파이썬 포트폴리오

컴퓨터정보공학과 20165505 김태민

목차

Chap01

파이썬의 언어의 개요와 첫 프로그래밍

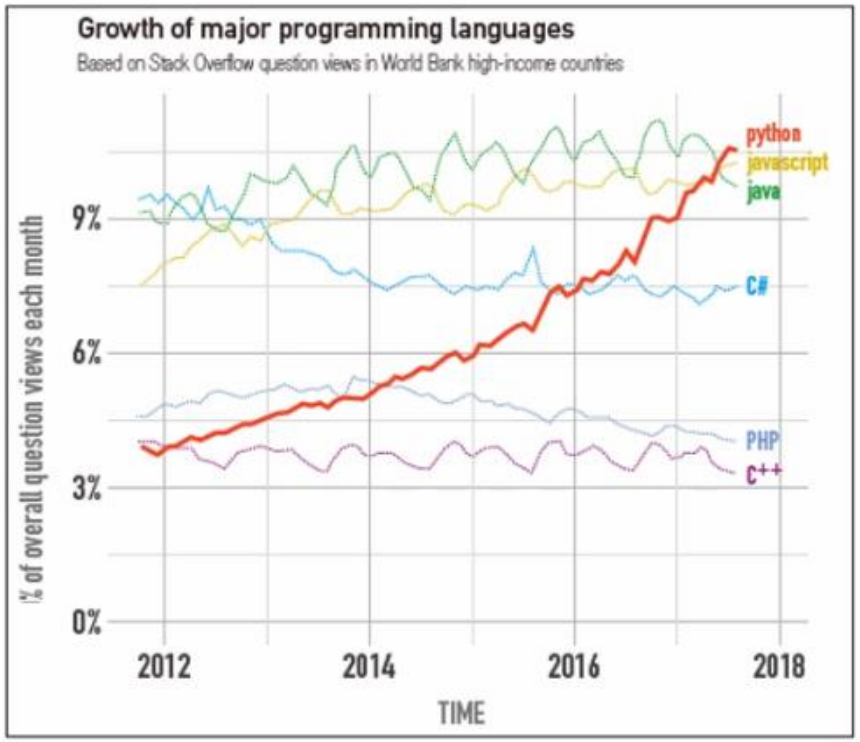
파이썬은 배우기 쉬운 프로그래밍 언어!!

=> 오픈 소스(open source) 프로그래밍 언어 (programming language)

=> 1991년 네덜란드의 귀도 반 로섬(Guido van Rossum)이 개발

=> 현재는 비영리 단체인 파이썬 소프트웨어 재단 (PSF: Python Software Foundation)이 관리

현재 파이썬의 인기



▲ 그림3 스택오버플로에서 조사한 최근에 빠르게 성장하는 언어 비교

파이썬을 활용한 프로그래밍 교육!!

=> 컴퓨터 전공자: 전문적인 프로그래밍 절차와 기술을 교육

=> 비전공자: 컴퓨터 사고력을 교육하는 데 적합

▼ 표1 코딩 절차와 컴퓨팅 사고력 요소

코딩 절차	컴퓨팅 사고력 요소 활용	내용
이해		• 주어진 문제를 이해하고 파악 • 분할: 문제를 좀 더 쉬운 작은 문제들로 분해
설계		• 패턴 인식: 분해된 문제들 사이의 유사성 또는 패턴을 탐색 • 추상화: 문제에서 불필요한 부분은 제거하고 주요 핵심 요소를 추려 내는 과정 • 알고리즘: 프로그래밍 언어인 파이썬에 맞는 입력과 출력, 변수 저장 그리고 절차에 따라 구성
구현		• 문제 해결을 위해 파이썬으로 코드 개발 • 작성된 코드의 실행과 테스팅(testing, simulation), 디버깅(debugging) 과정을 거치면서 코드를 수정하고 필요하면 다시 설계
공유	협업과 평가 (Collaboration & Evaluation)	• 자신이 구현한 프로그램을 발표하고 다른 학습자의 프로그램과 비교 • 현재의 기능을 향상시키는 방향으로 프로그램 개선 연구 • 교수자의 피드백 및 평가

컴퓨팅 사고력 유디스(UDIS)!!

=> 이해(U: Understand), 설계(D: Design), 구현(I: Implement), 공유(S: Share)의 절차

이해
=> 주어진 문제를 이해하고 파악

설계
=> 패턴을 탐색하고 추상화, 절차에 따른 구성

구현
=> 문제 해결을 위해 파이썬 코드 개발

공유
=> 자신이 구현한 프로그램을 발표하고 다른 학습잡의 프로그램과 비교

인터프리터 방식의 언어, 파이썬!!

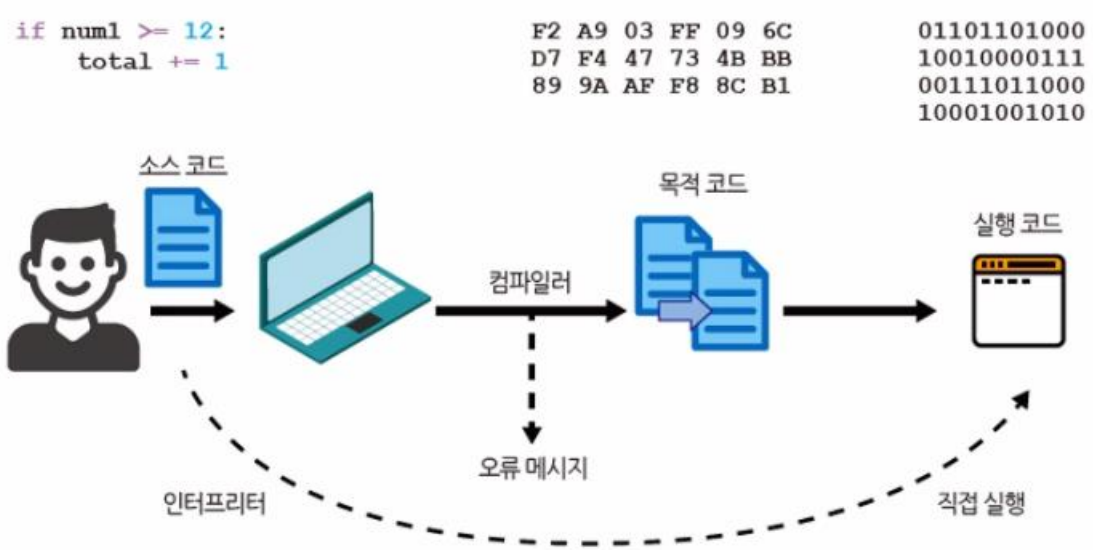
=> 동시 통역처럼, 프로그램의 코드가 한 줄씩 순서대로 해석되고 실행

=> 인터프리터(interpreter)
한 줄 한 줄의 해석을 담당하는 프로그램

파이썬은 베이직(basic) 언어와 마찬가지로 인터프리터 방식의 언어이며 자바와 C는 컴파일 방식의 언어이다.

컴파일 방식!!

=> 컴파일러(compiler)
컴파일(compiler)을 담당하는 개발 도구 소프트웨어



▲ 그림29 인터프리터와 컴파일러의 차이

Chap02

파이썬 프로그래밍을 위한 기초 다지기

문자열(String)!!

=> 일련(sequence)의 문자(character) 모임

=> 작은따옴표나 큰따옴표를 앞뒤로 둘러싸 표현

=> 파이썬은 문자와 문자열을 구분하지 않는다. 하나의 문자도 문자열로 판별한다.

```
>>> 'p'
'p'
>>> "python"
'python'
>>> '27'
'27'
>>> "3.14"
'3.14'
>>> print("Hello World!")
Hello World!
```

print() 함수!!

=> print() 함수는 코딩에서 콘솔에 자료를 출력하는 작업을 도와주는 역할은 한다고 생각하면 이해하기 쉽다.

