성된 리스트를 생성하라.

```
icecream = {'탱크보이': 1200, '폴라포': 1200, '빵빠레': 1800, '월드콘': 1500, '메로나': 1000}
```

97. 딕셔너리 values() 메서드

▶ icecream 딕셔너리에서 아이스크림 판매 금액의 총합을 출력하라.

```
icecream = {'탱크보이': 1200, '폴라포': 1200, '빵빠레': 1800, '월드콘': 1500, '메로나': 1000}
```

출력 예시:

```
6700
```

98. 딕셔너리 update 메서드

▶ 아래의 new_product 딕셔너리를 다음 icecream 딕셔너리에 추가하라.

```
icecream = {'탱크보이': 1200, '폴라포': 1200, '빵빠레': 1800, '월드콘': 1500, '메로나': 1000}
new_product = {'팥빙수':2700, '아맛나':1000}
```

실행 예시:

```
>> print(icecream)
{'탱크보이': 1200, '폴라포': 1200, '빵빠레': 1800, '월드콘': 1500, '메로나': 1000, '팥빙수':2700, '아맛나':1000}
```

99. zip과 dict

▶ 아래 두 개의 튜플을 하나의 딕셔너리로 변환하라. keys를 키로, vals를 값으로 result 이름의 딕셔너리로 저장한다.

```
keys = ("apple", "pear", "peach")
vals = (300, 250, 400)
```

실행 예시:

```
>> print(result)
{'apple': 300, 'pear': 250, 'peach': 400}
```

100. zip과 dict

▶ date와 close_price 두 개의 리스트를 close_table 이름의 딕셔너리로 생성하라.

```
date = ['09/05', '09/06', '09/07', '09/08', '09/09']
close_price = [10500, 10300, 10100, 10800, 11000]
```

실행 예시:

```
>> print(close_table)
{'09/05': 10500, '09/06': 10300, '09/07': 10100, '09/08': 10800, '09/09': 11000}
```