UNISTUDY

파이썬

16강. 사용자정의 함수

1. 사용자정의함수

- 사용자가 직접 함수 내에 필요한 코드를 작성해 놓고, 외부의 값을 인수로 받아서 처리한 후 처리 결과 를 반환하는 파이썬의 사용자정의함수의 형식은 다음과 같다.
- def 명령어 다음에 함수명을 지정하고, 인수를 받을 매개변수를 괄호 안에 넣는다. 함수의 블록은 콜론과 들여쓰기로 구분한다.
- 함수 내에서 필요한 실행문을 작성하고, 마지막 줄에 외부로 값을 반환하는 return 명령문을 작성한다.

```
def 함수명(매개변수) :
    실행문
    실행문
    return 값
```

1. 사용자정의함수

- 사용자정의함수는 일련의 실행문이 반복될 경우 이를 함수로 정의해 놓고, 필요 시 함수 호출을 통해서 실행이 가능하다.
- 즉 자주 반복되는 실행문을 매번 작성하지 않고 호출을 통해서 실행된다는 이점을 갖는다.

1. 사용자정의함수

chapter05.lecture.step01_func_basic.py

```
# (1) 인수가 없는 함수
def userFunc1():
    print('인수가 없는 함수')
    print('userFunc1')

userFunc1() # 함수 호출

# (2) 인수가 있는 함수

 def userFunc2(x, y):
     print('userFunc2')
     z = x + y
     print('z=', z)

 userFunc2(10, 20)  # 함수 호출
```

Python Console

```
인수가 없는 함수
userFunc1
```

```
userFunc2
z= 30
```

1. 사용자정의함수

```
# (3) return 있는 함수
def userFunc3(x, y):
    print('userFunc3')
    tot = x + y
    sub = x - y
    mul = x * y
    div = x / y

    return tot, sub, mul, div

# 실인수 : 키보드 입력
x = int(input('x 입력 : '))
y = int(input('y 입력 : '))

t, s, m, d = userFunc3(x, y)
print('tot =', t)
print('sub =', s)
print('mul =', m)
print('div =', d)
```

```
x 입력 : >? 10
y 입력 : >? 20

userFunc3
tot = 30
sub = -10
mul = 200
div = 0.5
```

1. 사용자정의함수

(1) 인수가 없는 함수
- userFunc1() 함수는 괄호 안에 매개변수가 없기 때문에 함수를 호출할 경우 인수 없이 빈괄호
- ‘userFunc1()’ 형식으로 호출한다.

(2) 인수가 있는 함수
- userFunc2(x, y) 함수는 괄호 안에 매개변수가 2개 있기 때문에 함수를 호출할 경우 2개의 실인수를 이용하여 ‘userFunc2(10, 20)’ 형식으로 호출한다.

(3) return 있는 함수
- 외부에서 두 개의 인수를 받아서 4개의 실행문으로 사칙연산을 수행하고, 계산결과를 return 명령문 으로 반환하는 함수이다. 실인수12) 두 개는 키보드로 입력받아서 x, y의 매개변수에 순서대로 전달된 다. 그리고 함수에서 반환 값이 있으면 함수를 호출하는 곳으로 반환되기 때문에 함수 호출 문장앞에 대입연산자(=)를 이용하여 4개 변수(t, s, m, d)로 사칙연산 결과를 순서대로 받아준다. 끝으로 받은 결과를 출력하여 사칙연산 결과를 확인한다.

2. 산포도 구하기

- 산포도란 평균으로부터 얼마나 값이 분산되어 있는지의 정도를 나타내는 척도로 분산과 표준편차를 사용한다.
- 특히 산포도는 자료 분석(Data Analysis)을 위한 기술통계량에서 자료의 특성을 파악하는데 중요한 역할을 한다. 먼저 분산을 구하는 수식을 알아보고, 이어서 표준편차를 알아보자. 분산의 수식 은 다음과 같다.

$$\sigma^2 = \frac{\sum(x_i - \mu)^2}{n}$$

<모 분산>

$$s^2 = \frac{\sum(x_i - \bar{x})^2}{n-1}$$

<표본 분산>

2. 산포도 구하기

- <모 분산>은 모집단을 대상으로 한 분산이고, <표본 분산> 모집단에서 임의 추출한 표본에 대한 분산이다. 표준편차는 분산에 양의 제곱근을 적용하면 된다.
- 다음 예문에서 사용자 정의 함수를 이용하여 <표본 분산>과 <표본 표준편차>를 구하는 방법에 대해서 알아본다.

