

7주차

파이썬

- 반복
- 리스트
- 함수
- 출력 형식

		강의 주제	주차
1부 컴퓨팅 사고력	컴퓨팅 사고력의 소개	IT 사회, 소프트웨어 세상, 컴퓨팅 사고력의 소개, 컴퓨팅 사고력의 개념	1
		컴퓨팅 사고력의 개념, 주위에서 볼 수 있는 컴퓨팅 사고력, 문제해결 방법	2
	문제해결 방법, 컴퓨터	문제해결 방법, 문제해결 과정 예, 문제해결을 위한 소프트웨어 설계 사상, 컴퓨터의 특징, 소프트웨어, 유한상태기계	3
2부 소프트웨어	알고리즘	알고리즘 소개, 알고리즘의 표현 방법, 의사코드, 흐름도	4
	프로그램	프로그램의 기능, 함수, 컴파일러	5
	파이썬	파이썬 소개 및 설치, 변수에 값 저장, 입력, 출력, 조건부 수행	6
		반복, 리스트, 함수, 출력 형식	7
3부 컴퓨팅 사고력 활용하기	데이터의 표현	이진수, 아스키코드, 오디오 데이터, 이미지 데이터, 자료구조	8
		인코딩 및 압축, 오류확인	9
	데이터의 저장과 검색	배열 및 연결 리스트, 선형검색, 이분검색, 색인순차검색, 해싱	10
		이진검색트리, 최대값 및 최소값 검색	11
	알고리즘설계	정렬, 분할정복 알고리즘, 탐욕적 알고리즘	12

4. 반복

✓ for

✓ while

```
sum=0
for d in range(10):
    sum = sum + d
print("합=", sum)
```

$d=0,1,2,3,\dots,9$

sum = sum + d의 들여쓰기는 필수

```
sum=0
d=0
while d <10:
    sum = sum + d
    d = d+1
print("합=", sum)
```

$d=0,1,2,3,\dots,9$


sum = sum + d, d=d+1의 들여쓰기는 필수

```
합= 45
>>>
```

- `for d in range(k)`는 d가 0부터 k-1까지 변한다

✓ 들여쓰기(indentation)의 효과


```
sum=0
for d in range(10):
    sum = sum + d
    print("합=", sum)
```



들여쓰기에 의해 매번 print

```
합= 0
합= 1
합= 3
합= 6
합= 10
합= 15
합= 21
합= 28
합= 36
합= 45
>>>
```

```
sum=0
d=0
while d < 10:
    sum = sum + d
    d = d+1
    print("합=", sum)
```



들여쓰기에 의해 매번 print

for

초기값

```
for d in range(10):
```

=

```
for d in range(0, 10):
```

d=0,1,2,3,...,9

```
for d in range(2, 10):
```

d=2,3,...,9

✓ 1부터 10까지 더하기

```
sum=0
for d in range(1,11):
    sum = sum + d
print("합=", sum)
```

range(1,11)은 d=1,2,3,...,10까지 변화시킨다

```
합= 55
>>>
```

✓ 1부터 10까지의 홀수만 더하기

```
sum=0
for d in range(1,11,2):
    sum = sum + d
    print(d)
print("합=", sum)
```

```
1
3
5
7
9
합= 25
>>>
```

- range(1,11,2)은 d=1,3,...,9까지 변화시킨다
- 증가분이 2라는 뜻

```
for d in range(a,b,c):
```

- d는 a의 값에서 시작
- $c > 0$ 일 때 d는 b-1의 값까지 변화한다. $c < 0$ 일 때 d는 b+1의 값까지 변화한다.
- 증분은 c이다.
- a, b, c는 정수이어야 한다.

✓ 증분은 음수가 될 수 있다.

```
sum=0
for d in range(10,1,-2):
    sum = sum + d
    print(d)
print("합=", sum)
```

```
10
8
6
4
2
합= 30
>>>
```

- 초기값 10, d>1인 경우에만 반복 수행



while

✓ 1부터 10까지 더하기

```
sum=0
d=1
while d <11:
    sum = sum + d
    d = d+1
print("합=", sum)
```

- while d<=10: 과 동일 효과

✓ 1부터 10까지의 홀수만 더하기

```
sum=0
d=1
while d <= 10:
    print(d)
    sum = sum + d
    d = d+2
print("합=", sum)
```