문자열 연산자 +, *

=> + 문자열에서 문자열 연결을 연결

=> * 문자열에서 문자열을 지정된 수만큼 반복

```
>>> print("원의 원주율 " + "3.141592")
원의 원주율 3.141592
>>> print("python " 'programming ' + 'language')
python programming language
>>> print("파이썬 언어는" + " 강력하다")
파이썬 언어는 강력하다
>>> print("파이썬 " + "언어! " + 3 * "방가 ")
파이썬 언어! 방가 방가 방가
```

수의 더하기(+), 빼기(-), 곱하기(*), 나누기 연산자(/)

```
>>> 5 + 3
8
>>> 5 - 3
2
>>> 5 * 3
15
>>> 5 / 3
1.6666666666666667
```

수의 몫(//), 나머지(%), 지수승 연산자(**)

=> //은 소수부 없이 정수만 남는다

```
>>> 8 // 5
1
>>> 5 % 3
2
>>> 5 ** 3
125
```

최근 결과의 특별한 저장 장소, 언더스코어(_)!!

=> 마지막에 실행된 결괏값은 특별한 저장 공간인 _에 대입된다.

```
>>> 60
60
>>> _
60
>>> 3 * _
180
```

표현식 문자열 실행 함수 eval()

=> 함수 eval('expression')은 실행 가능한 연산식 문자열인 expression을 실행한 결과를 반환한다.

```
>>> eval('3 + 15 / 2')
10.5
>>> eval('"java " * 3')
'java java java '
```

자료형과 type() 함수!!

=> 정수와 실수, 문자열 등을 자료형(data type)이라 한다.

```
>>> type(3)
<class 'int'>
>>> pi = 3.14
>>> type(pi)
<class 'float'>
>>> type('python')
<class 'str'>
>>> type(3 + 4j)
<class 'complex'>
```

변수와 대입연산자

=> 변수(variable)란, '변하는 자료를 저장하는 메모리 공간' 이다.

```
>>> a = b = c = 5
>>> print(a)
5
>>> print(b)
5
>>> print(c)
5
```

변수 이름을 붙일 때의 규칙

=> 문자는 대소문자의 영문자, 숫자, 그리고 _로 구성되며 대소문자는 구별이 된다.

=> 숫자는 맨 앞에 올 수 없다. 그러므로 영문자로 시작하자.

=> import, True, False 등 같은 키워드는 변수로 사용할 수 없다.

```
>>> value = 20
>>> print(value)
20
>>> value = 100
>>> print(value)
100
>>> 2020year = 2020
SyntaxError: invalid syntax
>>> sale@ = 20
SyntaxError: invalid syntax
>>> sale price = 16000
SyntaxError: invalid syntax
>>> import = 30
SyntaxError: invalid syntax
```

숫자가 맨앞에 왔으므로 에러!!
특수문자 @ 에러
공백 문자 에러
키워드 에러

divmod(a, b)

=> 나누기 목 연산 //와 나머지 연산 %를 함께 수행한다.

```
>>> divmod(17, 5)
(3, 2)
>>> d, m = divmod(17, 5)
>>> print(d, m)
3 2
```

d는 몫, m은 나머지 이다

표준 입력 함수 input()!!

=> 프로그램 작성 과정에서 사용자의 입력을 받아 처리해야 하는 경우 사용한다.

```
>>> input()
java
'java'
>>> univ = input("대학은? ")
대학은? 동양미래대학교
>>> print(univ)
동양미래대학교
```

문자열과 정수, 실수 간의 자료 변환 함수 str(), int(), float()

```
>>> str(235)
'235'
>>> int('6400')
6400
>>> float('3.141592')
3.141592
```

10진수의 변환 함수 bin(), oct(), hex()

=> 각각 10진수를 2진수, 8진수, 16진수로 변환

```
>>> bin(7), bin(16), bin(12)
('0b111', '0b10000', '0b1100')
>>> oct(8), oct(10), oct(12)
('0o10', '0o12', '0o14')
>>> hex(12), hex(15), hex(16)
('0xc', '0xf', '0x10')
```

Chap2 과제

1. 두 문자열을 표준 입력으로 받아 한 줄에 출력하는 프로그램 작성

```
st = input('문자열 1: ')
st2 = input('문자열 2: ')
print(st + ' ' + st2)
```

결과

```
문자열 1: python
문자열 2: 언어
python 언어
```

2. 아메리카노를 주문받아 총 가격을 출력하는 프로그램을 작성

```
a = 3500
st = input('아메리카노 몇 개 주문하세요? ')
money = a * int(st)
print('총 가격은', money, '이다.')
```

결과

```
아메리카노 몇 개 주문하세요? 7
총 가격은 24500 이다.
```

3. 섭씨 온도를 입력받아 화씨 온도로 변환하는 프로그램 작정

```
c = int(input('온도 입력 >> '))
f = c * 9 / 5 + 32
print('정확 계산: ', c, ', 화씨: ', f)
f2 = c * 2 + 30
print('약식 계산: ', c, ', 화씨: ', f2)
m = f2 - f
print('차이: ',m)
```

결과

```
온도 입력 >> 38
정확 계산:  38 , 화씨:  100.4
약식 계산:  38 , 화씨:  106
차이:  5.599999999999994
```

4. 두 16진수 실수를 입력받아 사칙연산을 수행하는 프로그램 작성

```
str = input('첫 번째 16진수 실수 입력 >> ')
str2 = input('두 번째 16진수 실수 입력 >> ')
f = float.fromhex(str)
f2 = float.fromhex(str2)
print('실수1: ', f, '실수2:', f2)
sum = f + f2
mus = f - f2
xam = f * f2
nam = f / f2
print('합: ', sum)
print('차: ', mus)
print('곱하기: ', xam)
print('나누기: ', nam)
```

결과

```
첫 번째 16진수 실수 입력 >> f
두 번째 16진수 실수 입력 >> e.1
실수1:  15.0 실수2: 14.0625
합:  29.0625
차:  0.9375
곱하기:  210.9375
나누기:  1.0666666666666667
```

나만의 프로젝트 Lab 코딩

1. 표준 입력으로 2개의 실수를 입력받아 더하기, 빼기, 곱하기, 나누기의 연산을 출력, 이후 다시 표준 입력으로 하나의 연산식을 입력받아 그 결과를 출력

```
num1 = float(input("첫 번째 실수 입력 ==> "))
num2 = float(input("두 번째 실수 입력 ==> "))

print("합: ", num1 + num2)
print("차: ", num1 - num2)
print("곱하기: ", num1 * num2)
print("나누기: ", num1 / num2)

expression = input("연산식 입력 >> ")

print("연산식: ", expression, "결과: ", eval(expression))
```

결과

```
첫 번째 실수 입력 ==> 3.4
두 번째 실수 입력 ==> 1.5
합:  4.9
차:  1.9
곱하기:  5.1
나누기:  2.2666666666666666
연산식 입력 >> 3 * 4 / 2
연산식:  3 * 4 / 2 결과:  6.0
```

2. 가장 먼저 변환할 수가 2진수, 8진수 16진수 중에 무엇인지 입력받은 후 표준 입력한 수의 2진수 8진수, 10진수, 16진수를 출력

```
base = int(input("입력할 정수의 진수(base)는? "))

number = input(str(base) + "진수 정수 입력 >> ")
data = int(number, base)
print("2진수", bin(data))
print("8진수", oct(data))
print("10진수", data)
print("16진수", hex(data))
```

결과

```
입력할 정수의 진수(base)는? 2
2진수 정수 입력 >> 1011
2진수 0b1011
8진수 0o13
10진수 11
16진수 0xb
```

Chap03

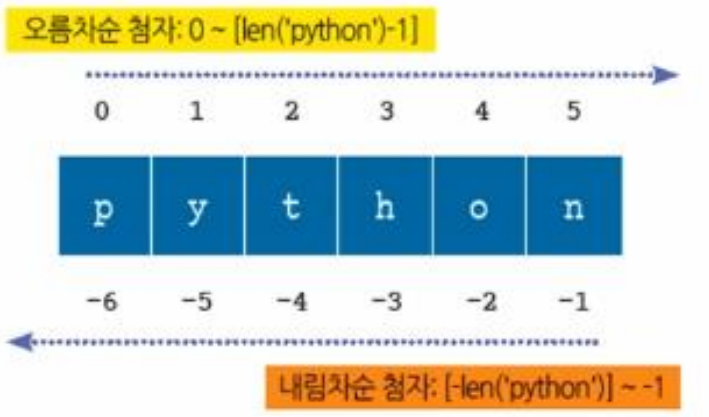
일상에서 활용되는 문자열과 논리 연산

함수 len()으로 문자열의 길이 참조

```
>>> a = 'python'
>>> len(a)
6
>>> len("파이썬")
3
```

문자열의 부분 문자열 참조 방식

=>문자열을 구성하는 문자는 0부터 시작되는 첨자(index)를 대괄호 안에 참조(indexing)가 가능하다. 또한 -1부터 시작해 -2, -3 작아지는 역순도 가능하다.