2. 산포도 구하기

chapter05.lecture.step02_func_app.py

```
from statistics import mean, variance
from math import sqrt

dataset = [2, 4, 5, 6, 1, 8]

# (1) 산술평균
def Avg(data):
    avg = mean(data)
    return avg

print('산술평균 =', Avg(dataset))


# (2) 분산/표준편차
def var_sd(data):
    avg = Avg(data) # 함수 호출
```

Python Console

```
산술평균 = 3
```

2. 산포도 구하기

```
    # list 내포
    diff = [ (d - avg)**2  for d in data]

    var = sum(diff) / (len(data) - 1)
    sd = sqrt(var)

    return var, sd


# (3) 함수 호출
v, s = var_sd(dataset)
print('분산 =', v)
print('표준편차=', s)
```

```
분산 = 2.5
표준편차= 1.5811388300841898
```

2. 산포도 구하기

- 해설 분산과 표준편차 함수 예

(1) 산술평균

- Avg() 함수는 6개의 원소를 갖는 dataset 변수의 평균을 계산하는 함수이다. 함수의 반환 값은 표본의 분산식에서 산술평균($\bar{x}$)으로 사용된다. mean() 함수는 statistics 모듈에서 제공되는 함수이다.

(2) 분산/표준편차

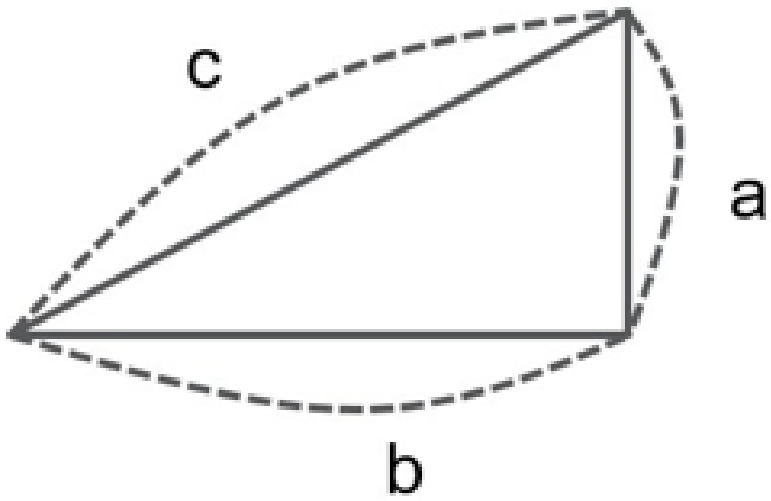
- var_sd() 함수는 Avg() 함수를 호출하여 산술평균을 반환 받고, 리스트 내포 명령문에 의해서 각 변량과 산술평균을 1:1로 차를 계산하여 diff 변수에 할당한다. 이어서 diff 변수의 합을 계산하고, 전체 변량의 길이에서 1를 뺀 값으로 나누면 표본의 분산이 계산된다. 표준편차는 분산의 결과에 양의 제곱근(sqrt)을 적용한다. sqrt 함수는 math 모듈에서 제공되는 함수이다.

(3) 함수 호출

- dataset 변수를 인수로 함수를 호출하면 분산(var)과 표준편차(sd)를 반환된다. 이를 v와 s 변수로 순서대로 받아서 계산된 분산과 표준편차를 출력한다.

3. 피타고라스 정리

- 직각삼각형에 관한 피타고라스 정리(Pythagorean theorem)는 아래 그림과 같은 직각삼각형의 세 변의 길이에서 a2+b2=c2 관계 가 성립한다는 이론이다.