- d = d+2는 증가분이 2라는 뜻

```
합= 55
>>>
```

```
1
3
5
7
9
합= 25
>>>
```



✓ 다양한 실수를 이용한 반복 제어 가능

- 1.5부터 3.5보다 작은 동안의 0.1씩 증가하는 실수의 합

```
sum=0
d=1.5
while d < 3.5:
    print(d)
    sum = sum + d
    d = d+0.1
print("합=", sum)
```

```
1.5
1.6
1.700000000000000002
1.800000000000000003
1.900000000000000004
2.000000000000000004
...
...
...
3.000000000000000013
3.100000000000000014
3.200000000000000015
3.300000000000000016
3.400000000000000017
합= 49.0000000000000014
>>>
```

실수를 2진수로 저장함에 따라 발생하는 오차



✓ d의 값을 단순히 증가, 감소가 아닌 것이어도 된다. – 다른 연산에 의한 변경 가능

- 100을 2로 계속 나눌 때의 값 출력
- $d > 1$ 일 때만 반복

```
d=100
while d>1:
    print(d/2)
    d=d/2
```

```
50.0
25.0
12.5
6.25
3.125
1.5625
0.78125
>>>
```



✓ 1,000보다 작은 2의 지수승 값중 가장 큰 수

```
n=2
while n <1000:
    print(n)
    n = n*2
```

- 종료 조건을 알지만, 정확한 수를 알기 어려울 때 사용

```
2
4
8
16
32
64
128
256
512
>>>
```

$\sum_{k=1}^n k \leq 1,000$ 인 최대 n?

```
k=1
sum =0
while sum+k <=1000:
    sum = sum +k
    k = k+1
print("n의 값 = ", k-1)
```

```
n의 값 = 44
>>>
```



[연습문제]

1. 1부터 100까지의 자연수에서 5의 배수들의 합을 계산한다.
2. 100부터 200까지의 자연수 중에서 홀수의 합을 계산한다.

나머지 연산 %

```
>>> 20%2  
0  
>>> 21%2  
1  
>>> 54%5  
4
```

20을 2로 나누었을 때 나머지는 0이다.

21을 2로 나누었을 때 나머지는 1이다.

54를 5로 나누었을 때 나머지는 4이다.



```
sum=0
d=1
while d<=100:
    if d%5==0:
        sum += d
        print(d)
    d +=1
print(sum)
```

sum+=d 와 sum = sum+d 은 동일

```
5
10
15
20
25
30
..
..
85
90
95
100
합 = 1050
>>>
```



5. 리스트



- 여러 개의 데이터들을 저장하는 자료형
- 하나의 이름을 사용
- 이름과 위치를 나타내는 첨자(인덱스)를 결합하여 변수 표현
- 외형은 배열(array)의 형식이지만, 내부적으로는 연결리스트로 구현되어 있다.
- 다른 프로그램의 배열과 연결리스트의 기능을 복합적으로 가짐



- `a=[5,1, 3, 2, 4, 0]`

```
a=[5,1, 3, 2, 4, 0]
print (a)
print(a[0], a[2])
```

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
5	1	3	2	4	0

```
[5, 1, 3, 2, 4, 0]
5 3
>>>
```

- `a=['on', 'before', 'from', 'to']`

a[0]	a[1]	a[2]	a[3]
on	before	from	to

- `a=['on', 56, 'after', 123]`

a[0]	a[1]	a[2]	a[3]
on	56	after	123



a[0] a[1] a[2] a[3] a[4] a[5]

0	0	0	0	0	0
---	---	---	---	---	---

- a=[0] *6 : 6개의 방을 0으로 설정
- a=[0,0,0,0,0,0] 과 동일

```
a=[0]*6  
print(a)
```

```
[0, 0, 0, 0, 0, 0]  
>>>
```

```
a=[0,0,0,0,0,0]  
print(a)
```

```
[0, 0, 0, 0, 0, 0]  
>>>
```



- 연산을 간단히 표시

하나의 변수명에 첨자만 달리하면서 데이터를 접근할 수 있다.