문자열[start: end: step]

=> start는 첨자의 시작, end는 첨자의 끝, step은 간격조정 생략시 기본값은 1 이다.

= >start, end는 비울 수 있으며 각각 '처음부터', '끝까지' 를 의마한다

```
>>> 'python'[1:5]
'ytho'
>>> 'python'[2:4]
'th'
>>> 'python'[-5:-1]
'ytho'
>>> 'python'[1:]
'ython'
>>> 'python'[0:5:2]
'pto'
```

문자열 관련 메소드

문자열을 바꿔 반환하는 replace()

=> 메소드 str.replace(a, b, c)는 문자열 str에서 a가 나타나는 모든 부분을 b로 모두 바꾼 문자열을 반환한다. c는 대체 횟수를 의미한다

```
>>> str = '파이썬 파이썬 파이썬'
>>> str.replace('파이썬', 'python')
'python python python'
>>> str.replace(" ", "")
'파이썬파이썬파이썬'
>>> str.replace("파이썬", "python", 2)
'python python 파이썬'
```

문자열 출현 횟수를 반환하는 메소드 count()

```
>>> str = "단순한 것이 복잡한 것보다 낫다."
>>> str.count("복잡")
1
>>> str.count("것")
2
```

문자와 문자 사이에 문자열을 삽입하는 join()

```
>>> num = "12345"
>>> "=>".join(num)
'1=>2=>3=>4=>5'
```

문자열을 찾는 메소드 find()와 index()

=> str.index(문자열)은 찾는 문자열이 없을 경우 ValuesError를 발생시키지만 find()는 -1를 반환하며, 있는 경우 해당 문자열의 첨자를 반환한다.

```
>>> str = "자바 C 파이썬 코틀린"
>>> str.find("자바")
0
>>> str.index("자바")
0
>>> str.find("C++")
-1
>>> str.index("C++")
Traceback (most recent call last):
  File "<pyshell#93>", line 1, in <module>
    str.index("C++")
ValueError: substring not found
```

문자열을 여러 문자열로 나누는 split() 메소드

=> 문자열 str의 공백 만큼 리스트형식으로 만들어준다. 단 split(', ') 특정 문자열이 있을 경우 이 부분을 구분자로 이용해 문자열을 나눠 준다.

```
>>> "사과 배 복숭아 딸기 포도".split()
['사과', '배', '복숭아', '딸기', '포도']
>>> "데스크톱 1000000, 노트북 1800000, 스마트폰 1200000".split(',')
['데스크톱 1000000', ' 노트북 1800000', ' 스마트폰 1200000']
```

다양한 문자열 변환 메소드

문자열 모두 대문자로 반환해주는 upper()
소문자로 반환해주는 lower()

```
>>> 'python'.upper()
'PYTHON'
>>> 'PYTHON'.lower()
'python'
```

폭을 지정하고 좌측, 우측, 중앙에 문자열을 배치하는 메소드 ljust(), rjust(), center()

center(a, b) => a는 폭을 의미, b는 폭에 채울 문자

```
>>> 'python'.ljust(10)
'python    '
>>> 'python'.rjust(10)
'    python'
>>> 'python'.center(10, '*')
'**python**'
```

문자열 format()을 이용한 간단한 출력 처리

=> 문자열 중간중간에 변수나 상수를 함께 출력하는 방법인 포맷팅

```
>>> a, b = 4, 10
>>> print("{} * {} = {}".format(a, b, a * b))
4 * 10 = 40
>>> print("{0} * {1} = {2}".format(a, b, a * b))
4 * 10 = 40
```

=> {0}, {1} 인자 처럼 인자의 순서를 나타내는 정수

C언어 포맷팅 방식 format()방법

```
>>> c, d = 20, 30
>>> print("%d + %d = %d" % (c, d, c + d))
20 + 30 = 50
```

Chap3 과제

1. 한 줄의 문자열을 표준 입력으로 받아 문자열의 소속 문자를 참조할 범위를 출력하고 다시 첨자 하나를 입력받아 문자열의 전체와 길이 그리고 참조 문자를 출력하는 프로그램 작성

```
str = input('문자열 입력 >> ')
n = len(str)
print('참조할 첨자: 0 ~ ', n - 1)
str2 = int(input('참조할 첨자 입력 >> '))
print('문자열: ', str, '길이: ', n)
print("참조 문자: ", str[str2])
```

결과

```
문자열 입력 >> python is a good language!
참조할 첨자: 0 ~  25
참조할 첨자 입력 >> 12
문자열:  python is a good language! 길이:  26
참조 문자:  g
```

2. 파이썬의 다음 첫번째 철학을 저장한 후, 메소드 replace()를 사용해 두 번째 철학으로 다시 저장해 출력하는 프로그램 작성

Beauitiful is better than ugly.
Explicit is better than implicit.

```
str = 'Beautiful is better than ugly.'
print(str)
print('위 철학을 메소드 replace()를 사용해 다음 철학으로 다시 저장')
str2 = str.replace('Beautiful', 'Explicit')
str3 = str2.replace('ugly.', 'implicit.')
print(str3)
```

결과

```
Beautiful is better than ugly.
위 철학을 메소드 replace()를 사용해 다음 철학으로 다시 저장
Explicit is better than implicit.
```

3. 문자열의 split() 메소드를 사용해 '14:21:45'와 같은 시각 정보를 표준 입력으로 받아 입력된 문자열을 출력하고 다시 시, 분, 초로 출력하는 프로그램 작성

```
str = input('시각 정보(16:30:15) 입력 >> ')
print('입력 시각 정보: ', str)
hours, mins, secs = str.split(':')
print(hours,'시 ', mins, '분', secs, '초')
```

결과

```
시각 정보(16:30:15) 입력 >> 23:03:05
입력 시각 정보:  23:03:05
23 시  03 분 05 초
```

4. -7에서 0까지 10진수와 2진수, 8진수, 16진수를 다음과 같이 정형화 출력하는 프로그램 작성

```
mask = 0xff
print('10진수: {0:2} 2진수: {1:08b} 8진수: {1:5o} 16진수 {1:3x}'.format(-7, -7 & mask))
print('10진수: {0:2} 2진수: {1:08b} 8진수: {1:5o} 16진수 {1:3x}'.format(-6, -6 & mask))
print('10진수: {0:2} 2진수: {1:08b} 8진수: {1:5o} 16진수 {1:3x}'.format(-5, -5 & mask))
print('10진수: {0:2} 2진수: {1:08b} 8진수: {1:5o} 16진수 {1:3x}'.format(-4, -4 & mask))
print('10진수: {0:2} 2진수: {1:08b} 8진수: {1:5o} 16진수 {1:3x}'.format(-3, -3 & mask))
print('10진수: {0:2} 2진수: {1:08b} 8진수: {1:5o} 16진수 {1:3x}'.format(-2, -2 & mask))
print('10진수: {0:2} 2진수: {1:08b} 8진수: {1:5o} 16진수 {1:3x}'.format(-1, -1 & mask))
print('10진수: {0:2} 2진수: {1:08b} 8진수: {1:5o} 16진수 {1:3x}'.format(0, 0 & mask))
```

