1월8일실습_1

[실습 1]

1. 파일명 : funcLab8.py
2. 구현해야 하는 함수 사양
   함수명 : **print_triangle_withdeco**
   매개변수 : 2개(**숫자와 데코문자**)
   여기에서 데코문자는 기본값을 갖는다. **기본값은 '%'**로 정한다.
   리턴값 : 없음
   기능 : **전달된 숫자 개수로 구성되는 삼각형을 출력**한다. 출력 형식은 다음과 같다.
   숫자 2 만 전달시
   %
   %%
   숫자 5 와 데코문자 '*' 전달시
   *

   **

   ***

   ****

   *****

전달되는 **아규먼트 값은 1~10으로 제한**한다. **1~10 이외의 값이 전달된 경우에는 처리하지 않는다.**

3. 숫자를 다양하게 지정해서 print_triangle_withdeco () 함수를 호출해 본다.

```
def print_triangle_withdeco(a,b='%'):
    if 1<=a<=10:
        for i in range(1,a+1):
            print(' '*(a-i),b * i)


print_triangle_withdeco(3,'*')
print_triangle_withdeco(5, '^')
print_triangle_withdeco(6)
print_triangle_withdeco(13)
```

[실습 2]

최댓값을 구하는 기능은 함수를 사용하지 않고 제어문으로 직접 구현한다.

1. listLab1.py 이라는 소스를 생성한다.
2. 다음 값들로 구성되는 **리스트**를 생성하여 listnum 에 저장한다.
   10, 5, 7, 21, 4, 8, 18
3. listnum 에 저장된 값들 중에서 **최댓값을 추출**하여 다음과 같이 출력한다.
   최댓값 : 21

```
listnum = [10,5,7,21,4,8,18]
result = listnum[0]
for x in range(1,len(listnum)):
    samp = listnum[x]
    if result < samp:
        result = samp

print("최댓값:",result)
```

[실습 3]

최솟값을 구하는 기능은 함수를 사용하지 않고 제어문으로 직접 구현한다.

1. listLab2.py 이라는 소스를 생성한다.
2. 다음 값들로 구성되는 **리스트**를 생성하여 listnum 에 저장한다.
   10, 5, 7, 21, 4, 8, 18
3. listnum 에 저장된 값들 중에서 **최솟값을 추출**하여 다음과 같이 출력한다.
   최솟값 : 4

```
listnum = [10,5,7,21,4,8,18]
result = listnum[0]
for x in range(1,len(listnum)):
    samp = listnum[x]
    if result > samp:
        result = samp

print("최솟값:",result)
```

[실습 4]

최댓값과 최솟값을 구하는 기능은 함수를 사용하지 않고 제어문으로 직접 구현한다.

1. listLab3.py 이라는 소스를 생성한다.
2. 다음 값들로 구성되는 **리스트**를 생성하여 listnum 에 저장한다.
   10, 5, 7, 21, 4, 8, 18
3. listnum 에 저장된 값들 중에서 **최댓값 최솟값을 추출**하여 다음과 같이 출력한다.
   최솟값 : 4, 최댓값 : 21

```
listnum = [10,5,7,21,4,8,18]
result_b = listnum[0]
result_s = listnum[0]
for x in range(1,len(listnum)):
    samp = listnum[x]
    if result_b < samp:
        result_b = samp
    elif result_s > samp:
        result_s = samp

print("최솟값:",result_s,"최댓값:",result_b)
```

최솟값: 4 최댓값: 21

[실습 5]

1. listLab4.py 이라는 소스를 생성한다.
2. **비어있는 리스트**를 하나 생성하여 listnum 이라는 변수에 저장한다.
3. **1~50 사이의 난수를 10개 추출**하여 listnum 에 추출 순서대로 저장한다. (for문 사용)
4. listnum의 **모든 값들을 출력**한다.(이 때 반복문을 사용하지 않아도 된다.)
5. 리스트에서 **10보다 작은 값들은 100으로 변경**한다. (for문 사용)
6. listnum의 모든 값들을 출력한다.(이 때 반복문을 사용하지 않아도 된다.)
7. **인덱싱 방법**으로 listnum의 첫 번째 데이터를 출력한다.
8. 인덱싱 방법으로 listnum의 마지막 데이터를 출력한다.
9. **슬라이싱 방법**으로 listnum의 두 번째 데이터부터 여섯 번째 데이터만 추출하여 출력한다.
10. 슬라이싱 방법으로 listnum의 데이터를 **역순으로 출력**한다.
11. 슬라이싱 방법으로 listnum의 데이터를 모두 출력한다.
12. 인덱싱 방법으로 5번째 데이터를 **삭제**한다.
13. 슬라이싱 방법으로 listnum의 데이터를 모두 출력한다.
14. 슬라이싱 방법으로 2~3번째 데이터를 삭제한다.
15. 슬라이싱 방법으로 listnum의 데이터를 모두 출력한다.

```
import random
listnum = []
for x in range(10):
    listnum.append(random.randint(1,50))

print(listnum)

for x in range(10):
    if listnum[x] < 10:
        listnum[x] = 100

print(listnum)

print(listnum[0])
print(listnum[-1]) # listnum[len(listnum)-1]
print(listnum[1:6])
print(listnum[::-1])
print(listnum[:])
del listnum[4]
print(listnum[:])
listnum[1:3] = []
print(listnum[:])
```

[실습 6]

1. 파일명 : funcLab9.py
2. 구현해야 하는 함수 사양
   함수명 : sumEven1
   매개변수 : **가변형(전달받을 수 있는 아규먼트 개수에 제한이 없다.)**
   리턴값 : 1개
   기능 : 아규먼트가 몇 개가 전달되든 처리해야 한다.
   아규먼트는 **1 이상의 숫자만** 온다고 정한다.
   전달된 아규먼트들에서 **짝수에 해당하는 숫자들만 합**을 계산해서 리턴한다.
   전달된 아규먼트들 중에 **짝수가 없으면 0을 리턴**한다.
   아규먼트가 전달되지 않으면 -1을 리턴한다.
3. 숫자를 다양하게 지정해서 sumEven1() 함수를 호출해 본다.

```
def sumEven1(*p):
    ans=0
    for data in p:
        if data%2==0:
            ans+= data

    if len(p)==0:
        ans=-1
    return ans

print(sumEven1(1,2,3,4))
print(sumEven1())
print(sumEven1(100))
print(sumEven1(3,5,7))
```

6
-1
100
0

[실습 6]

1. 파일명 : funcLab10.py
2. 구현해야 하는 함수 사양
   함수명 : sumAll
   매개변수 : **가변형(전달받을 수 있는 아규먼트 개수에 제한이 없다.)**
   리턴값 : 1개
   기능 : 아규먼트가 몇 개가 전달되든 처리해야 한다.
   호출시 전달되는 아규먼트의 데이터 타입에는 제한이 없다.
   그러므로 전달된 **아규먼트의 타입을 채크하여 숫자만 처리**하고 숫자가 아닌 데이터는 무시한다.
   아규먼트가 전달되지 않았거나 전달되었다 하더라도 숫자가 없으면 None 을 리턴한다.
3. 숫자를 다양하게 지정해서 sumAll() 함수를 호출해 본다

```
def sumALL(*p):
    count=0
    ans=0
    for i in p:
        if type(i) == int:
            count+=1
            ans+=i
        else: pass
    if len(p)==0 or count ==0 :
        ans = None
    return ans

print(sumALL(1,2,3))
print(sumALL(3,'a', '*'))
print(sumALL())
print(sumALL('a','b', True))
```

6
3
None
None