1년의 전체 날 수

```
a=[31,28,31,30,31,30,31,31,30,31,30,31]  
days=0  
for i in range(0,12):  
    days+=a[i]  
print(days)
```

```
365  
>>>
```

`days += a[i]` = `days = days+a[i]`

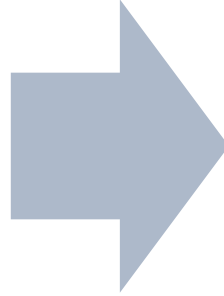
✓ +=, -= , *=, /= 도 동일한 방식

days += 2

days -= 2

days *= 2

days /= 2



days = days + 2

days = days - 2

days = days * 2

days = days / 2



리스트의 구성요소를 직접 반복문 for에 불러올 수 있다.

```
season=['spring', 'summer', 'fall', 'winter']  
for s in season:  
    print(s)
```

```
season=['spring','summer', 'fall', 'winter']  
for k in range(0,4):  
    print(season[k])
```

```
spring  
summer  
fall  
winter  
>>>
```



- 리스트에서 지원하는 함수를 이용하여 다양한 기능을 쉽게 구현할 수 있다.
- 사용형식 - 리스트명.함수명(파라미터) 또는 함수명(리스트명)

- a=[5,1, 3, 2, 4, 0]

a[0] a[1] a[2] a[3] a[4] a[5]

5	1	3	2	4	0
---	---	---	---	---	---

a[1]을 변경



정렬



추가



삭제



```
>>> a=[5, 1, 3, 2, 4, 0]
>>> a[1]=100
>>> a
[5, 100, 3, 2, 4, 0]
>>> a.sort()
>>> a
[0, 2, 3, 4, 5, 100]
>>> a.append(9)
>>> a
[0, 2, 3, 4, 5, 100, 9]
>>> a.remove(9)
>>> a
[0, 2, 3, 4, 5, 100]
>>>
```



- a=[5,1, 3, 2, 4, 0]

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
5	1	3	2	4	0

2의 개수 ----->

역으로 나열 ----->

제일 우측 값 ----->

인덱스1에 있는 데이터 제거 ----->

```
>>> a=[5,1,3,2,4,0]
>>> a.count(2)
1
>>> a.reverse()
>>> a
[0, 4, 2, 3, 1, 5]
>>> a.pop()
5
>>> a
[0, 4, 2, 3, 1]
>>> a.pop(1)
4
>>> a
[0, 2, 3, 1]
```



- 자료구조에 연결하여 함수를 사용하는 것을 객체지향 (object-oriented)이라고 한다.
- 정의된 기능들은 해당 자료구조에서만 사용

다양한 데이터형에 사용할 수 있는 함수

- `a=[5,1, 3, 2, 4, 0]`

<code>a[0]</code>	<code>a[1]</code>	<code>a[2]</code>	<code>a[3]</code>	<code>a[4]</code>	<code>a[5]</code>
5	1	3	2	4	0

리스트 내의 데이터 개수

최대

최소

```
>>> a=[5,1,3,2,4,0]
>>> len(a)
6
>>> max(a)
5
>>> min(a)
0
```

함수명(변수명)



리스트의 함수를 이용한 프로그램 예

- 리스트의 가운데 값 출력하기

```
정렬 -----> a=[4,1,3,9,10]
리스트의 크기 -----> a.sort()
리스트의 가운데 위치 -----> n=len(a)
리스트의 가운데 값 출력 -----> n=int(n/2)
                                print(a.pop(n))
```

```
4
>>>
```

`int(n/2)`: $n/2$ 의 값을 정수값으로 만든다.



행렬의 표현

A=

1	2	3
4	5	6

첫 행 -----> >>> A[0]
[1, 2, 3]
둘째 행 -----> >>> A[1]
[4, 5, 6]
행의 개수 -----> >>> len(A)
2
열의 개수 -----> >>> len(A[0])
3
A를 출력 -----> >>> for r in A:
 print(r)

```
[1, 2, 3]
[4, 5, 6]
>>>
```



A

1	2	3
4	5	6

B

1	2
3	4
5	6

```
A=[[1,2,3],[4,5,6]]
B=[[1,2],[3,4],[5,6]]
mul = [[0,0],[0,0]]
```

```
for i in range(len(A)):
    for j in range(len(B[0])):
        for k in range(len(B)):
            mul[i][j]+=A[i][k]*B[k][j]
for m in mul:
    print(m)
```

A의 행의 개수

B의 열의 개수

B의 행의 개수

```
[22, 28]
[49, 64]
>>>
```



6. 함수(function)

- 반복적으로 나타나는 기능을 독립적으로 구현
- 프로그램이 단순화 되고 가독성 증가

함수정의

```
def add(a):  
    a = a+1  
    return a
```

함수사용

-----> print(add(a))

함수사용

-----> print(add(a))

```
a=3
```

```
print(add(a))
```

```
a=5
```

```
print(add(a))
```

```
4  
6  
>>>
```

✓ 함수사용 이전에 함수가 정의되어야 한다.