결과

```
10진수: -7 2진수: 11111001 8진수:   371 16진수  f9
10진수: -6 2진수: 11111010 8진수:   372 16진수  fa
10진수: -5 2진수: 11111011 8진수:   373 16진수  fb
10진수: -4 2진수: 11111100 8진수:   374 16진수  fc
10진수: -3 2진수: 11111101 8진수:   375 16진수  fd
10진수: -2 2진수: 11111110 8진수:   376 16진수  fe
10진수: -1 2진수: 11111111 8진수:   377 16진수  ff
10진수:  0 2진수: 00000000 8진수:     0 16진수   0
```

나만의 프로젝트 Lab1, 2 코딩

1. 총 가격을 입력받아 해당 조건에 맞는 원가격, 할인된 가격, 할인율, 할인액을 출력

```
price = int(input("총 가격(원 가격) 입력 >> "))

rate1 = (10000 <= price < 20000) * 1 / 100
rate2 = (20000 <= price < 40000) * 2 / 100
rate3 = (40000 <= price) * 4 / 100
rate = rate1 + rate2 + rate3

discount = price * rate
disprice = price - discount
print("원 가격: ", price, "할인된 가격: ", disprice)
print("할인율: ", rate, "할인액: ", discount)
```

결과

```
총 가격(원 가격) 입력 >> 50000
원 가격:  50000 할인된 가격:  48000.0
할인율:  0.04 할인액:  2000.0
```

2. 표준 입력으로 3개의 단어를 콤마로 구분해 입력받고 3개의 단어를 추출해 각 단어와 역순의 단어를 출력 그리고 그단어가 회문인지를 판단하는 논리 값도 출력하는 프로그램 작성

```
str = input("콤마로 구분된 단어 4개 입력 >> ")
str = str.replace(' ', '')
word1, word2, word3 = str.split(',')

print("단어: {}, 역순: {}, 회문: {}".format(word1, word1[::-1]
    , (word1 == word1[::-1])))
print("단어: {}, 역순: {}, 회문: {}".format(word2, word2[::-1]
    , (word2 == word2[::-1])))
print("단어: {}, 역순: {}, 회문: {}".format(word3, word3[::-1]
    , (word3 == word3[::-1])))
```

결과

```
콤마로 구분된 단어 4개 입력 >> 여보보여, 파이썬, level
단어: 여보보여, 역순: 여보보여, 회문: True
단어: 파이썬, 역순: 썬이파, 회문: False
단어: level, 역순: level, 회문: True
```

Chap04

일상생활과 비유되는 조건과 반복

조건에 따른 선택을 결정하는 if문

=> if문에서 논리 표현식 이후에는 반드시 콜론(:)이 있어야 한다.
=> 콜론 이후 다음 줄부터 시작되는 블록은 반드시 들여쓰기(indentation) 해야한다 그렇지 않으면 오류 발생

```
>>> pm = 90
>>> if 81 < pm:
        print("마스크를 착용합시다")


마스크를 착용합시다
>>>
```

=> pm이 81보다 크므로 True이며 print()을 호출

논리 표현식 결과가 True와 False에 따라 나뉘는 if else문

=> if문이 True이면 콜론 이후 블록을 실행하며 False일 경우 else: 이후의 블록을 실행한다.

```
>>> n = 20
>>> if n % 2 == 0:
        print("짝수이다")
else:
        print("홀수이다")


짝수이다
```

다중 택일 결정 구조인 if elif문

=> 조건의 여러 경로 중 하나를 선택하는 조건문이다

```
pm = float(input("미세먼지(10마이크로그램)의 농도는 ?"))
if 151 <= pm:
    print("미세먼지 농도: {:.2f}, 등급: {}".format(pm, "매우나쁨"))
elif 81 <= pm:
    print("미세먼지 농도: {:.2f}, 등급: {}".format(pm, "나쁨"))
elif 31 <= pm:
    print("미세먼지 농도: {:.2f}, 등급: {}".format(pm, "보통"))
else:
    print("미세먼지 농도: {:.2f}, 등급: {}".format(pm, "좋음"))


미세먼지(10마이크로그램)의 농도는 ?82
미세먼지 농도: 82.00, 등급: 나쁨
```

항목 값으로 반복을 실행하는 for문

=> 여러 자료 값이 순서대로 구성된 시퀀스에서 자료 값의 개수만큼 반복적인 작업을 수행

```
>>> for i in 1, 2, 3, 4, 5, 6:
        print("%d " % (i), end = " ")
        print("번째 숫자")


1  번째 숫자
2  번째 숫자
3  번째 숫자
4  번째 숫자
5  번째 숫자
6  번째 숫자
```

i in 1, 2, 3, 4, 5, 6:

i는 반복몸체가 수행할때마다 1부터 6까지 차례대로 초기화대며, 마지막은 콜론으로 마무리

반복몸체는 들여쓰기한 print() 2개를 실행 즉, 들여쓰기가 들어간 문장만 실행

내장 함수 range()

=> 반복 for문의 시퀀스에 내장 함수 range()를 활용하면 매우 간단하다.

```
>>> for i in range(1, 10, 2):
        print(i, end = " ")


1 3 5 7 9
```

range(start, end, step)

start: 시퀀스의 시작
stop: 시퀀스의 끝, 실질적으로 stop-1 까지이다.
step: 증가값

횟수가 정해지지 않은 반복 while

=> while문은 횟수가 정해놓지 않고 어떤 조건이 False가 될 때까지 반복을 수행하는데 적합하다.

while 논리표현식:
        문장 1
        문장 2
else:
        문장3

=> while문 앞에 논리 표현식이 위치하며 반드시 콜론이 필요하다

=> 논리표현식이 False가 될 때까지 문장1, 문장2를 반복 else:은 False가 돼 반복이 종료된 후 문장3을 실행

```
>>> n = 1
>>> while n <= 5:
        print(n, end = " ")
        n += 1
else:
        print("₩n 반복 while 종료: n => %d" % n)


1 2 3 4 5
 반복 while 종료: n => 6
```

반복을 제어하는 break문과 continue문

=> while문의 논리 표현식이 True이면 계속적으로 반복되어 이를 무한 반복이라 한다.

=>for문이나 while문의 반복 내부에서 문장 break는 else: 실행하지 않고 반복 종료 한다.

```
while True:
    menu = input("[0]종료 [1]계속 ? ")
    if menu == "0":
        break
print("종료")
```

```
[0]종료 [1]계속 ? 1
[0]종료 [1]계속 ? 1
[0]종료 [1]계속 ? 1
[0]종료 [1]계속 ? 1
[0]종료 [1]계속 ? 0
종료
```

=> 위에 코드를 보았듯이 무한 반복 while문에서 0 을 입력했을 때 종료되는것을 확인 할수 있다.

=> for문과 while문 내부에서 continue 문장은 이후의 반복 몸체를 실행하지 않고 다음 반복을 위해 논리 조건을 수행한다.

```
days = ["monday", "tuesday", "wednesday"]

while True:
    user = input("월, 화, 수 중 하나 영어 단어 입력 >> ")
    if user not in days:
        print("잘못 입력했어요!")
        continue
    print("입력: %s, 철자가 맞습니다." % user)
    break

print(" 종료 ".center(15, "*"))
```

```
월, 화, 수 중 하나 영어 단어 입력 >> 월요일
잘못 입력했어요!
월, 화, 수 중 하나 영어 단어 입력 >> 화요일
잘못 입력했어요!
월, 화, 수 중 하나 영어 단어 입력 >> wednesday
입력: wednesday, 철자가 맞습니다.
****** 종료 ******
```

=> days에 있는 값이 아닌 값을 입력하면 continue로 인해 반복몸체가 처음부터 다시 시작하는 코딩이다.