3. 피타고라스 정리

- 피타고라스 정의 함수 예
- 피타고라스의 식[(빗변)의 제곱 = (밑변)의 제곱+(높이)의 제곱]에 함수의 결과(a, b, c)를 적용하 면 32 + 42 = 52과 같다.

chapter05.lecture.step02_func_app.py

```
# 피타고라스 정리
def pytha(s, t) :
    a = s**2 - t**2
    b = 2 * s * t
    c = s**2 + t**2
    print("3변의 길이 : ",a,b,c)

pytha(2, 1) # s, t의 인수는 양의 정수를
갖는다.
```

Python Console

```
3변의 길이 :  3 4 5
```

4. 몬테카를로 시뮬레이션

- 현실적으로 불가능한 문제의 해답을 얻기 위해서 난수의 확률분포를 이용하여 모의실험으로 근사적 해를 구하는 기법을 의미한다.
- 예를 들면 동전을 1,000번 던져서 앞면이 나올 확률이나 뒷면이 나올 확 률 구하기 위해서 실제 동전을 1,000번 던질 수는 없다.
- 이러한 문제를 해결하기 위해서 컴퓨터와 같 은 도구를 이용한 가상 모의실험을 몬테카를로 시뮬레이션(Monte-carlo simulation)이라고 한다.
- 다음 예문은 동전 앞면과 뒷면의 난수 확률분포의 기대확률 모의실험 한 예문이다.

4. 몬테카를로 시뮬레이션

chapter05.lecture.step02_func_app.py | Python Console

```
# 단계 1 : 동전 앞면과 뒷면의 난수 확률분포 함수 정의
def coin(n) :
    result = []
    for i in range(n) :
        r = random.randint(0, 1)
        if (r == 1) :
            result.append(1) # 앞면
        else :
            result.append(0) # 뒷면
```

4. 몬테카를로 시뮬레이션

```
    return result

print(coin(10))                                  [1, 0, 0, 1, 0, 1, 0, 1, 0, 0]


# 단계 2 : 몬테카를로 시뮬레이션 함수 정의
def montaCoin(n) :
    cnt = 0
    for i in range(n) :
        cnt += coin(1)[0] # coin 함수 호출

    result = cnt / n  # 누적 결과를 시행
횟수(n)로 나눈다.
    return result

#단계 3 : 몬테카를로 시뮬레이션 함수 호출
print(montaCoin(10))
print(montaCoin(30))                             0.4
print(montaCoin(100))                            0.5666666666666667
print(montaCoin(1000))                           0.36
print(montaCoin(10000))                          0.501
                                                 0.5019
```

4. 몬테카를로 시뮬레이션

(1) 단계 1 : 동전 앞면과 뒷면의 난수 확률분포 함수 정의
- randint 클래스에 의해서 생성된 난수 정수 0 또는 1을 대상으로 1이면 동전의 앞면, 0이면 동전의 뒷면으로 가정하여 시행 횟수 n 만큼 반복하여 난수의 확률분포를 result 변수에 추가한다.

(2) 단계 2 : 몬테카를로 시뮬레이션 함수 정의
- 시행횟수 n 만큼 coin 함수를 1회 호출하여 반환된 값을 cnt 변수에 누적하고, 누적 결과를 시행 횟수 n으로 나누어서 result 변수에 할당하고, 이를 반환하는 함수이다.

(3) 단계 3 : 몬테카를로 시뮬레이션 함수 호출
- 동전을 던진 경우 나올 수 있는 기대 확률은 앞면과 뒷면 각각 0.5(1/2)에 해당된다. 하지만 일정한 시행 횟수이하인 경우에는 기대확률이 나타나지 않지만 시행 횟수를 무수히 반복할 경우에는 동전의 앞면과 뒷면의 기대 확률은 0.5에 가까워진다. 즉 시행 횟수(표본 수)가 무한이 클수록 동전의 앞면과 뒷면이 나올 기대확률(0.5)에 근사적 해가 구해지는 것을 몬테카를로 시뮬레이션 함수를 통해서 확인할 수 있다.

THANK YOU