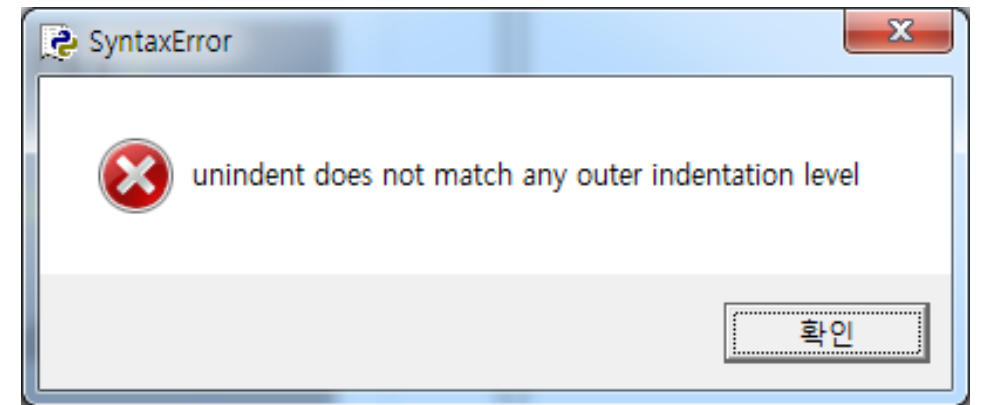
- 함수 내부의 들여쓰기가 일치하여야 한다.

```
def add(a):  
    a = a+1  
    return a  
a=3  
print(add(a))  
a=5  
print(add(a))
```

O

```
def add(a):  
    a = a+1  
    return a  
a=3  
print(add(a))  
a=5  
print(add(a))
```

X



- 함수 내부에도 명령문에 따른 적절한 들여쓰기를 사용하여야 한다.

```
def add(a):  
    if a>1:  
        a = a+1  
    return a  
a=3  
print(add(a))  
a=5  
print(add(a))
```

O



- 주 프로그램의 변수명과 함수내의 변수명

- ✓ 서로 다를 수 있다.
- ✓ 매개변수의 위치에 의한 매핑

```
def add(k):  
    k= k+1  
    return k  
a=3  
print(add(a))
```

```
4  
>>>
```



- 함수 내부에서 변수의 값이 바뀌더라도 외부의 변수값은 바뀌지 않는다.

```
def add(a):  
    a = a+1  
    print("내부", a)  
    return a  
  
a=3  
print(add(a))  
print("외부", a)
```



```
내부 4  
4  
외부 3  
>>>
```



- 함수 내부에 두 개 이상의 변수를 전달할 수 있다.

```
def sum_of_squares(x,y):  
    c = x*x+y*y  
    return c  
a=3  
b=4  
print(sum_of_squares(a,b))
```

```
25  
>>>
```



- 함수 내부에 네 개의 변수를 전달한 예

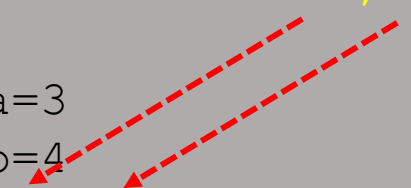
```
def average(a,b,c,d):  
    return (a+b+c+d)/4  
w=3  
x=5  
y=7  
z=10  
print(average(w,x,y,z))
```

```
6.25  
>>>
```



- 함수의 결과로 외부에 두 개 이상의 값을 전달할 수 있다.

```
def comp(x, y):  
    sum = x+y  
    mul = x*y  
    return sum, mul  
  
a=3  
b=4  
c, d = comp(a, b)  
print('Sum is', c)  
print('Multiplication is ', d)
```

A diagram with two red dashed arrows. One arrow starts from the 'sum' variable in the function's return statement and points to the variable 'c' in the function call. The other arrow starts from the 'mul' variable in the function's return statement and points to the variable 'd' in the function call.

```
Sum is 7  
Multiplication is 12  
>>>
```

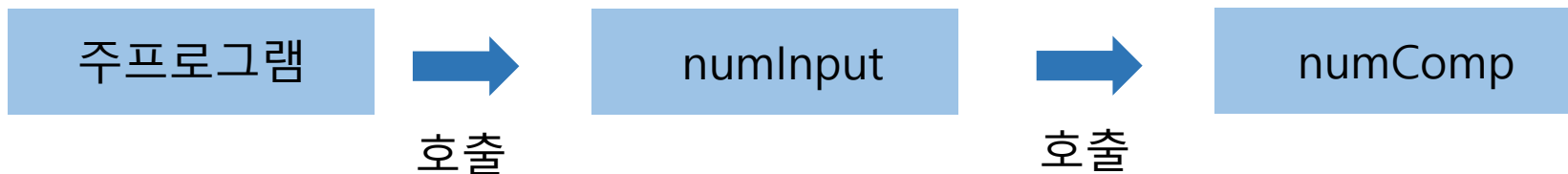


- 함수는 또 다른 함수를 호출할 수 있다.