Chap4 과제

1. 월을 표준 입력으로 입력받아 계절을 출력하는 프로그램 작성

```
mon = int(input('월 입력 ? '))

if (4 <= mon & mon <= 5):
    print('{}월 봄'.format(mon))
elif (6 <= mon & mon <= 8):
    print('{}월 여름'.format(mon))
elif (9 <= mon & mon <= 10):
    print('{}월 가을'.format(mon))
else:
    print('{}월 겨울'.format(mon))
```

결과

```
월 입력 ? 3
3월 겨울
```

2. 1에서 99까지의 난수인 임의의 정수를 3개를 대상으로 가장 큰 정수를 출력하는 프로그램 작성

```
from random import randint
a = randint(1, 99)
b = randint(1, 99)
c = randint(1, 99)

if (a > b):
    d = a
else:
    d = b
if (d < c):
    d = c
print(a, b, c, '중에서 최대: ', d)
```

결과

```
95 38 84 중에서 최대:  95
```

3. 섭씨온도 20도에서 41도까지 3도씩 증가하면서 화씨로 변환해 출력하는 프로그램 작성

```
for i in range(20, 44, 3):
    f = i * 9 / 5 + 32
    f2 = i * 2 + 30
    print('섭씨: {0} 화씨: {1} 화씨(약식) {2} 차이 {3:4.2f}'.format(i, f, f2, abs(f - f2)))
```

결과

```
섭씨: 20 화씨: 68.0 화씨(약식) 70 차이 2.00
섭씨: 23 화씨: 73.4 화씨(약식) 76 차이 2.60
섭씨: 26 화씨: 78.8 화씨(약식) 82 차이 3.20
섭씨: 29 화씨: 84.2 화씨(약식) 88 차이 3.80
섭씨: 32 화씨: 89.6 화씨(약식) 94 차이 4.40
섭씨: 35 화씨: 95.0 화씨(약식) 100 차이 5.00
섭씨: 38 화씨: 100.4 화씨(약식) 106 차이 5.60
섭씨: 41 화씨: 105.8 화씨(약식) 112 차이 6.20
```

4. 정수 1개를 표준 입력으로 받아 소수인지를 판별하는 프로그램 작성

```
a = int(input('소수(prime number)인지를 판별한 2 이상의 정수 입력 >> '))
b = 0

for i in range(2, a + 1):
    if (a % i == 0):
        b += 1
if (b == 1):
    print('정수 {}는 소수이다.'.format(a))
```

결과

```
소수(prime number)인지를 판별한 2 이상의 정수 입력 >> 11
정수 11는 소수이다.
```

나만의 프로젝트 Lab1, 2 코딩

1. 컴퓨터가 정한 1에서 10사이의 정수인 정답을 사용자가 맞히는 프로그램을 작성해 보자, 사용자가 정답을 맞출 때 까지 계속 힌트를 준다. 입력한 수보다 작은 수 또는 큰수 라는 정보를 제공

```
from random import randint

number = randint(1, 10)
unum = int(input("1에서 10사이의 수를 맞히세요 >> "))

while True:
    if number == unum:
        print("축하한다. {}: 정답이다".format(number))
        break;
    elif number > unum:
        print("{}보다 더 큰 수로 다시 입력 >> ".format(unum), end =" ")
    elif number < unum:
        print("{}보다 더 작은 수로 다시 입력 >> ".format(unum), end = " ")
    unum = int(input())

print(" 종료".center(30, '*'))
```

결과

```
1에서 10사이의 수를 맞히세요 >> 5
5보다 더 큰 수로 다시 입력 >>  4
4보다 더 큰 수로 다시 입력 >>  3
3보다 더 큰 수로 다시 입력 >>  8
축하한다. 8: 정답이다
************** 종료***************
```

```
menu = '''Coffee menu!
    1. 아메리카노      2000
    2. 카페라테        2500
    3. 카푸치노        3000
    4. 캐러멜마키아또 4000
    0. 주문 종료'''

print("환영합니다. 음료를 선택하세요")
total = 0
while True:
    print(menu)
    order = int(input("종류? "))
    if(order == 0):
        print(" 주문종료 ".center(18, '*'))
        break
    num = int(input("수량? "))
    if(order == 1):
        print("아메리카노 %d개 주문" % (num
        total += 2000 * num
    elif(order == 2):
        print("카페라테 %d개 주문" % (num))
        total += 2500 * num
    elif(order == 3):
        print("카푸치노 %d개 주문" % (num))
        total += 3000 * num
    elif(order == 4):
        print("캐러멜마키아또 %d개 주문" %
        total += 4000 * num
    print("현재 주문 가격: %d원" % (total))

print("총 주문 가격: %d원" % (total))
print(" 안녕! ".center(18, '*'))
```

```
환영합니다. 음료를 선택하세요
Coffee menu!
    1. 아메리카노      2000
    2. 카페라테        2500
    3. 카푸치노        3000
    4. 캐러멜마키아또 4000
    0. 주문 종료
종류? 1
수량? 1
아메리카노 1개 주문
현재 주문 가격: 2000원
Coffee menu!
    1. 아메리카노      2000
    2. 카페라테        2500
    3. 카푸치노        3000
    4. 캐러멜마키아또 4000
    0. 주문 종료
종류? 2
수량? 2
카페라테 2개 주문
현재 주문 가격: 7000원
Coffee menu!
    1. 아메리카노      2000
    2. 카페라테        2500
    3. 카푸치노        3000
    4. 캐러멜마키아또 4000
    0. 주문 종료
종류? 0
****** 주문종료 ******
총 주문 가격: 7000원
******* 안녕! *******
```

2. 커피 종류와 수량을 입력받아 총 주문 가격을 출력하는 프로그램 작성 0을 입려갛면 주문을 종료 1 ~4 까지 커피를 주문하면 바로 수량을 입력받고 주문가격도 표시

```
menu = '''Coffee menu!
    1. 아메리카노     2000
    2. 카페라테       2500
    3. 카푸치노       3000
    4. 캐러멜마키아또 4000
    0. 주문 종료'''

print("환영합니다. 음료를 선택하세요")
total = 0
while True:
    print(menu)
    order = int(input("종류? "))
    if(order == 0):
        print(" 주문종료 ".center(18, '*'))
        break
    num = int(input("수량? "))
    if(order == 1):
        print("아메리카노 %d개 주문" % (num))
        total += 2000 * num
    elif(order == 2):
        print("카페라테 %d개 주문" % (num))
        total += 2500 * num
    elif(order == 3):
        print("카푸치노 %d개 주문" % (num))
        total += 3000 * num
    elif(order == 4):
        print("캐러멜마키아또 %d개 주문" % (num))
        total += 4000 * num
    print("현재 주문 가격: %d원" % (total))

print("총 주문 가격: %d원" % (total))
print(" 안녕! ".center(18, '*'))
```

결과

```
환영합니다. 음료를 선택하세요
Coffee menu!
    1. 아메리카노     2000
    2. 카페라테       2500
    3. 카푸치노       3000
    4. 캐러멜마키아또 4000
    0. 주문 종료
종류? 1
수량? 1
아메리카노 1개 주문
현재 주문 가격: 2000원
Coffee menu!
    1. 아메리카노     2000
    2. 카페라테       2500
    3. 카푸치노       3000
    4. 캐러멜마키아또 4000
    0. 주문 종료
종류? 2
수량? 2
카페라테 2개 주문
현재 주문 가격: 7000원
Coffee menu!
    1. 아메리카노     2000
    2. 카페라테       2500
    3. 카푸치노       3000
    4. 캐러멜마키아또 4000
    0. 주문 종료
종류? 0
****** 주문종료 ******
총 주문 가격: 7000원
******* 안녕! *******
```

Chap05

항목의 나열인 리스트와 튜플

여러 항목(Item)을 관리하는 list

=> 리스트는 항목의 나열인 시퀀스다. 콤마로 구분된 항목들의 리스트로 표현된다.

=> print()로 리스트를 출력하면 대괄호의 리스트 형태로 출력할 수 있다.

```
>>> coffee = ['에스프레소', '아메리카노', '카페라테', '카페모카']
>>> print(coffee)
['에스프레소', '아메리카노', '카페라테', '카페모카']
>>> type(coffee)
<class 'list'>
```

빈 리스트의 생성과 항목 추가

=> 인자가 없는 내장 함수 list()로도 빈 리스트를 만들 수 있으며 항목을 추가하면 리스트의 가장 뒤에 배치하게 된다.

```
>>> pi = list() # pi = [] 도 가능
>>> pi.append('C++')
>>> pi.append('java')
>>> print(pi)
['C++', 'java']
```

리스트의 첨자로 항목 수정

```
>>> top = ['BTS', '봄빨간사춘기', 'BTS', '블랙핑크', '태연', 'BTS']
>>> top[1] = '장범준'
>>> top[3] = '잔나비'
>>> print(top)
['BTS', '장범준', 'BTS', '잔나비', '태연', 'BTS']
>>> top[top.index('태연')] = '노라조'
>>> print(top)
['BTS', '장범준', 'BTS', '잔나비', '노라조', 'BTS']
```

리스트 내부에 다시 리스트를 포함하는 중첩 리스트

```
>>> anima = [['사자', '코끼리', '호랑이'], '조류', '어류']
>>> print(anima)
[['사자', '코끼리', '호랑이'], '조류', '어류']
>>> print(anima[0])
['사자', '코끼리', '호랑이']
```

리스트이 항목 삽입과 삭제

kpop.insert(a, b) # 메소드 insert()로 삽입

=> 항목을 삽입할 첨자 위치 a, 삽입할 항목 b

```
>>> kpop = []
>>> kpop.insert(0, '블랙핑크')
>>> kpop.insert(0, '장범준')
>>> print(kpop)
['장범준', '블랙핑크']
>>> kpop.insert(1, 'BTS')
>>> print(kpop)
['장범준', 'BTS', '블랙핑크']
```

항목을 삭제하는 remove(), del, clear()

kpop.remove(항목) => 해당 항목이 kpop 리스트에서 삭제

del kpop[첨자] => 해당 첨자에 해당하는 항목이 삭제되며 첨자 없이 변수로만 사용시 kpop이 메모리에서 제거된다.

kpop.clear() => kpop이라는 리스트를 빈 리스트로 만든다. 즉 모든 항목을 지운다.

```
>>> print(kpop)
['장범준', 'BTS', '블랙핑크', '잔나비']
>>> kpop.remove('장범준')
>>> print(kpop)
['BTS', '블랙핑크', '잔나비']
>>> print(kpop.pop(1))
블랙핑크
>>> print(kpop)
['BTS', '잔나비']
>>> del kpop[1]
>>> print(kpop)
['BTS']
>>> kpop.clear()
>>> print(kpop)
[]
>>> del kpop
>>> print(kpop)
Traceback (most recent call last):
  File "<pyshell#40>", line 1, in <module>
    print(kpop)
NameError: name 'kpop' is not defined
```

리스트의 추가, 연결과 반복

리스트에 리스트를 추가하는 메소드 extend()

=> 리스트 day.extend(list)는 인자인 list를 가장 뒤에 추가한다.

=> 리스트를 연결하는 +연산자, *연산자

```
>>> day = ['월', '화', '수']
>>> day2 = ['목', '금', '토', '일']
>>> day.extend(day2)
>>> print(day)
['월', '화', '수', '목', '금', '토', '일']
>>> korean = ['불고기', '설렁탕']
>>> chinese = ['탕수육', '기스면']
>>> food = korean + chinese
>>> print(food)
['불고기', '설렁탕', '탕수육', '기스면']
>>> print(day2)
['목', '금', '토', '일']
>>> print(day2 * 2)
['목', '금', '토', '일', '목', '금', '토', '일']
```

리스트 항목의 순서와 정렬

=> reverse() 항목의 순서를 반대로 뒤집는다

=> sorted(list) 항목의 순서를 오름차순으로 반환한다.

```
>>> one = '잣밤배귤감'
>>> wlist = list(one)
>>> print(wlist)
['잣', '밤', '배', '귤', '감']
>>> wlist.reverse()
>>> print(wlist)
['감', '귤', '배', '밤', '잣']
>>> wlist2 = sorted(wlist)
>>> print(wlist2)
['감', '귤', '밤', '배', '잣']
```

리스트 컴프리헨션

=>조건을 만족하는 항목으로 리스트를 간결히 생성하는 컴프리헨션

=> 리스트 컴프리헨션을 사용하면 어떤 조건을 적용한 후 그 반환 값을 항목으로 갖는 다른 리스트로 쉽게 만들 수 있다.

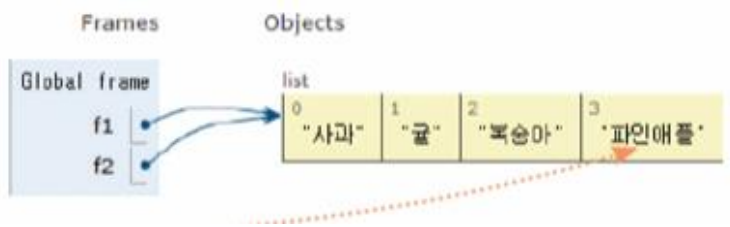
```
>>> even = []
>>> for i in range(2, 11, 2):
        even.append(i)


>>> print(even)
[2, 4, 6, 8, 10]
```

리스트의 대입과 복사

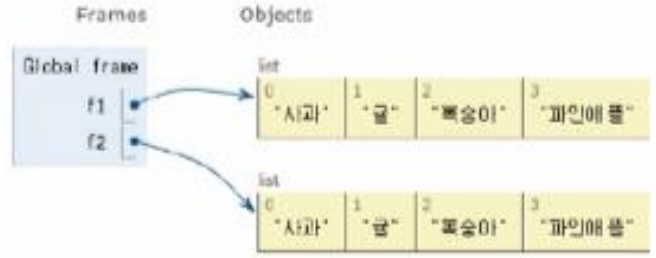
=> 리스트의 대입에는 =은 얕은복사, copy() 깊은 복사로 나뉜다.

```
>>> f1 = ['사과', '귤', '복숭아', '파인애플']
>>> f2 = f1
>>> print(f2)
['사과', '귤', '복숭아', '파인애플']
>>> print(f1)
['사과', '귤', '복숭아', '파인애플']
```



=> f2 = f1 얕은복사는 같은 메모리 공간을 공유한다.

```
>>> f1 = ['사과', '귤', '복숭아', '파인애플']
>>> f3 = f1.copy()
>>> f3.pop()
'파인애플'
>>> print(f1, f3)
['사과', '귤', '복숭아', '파인애플'] ['사과', '귤', '복숭아']
```



=> f3 = f1.copy()는 깊은복사이며 메모리 공간이 따로 있다.

수정할 수 없는 항목의 나열인 튜플

=> 튜플은 리스트와 같은 항목의 나열인 시퀀스이며 리스트와 달리 항목의 순서나 내용의 수정이 불가능하다.
=> empty = tuple(항목1, 항목2, ...)

```
>>> singer = ('BTS', '볼빨간사춘기', 'BTS', '블랙핑크', '태연')
>>> empty = tuple()
>>> print(empty)
()
>>> print(singer)
('BTS', '볼빨간사춘기', 'BTS', '블랙핑크', '태연')
>>> credit = (16, 17)
>>> credit = 16, 17
>>> print(credit) # 튜플은 수정이 불가하다
(16, 17)
```

=> 튜플은 리스트와 마찬가지로 참조, 연결, 반복, 정렬, 삭제 등등 가능하다.

```
>>> print(singer)
('BTS', '볼빨간사춘기', 'BTS', '블랙핑크', '태연')
>>> singer[1:3]
('볼빨간사춘기', 'BTS')
>>> kpop = ('우주소녀', '레드벨벳')
>>> print(singer + kpop)
('BTS', '볼빨간사춘기', 'BTS', '블랙핑크', '태연', '우주소녀', '레드벨벳')
>>> print(kpop * 3)
('우주소녀', '레드벨벳', '우주소녀', '레드벨벳', '우주소녀', '레드벨벳')
>>> print(sorted(singer))
['BTS', 'BTS', '볼빨간사춘기', '블랙핑크', '태연']
>>> del kpop
>>> print(kpop)
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    print(kpop)
NameError: name 'kpop' is not defined
```