주프로그램 → numInput → numComp

```
def numComp(a,b,c):  
    a=int(a)  
    b=int(b)  
    c=int(c)  
    return a*a + b*b +c*c  
  
def numInput ( ):  
    a, b, c = input("A = "), input("B = "),input("C = ")  
    print("제공 합 = ", numComp(a,b,c))  
  
numInput()
```

```
A = 4  
B = 5  
C = 6  
제공 합 = 77  
>>>
```



피보나치 수열

- $\text{fib}(n)$ 을 정의하는데, $\text{fib}(n-1)$ 과 $\text{fib}(n-2)$ 를 사용한다
 - ✓ Fibonacci numbers
 - ✓ $\text{fib}(0)=0$, $\text{fib}(1)=1$, $\text{fib}(2)=\text{fib}(1)+\text{fib}(0)=1$
 - ✓ $\text{fib}(3)=\text{fib}(2)+\text{fib}(1)=1+1=2$
 - ✓ 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89,

$$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2), n \geq 2$$



✓ 재귀(recursion) 가능

- 함수가 자신을 호출하는 것
- 피보나치 수열 생성 함수를 구현

```
def fib(n):  
    if n==0:  
        return 0  
    elif n==1:  
        return 1  
    else:  
        return fib(n-1) + fib(n-2)  
  
n=6  
print(fib(n))
```

```
8  
>>>
```

$$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$$

n	0	1	2	3	4	5	6	7
fib(n)	0	1	1	2	3	5	8	13



n!을 재귀를 이용하여 구현

```
def fact(n):  
    if n==1:  
        return 1  
    else:  
        return n * fact(n-1)  
  
n=4  
print(fact(n))
```

```
24  
>>>
```

$$\text{fact}(n) = n \times \text{fact}(n-1)$$

$$n! = n \times (n-1) \times (n-2) \times \dots \times 2 \times 1$$

(n-1)!



- 함수 내부에 리스트를 전달하여 사용할 수 있다.

```
def days(a):  
    d = 0  
    for i in range(0,12):  
        if i%2 == 0:  
            d +=a[i]  
    return d  
a=[31,28,31,30,31,30,31,31,30,31,30,31]  
print(days(a))
```

```
184  
>>>
```

홀수 달의 날수의 총합



7. 출력형식

형식이 있는 output: 1개 이상의 값 또는 변수를 정해진 형식으로 출력

두 개 이상의 변수 또는 값이
있을 경우 괄호 필요

영역구분

```
print('A is %3.1f, B is %3d' % (1.2, 345) )
```

format 문자열

출력할 변수 또는 값

```
A is 1.2, B is 345  
>>>
```

```
a=1.2  
b=345  
print('A is %3.1f, B is %3d' % (a, b) )
```

```
A is 1.2, B is 345  
>>>
```

%3.1f

#

.

#

%7.3f

#

#

#

.

#

#

#

%3d

#

#

#

총길이

소숫점우측길이

정수길이

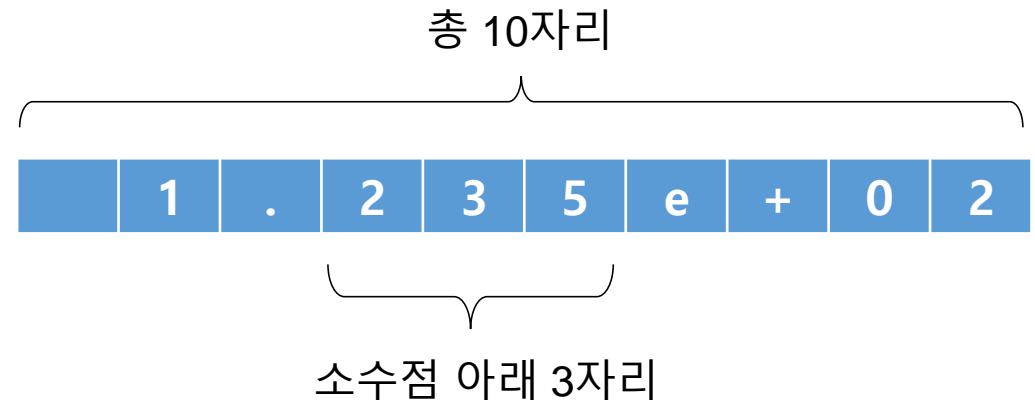



```

>>> a=123.4567
>>> print("%10.3e" % a)
1.235e+02
>>> print('%10.4e' % a)
1.2346e+02
>>> print('%13.4E' % a)
1.2346E+02
>>> print('%10.1f' % a)
123.5
>>> print('%11.5f' % a)
123.45670
>>>