나만의 프로젝트 Lab1, 2

1. 스포츠 종목과 팀원 수로 구성된 리스트를 1차원배열, 2차원배열, 컴프리헨션으로 각각출력

```
sport = ['축구', '야구', '농구', '배구']
num = [11, 9, 5, 6]

print(sport)
print (num)

for i in range(len(sport)):
    print("{}: {}명".format(sport[i], num[i]), end = " ")

print()
sponum = [sport, num]
print(sponum)
for i in range(len(sport)):
    print("{}: {}명".format(sponum[0][i], sponum[1][i]), end = " ")

print()
psponum = [[sport[i], num[i]] for i in range(4)]
print(psponum)

for i in psponum:
    print("%s: %d명" % (i[0], i[1]), end = " ")
```

결과

```
['축구', '야구', '농구', '배구']
[11, 9, 5, 6]
축구: 11명 야구: 9명 농구: 5명 배구: 6명
[['축구', '야구', '농구', '배구'], [11, 9, 5, 6]]
축구: 11명 야구: 9명 농구: 5명 배구: 6명
[['축구', 11], ['야구', 9], ['농구', 5], ['배구', 6]]
축구: 11명 야구: 9명 농구: 5명 배구: 6명
```

2. 주문을 위해 콤보의 종류와 수량을 입력받아 주문한 내역과 가격 그리고 총 주문 가격을 출력하는 프로그램 작성 0을 누르면 주문 종료 1~ 4누르면 주문과 그리고 수량 마지막에는 총 주문가격

결과

```
주문할 콤보 번호와 수량을 계속 입력하세요!
        0 주문종료
        1 올인원팩   6000 원
        2 투게더팩   7000 원
        3 트리오팩   8000 원
        4 패밀리팩  10000 원
 >> 2 2
투게더팩, 2개 주문
투게더팩 주문 가격 14000, 총 가격 14000
주문할 콤보 번호와 수량을 계속 입력하세요!
        0 주문종료
        1 올인원팩   6000 원
        2 투게더팩   7000 원
        3 트리오팩   8000 원
        4 패밀리팩  10000 원
 >> 4 2
패밀리팩, 2개 주문
패밀리팩 주문 가격 20000, 총 가격 34000
주문할 콤보 번호와 수량을 계속 입력하세요!
        0 주문종료
        1 올인원팩   6000 원
        2 투게더팩   7000 원
        3 트리오팩   8000 원
        4 패밀리팩  10000 원
 >> 0

****** 주문 종료 *******
총 주문 가격 34000 원
주문을 마치겠습니다
======= 안녕! ========
```

```
menu = ('수분충료', '올인원팩', '투게더팩', '트리오팩', '패밀리팩')
price = (0, 6000, 7000, 8000, 10000)

msg = '주문할 콤보 번호와 수량을 계속 입력하세요!'
for i in range(len(menu)):
    msg += '₩n₩t %d %s' % (i, menu[i])
    if i != 0:
        msg += ' %5d 원' % (price[i])
msg += '₩n >> '

more = True
total = 0
while more:
    instr = input(msg)
    if instr.count(' ') > 0:
        order, cnt = instr.split()
        cnt = int(cnt)
    else:
        order = instr
    order = int(order)
    if order == 0:
        print()
        print(' 주문 종료 '.center(20, '*'))
        more = False
    elif 1 <= order <= 4:
        print('%s, %d개 주문' % (menu[order], cnt))
        sub = price[order] * cnt
        total += sub
        print('%s 주문 가격 %d, 총 가격 %d' % (menu[order], sub, total))
    else:
        print('모르겠어요. 다시 주문하세요!')

else:
    print('총 주문 가격 %d 원' % total)
    print('주문을 마치겠습니다')
    print(' 안녕! '.center(20, '='))
```

Chap06

일상에서 활용되는 문자열과 논리 연산

키와 값인 쌍의 나열인 딕셔너리 dict()

=> 딕셔너리는 말 그대로 사전을 생각하면 된다. 사전은 여러 낱말 각각의 의미, 어원따위를 설명한 내용을 담고 있다. 여기서 낱말은 키, 해설은 값 이라고 생각하면된다.

=> { 키: 값, 키: 값} 또는 { 키= 값, 키 = 값} 으로도 표현 할수 있다.

```
>>> coffeeprice = {'에스프레소': 2500, '아메리카노': 2800
                , '카페라테': 3200}
>>> lect = {} # 또는 lect = dict() 빈 딕셔너리 만들기
>>> day = dict(월='monday', 화='tuesday', 수='wednesday')
>>> print(coffeeprice)
{'에스프레소': 2500, '아메리카노': 2800, '카페라테': 3200}
>>> print(day)
{'월': 'monday', '화': 'tuesday', '수': 'wednesday'}
```

항목의 삽입, 항목의 순회

=> real[키] = 값 형태로 real딕셔너리에 항목을 삽입

=> coffeeprice.keys() 딕셔너리에 있는 키로만 구성된 리스트로 반환

=> coffeeprice.item() 키와 값을 리스트 형태로 반환

=> coffeeprice.values() 값으로 구성된 리스트를 반환

```
>>> real = {3.14: '원주율'}
>>> real[2.71] = '자연수'
>>> real[2.72] = '자연수' # 2.72라는 키가 없으므로 항목을 추가
>>> print(real)
{3.14: '원주율', 2.71: '자연수', 2.72: '자연수'}
>>> print(coffeeprice)
{'에스프레소': 2500, '아메리카노': 2800, '카페라테': 3200}
>>> print(coffeeprice.keys())
dict_keys(['에스프레소', '아메리카노', '카페라테'])
>>> print(coffeeprice.items())
dict_items([('에스프레소', 2500), ('아메리카노', 2800), ('카페라테', 3200)])
>>> print(coffeeprice.values())
dict_values([2500, 2800, 3200])
```

딕셔너리 항목의 참조와 삭제, 결합

=> 딕셔너리 항목의 참조 get()

```
>>> print(city)
{'대한민국': '부산', '뉴질랜드': '웰링톤', '캐나다': '몬트리올'}
>>> city.get('대한민국') # get()메소드를 사용하여 키의 해당 값을 반환
'부산'
>>> city.get('미국', '없네요') # 키의 해당 값이 없을 경우 없네요를 반환
'없네요'
>>> city.get('미국') # 인자가 없을경우 None을 반환
```

=> 딕셔너리를 결합하는 메소드 update()

```
>>> kostock = {'Samsung Elec.': 40000, 'Daum KAKAO': 80000}
>>> usstock = {'MS': 150, 'Apple': 180}
>>> kostock.update(usstock) # update(딕셔너리)는 다른 딕셔너리와 합병
>>> print(kostock)
{'Samsung Elec.': 40000, 'Daum KAKAO': 80000, 'MS': 150, 'Apple': 180}
>>> usstock.update({'MS': 200}) # 동일안 키가 있다면 인자 값으로 대체
>>> print(usstock)
{'MS': 200, 'Apple': 180}
```

=> 딕셔너리의 항목을 삭제하는 메소드 pop(), popitem(), clear(), del

```
>>> print(city)
{'대한민국': '부산', '뉴질랜드': '웰링톤', '캐나다': '몬트리올'}
>>> city.get('대한민국') # get()메소드를 사용하여 키의 해당 값을 반환
'부산'
>>> city.get('미국', '없네요') # 키의 해당 값이 없을 경우 없네요를 반환
'없네요'
>>> city.get('미국') # 인자가 없을경우 None을 반환
>>> print(city)
{'대한민국': '부산', '뉴질랜드': '웰링톤', '캐나다': '몬트리올'}
>>> print(city.pop('뉴질랜드')) # pop(키) 키의 항목을 삭제 그리고 반환
웰링톤
>>> print(city)
{'대한민국': '부산', '캐나다': '몬트리올'}
>>> print(city.popitem()) # popitem()은 임의의 (키, 값)을 삭제
('캐나다', '몬트리올')
>>> print(city)
{'대한민국': '부산'}
>>> city.clear() # 모든 항목을 제거하는 clear()
>>> print(city)
{}
>>> del city # del 변수자체를 메모리상에서 제거한다
>>> print(city)
Traceback (most recent call last):
  File "<pyshell#45>", line 1, in <module>
    print(city)
NameError: name 'city' is not defined
```

중복과 순서가 없는 집합

=> 원소는 중복 될수 없으며 서로 다른 값이여야 한다.

=> 순서가 없다. set()으로도 사용

```
>>> si = {1, 2, 3, 4, 5}
>>> s2 = set([10, 11, 12]) # 리스트형식이나닌 튜플도 가능
>>> print(si, s2)
{1, 2, 3, 4, 5} {10, 11, 12}
```

집합의 원소 추가와 삭제

=>집합의 원소를 추가하는 메소드 add()

```
>>> odd = {1, 2, 3} # 집합을 생성
>>> odd.add(7)
>>> odd.add(9)
>>> print(odd)
{1, 2, 3, 7, 9}
```

=> 집합의 원소를 삭제하는 메소드 remove(), discard(), pop(), clear()

```
>>> print(odd)
{1, 2, 3, 9}
>>> odd.remove(7) # 항목에 없능 원소를 삭제를 하면 오류를 발생
Traceback (most recent call last):
  File "<pyshell#66>", line 1, in <module>
    odd.remove(7) # 항목에 없능 원소를 삭제를 하면 오류를 발생
KeyError: 7
>>> odd.discard(7) # remove()와 같으며 대신 오류가 발생하지 않는다
>>> print(odd.pop()) # 임의원소 삭제
1
>>> print(odd)
{2, 3, 9}
>>> odd.clear() # 모든 항목을 삭제
>>> print(odd)
set()
```

집합의 주요 연산인 합집합, 교집합, 차집합, 여집합

=> 합집합 연산자 |와 메소드 union(), update()

```
>>> a = {4, 6, 8, 10, 12}
>>> b = {3, 6, 9, 12}
>>> a | b
{3, 4, 6, 8, 9, 10, 12}
>>> a.union(b)
{3, 4, 6, 8, 9, 10, 12}
>>> a.update(b) # a를 update() 사용해 합집합 결과를 위해 사용
>>> print(a)
{3, 4, 6, 8, 9, 10, 12}
```

=> 교집합 연산자 & 와 메소드 intersection()

```
>>> a = {4, 6, 8, 10, 12}
>>> b = {3, 6, 9, 12}
>>> a & b
{12, 6}
>>> a.intersection(b) # intersection() 반환이므로 a, b에 영향을 안준다
{12, 6}

>>> a.intersection_update(b)
>>> # a의 값에 a & b를 업데이트한다.
```

=> 차집합 연산자 -와 메소드 difference()

```
>>> a = {4, 6, 8, 10, 12}
>>> b = {3, 6, 9, 12}
>>> a - b # b - a 와 서로 다르다
{8, 10, 4}
>>> a.difference(b)
{8, 10, 4}
>>> b - a
{9, 3}
```

=> 여집합 연산자 ^와 메소드 symmetric_difference()

```
>>> print(a)
{4, 6, 8, 10, 12}
>>> print(b)
{9, 3, 12, 6}
>>> a ^ b
{3, 4, 8, 9, 10}
>>> a.symmetric_difference(b)
{3, 4, 8, 9, 10}
```

내장 함수 zip()과 enumerate() 시퀀스 간의 변환

=> zip()은 동일한 수로 이뤄진 여러 개의 튜플 항목 시퀀스를 각각의 리스트로 묶어 주는 역할을 하는 함수이다.

```
>>> a = ['FTP', 'telnet', 'SMTP', 'DNS']
>>> b = (20, 23, 25, 53)
>>> z = zip(a, b)
>>> type(z)
<class 'zip'>
>>> list(zip(a, b)) # 리스트형태로 항목이 튜플인 리스트 생성
[('FTP', 20), ('telnet', 23), ('SMTP', 25), ('DNS', 53)]
>>> tuple(zip(a, b)) # 항목인 튜플인 튜플 생성
(('FTP', 20), ('telnet', 23), ('SMTP', 25), ('DNS', 53))
>>> dict(zip(a, b)) # 딕셔너리를 생성하는 zip()
{'FTP': 20, 'telnet': 23, 'SMTP': 25, 'DNS': 53}
```

=> 첨자를 자동으로 만들어 첨자와 값과의 쌍인 튜플을 만들어 주는 내장 함수 enimerate()

```
>>> lst = [10, 20, 30]
>>> list(enumerate(lst))
[(0, 10), (1, 20), (2, 30)]
>>> subj = ['국어', '영어', '수학']
>>> for tp in enumerate(subj):
        print('lst[{}]: {}'.format(tp[0], tp[1]))


lst[0]: 국어
lst[1]: 영어
lst[2]: 수학
```

나만의 프로젝트 Lab1, 2 코딩

1. 딕셔너리와 리스트 튜플 등을 활용해 게임은 10회 연속이며 난수를 사용해 가위바위보 중 하나를 선정해 승부를 판정하는 프로그램 작성

```
from random import choice

dcs = {'가위': '보오', '바위': '가위', '보오': '바위'}
tit = ['비김', '철수', '영희', '승지']
rsp = ('가위', '바위', '보오')

print('*' * 17)
print('{:4} {:4} {:4}'.format(tit[1], tit[2], tit[3]))
print('*' * 17)

for _ in range(10):
    cs = choice(rsp)
    yh = choice(rsp)
    print('{:4} {:4}'.format(cs, yh), end = ' ')

    if cs == yh:
        index = 0
    elif dcs[cs] == yh:
        index = 1
    else:
        index = 2
    print('{:4}'.format(tit[index]))
```

결과

```
*****************
철수   영희   승지
*****************
보오   바위   철수
보오   바위   철수
바위   보오   영희
가위   바위   영희
가위   보오   철수
가위   바위   영희
보오   가위   영희
바위   보오   영희
가위   보오   철수
보오   바위   철수
```

2. 딕셔너리로 가상의 kpop 차트를 만들어 출력하는 프로그램 작성 가수의 모음인 리스트와 노래의 모음인 리스트를 만든 후 함수zip()과 enumerate()를 사용

```
singer = ['BTS', '봄빨간사춘기', 'BTS', '블랙핑크', '태연', 'BTS']
song = ['작은 것들을 위한 시', '나만 봄', '소우주', 'Kill This Love', '사계']

kpop = list(zip(singer, song))
print(kpop)

kpchart = dict(enumerate(kpop, start = 1))

print(kpchart)
print()

import pprint
pprint.pprint(kpchart)
```

결과

```
[('BTS', '작은 것들을 위한 시'), ('봄빨간사춘기', '나만 봄'), ('BTS', '소우주'),
('블랙핑크', 'Kill This Love'), ('태연', '사계')]
{1: ('BTS', '작은 것들을 위한 시'), 2: ('봄빨간사춘기', '나만 봄'), 3: ('BTS', '
소우주'), 4: ('블랙핑크', 'Kill This Love'), 5: ('태연', '사계')}

{1: ('BTS', '작은 것들을 위한 시'),
 2: ('봄빨간사춘기', '나만 봄'),
 3: ('BTS', '소우주'),
 4: ('블랙핑크', 'Kill This Love'),
 5: ('태연', '사계')}
```

마무리

이번에 파이썬 포트폴리오 하면서 많은 보람을 느낀것 같았습니다.

50슬라이드 가까이 되는 이 포트폴리오를 처음 만들때는 힘들었지만 어떻게 보면은 다시 공부하는것도 있으면서 정리해보는 시간이 되었습니다.

가뜩이나 여러가지 언어(C언어, java, JSP등등...) 많이 배우다 보니 C언어를 하는대 파이썬처럼 코딩하게 되고 이런 상황이 나오게 되긴했지만 파이썬 포트폴리오를 만들면서 파이썬에 대해다시 정리 하게 되어 도움이 많이 되었습니다.

감사합니다.

컴퓨터정보공학과 20165505 김태민