```

%10.3e



%10.4e



%10.4E

대문자 E

- 1.235e+02는 1.235×10^2 을 표시한 것
- 1.235×10^2 은 1.234567×10^2 가 반올림된 것



```

sum=0
d=1.5
while d < 3.5:
    print(d)
    sum = sum + d
    d = d+0.1
print("합=",sum)

```

```

1.5
1.6
1.700000000000000002
1.800000000000000003
...
...
...
3.300000000000000016
3.400000000000000017
합= 49.0000000000000014
>>>

```

출력 형식을 정의한 후

```

sum=0
d=1.5
while d < 3.5:
    print("%6.2f" % d)
    sum = sum + d
    d = d+0.1
print("합= %6.2f" % sum)

```

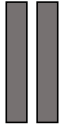
```

1.50
1.60
1.70
1.80
...
...
3.30
3.40
합= 49.00
>>>

```



반복



```
interest = 200*0.1
print('이자율 0.1 이자 %4.1f 원' %
      interest)

interest = 200*0.2
print('이자율 0.2 이자 %4.1f 원' %
      interest)
```

```
이자율 0.1 이자 20.0 원
이자율 0.2 이자 40.0 원
>>>
```

반복

동일한 기능을 수행하는 부분을 함수로 만든다.



```
def f1(interestRate):  
    interest = 200*interestRate  
    print('이자율 %4.2f    이자 %4.1f 원' % (interestRate, interest))  
  
f1(0.1)  
f1(0.2)
```

매개변수 1개

```
def f2(money, interestRate):  
    interest = money*interestRate  
    print('이자율 %4.2f    이자 %4.1f 원' % (interestRate, interest))  
  
f2(200, 0.1)  
f2(200, 0.2)
```

매개변수 2개

```
이자율 0.10    이자 20.0 원  
이자율 0.20    이자 40.0 원  
>>>
```



총금액 출력

```
def f3(money, interestRate):  
    interest = money*interestRate  
    print('총금액 %5d   이자율   %4.2f   이자 %5.1f 원' % (money, interestRate,  
interest))  
  
f3(1200, 0.1)  
f3(3000, 0.2)
```

```
총금액   1200   이자율   0.10   이자  120.0 원  
총금액   3000   이자율   0.20   이자  600.0 원  
>>>
```



[연습문제]

1. 한 학생의 성적은 5 과목의 성적으로 구성되어 있다. 5 과목의 성적이 리스트에 저장되어 있을 때, 성적을 받아서 평균과 최고점수를 알려 주는 함수를 작성하시오.
2. 데이터를 입력 받아 필요한 계산을 수행하는 프로그램을 작성하라. inputData()와 dataProcess() 함수가 필요하다.
 - (1) 주함수는 listA = inputData()를 호출한다.
 - 양의 값을 갖는 데이터를 입력 받아 리스트에 저장한다.
 - 마지막 데이터가 0인 경우 리스트입력 종료.
 - 리스트를 주함수에 전달한다.
 - (2) 주함수는 dataProcess(listA)를 호출한다.
 - 데이터의 개수가 2개인 경우 사각형의 면적, 둘레의 길이를 출력하고,
 - 3개인 경우는 직육면체의 부피를 계산하여 출력한다.





- 애기가 처음 말을 배우는 마음으로 프로그래밍 언어를 공부
- 영어 회화가 이해로 이루어지지 않는다.
- 프로그래밍 언어를 익히는 지름길은 오직 연습
- 다양한 예제 프로그램을 점검
- 스스로 문제를 생성하여 직접 프로그램 해 봐야 한다.
- Do it yourself !



7주차 강의 요약

- 반복
 - ✓ for, while
- 리스트
- 함수
- 출력 형식



7주차 끝

수고하셨습니다.