

Professional Coding Specialist

COS Pro 파이썬 1 급

2 강. 문법 정리 2

1. 반복문과 함수
 2. 리스트
-

과정 소개

COS Pro 1 급 파이썬 시험 대비를 위한 문법 정리 두 번째 시간으로 파이썬에서 반복문을 위해 사용하는 for 문과 while 에 대해서 정리한 후 파이썬 함수를 정의하고 사용하는 방법을 소개한다. 그리고 파이썬 리스트에 리스트 메소드와 파이썬 내장함수를 적용하여 원하는 값을 산출하는 방법을 알아보고, 파이썬 특유의 표현법인 리스트 컴프리헨션을 활용하여 원하는 리스트를 생성하는 방법에 대해 정리해 보고자 한다.

학습 목차

1. 반복문과 함수
2. 리스트

학습 목표

1. 파이썬 반복문 구성을 위해 for 문과 while 문의 특징과 사용법을 이해하여 프로그램을 작성할 때 활용할 수 있다.
2. 반복문에서 사용하는 키워드인 continue / break/ else 의 기능을 이해하고 이를 활용할 수 있다.
3. 파이썬 리스트를 생성하고, 리스트 메소드와 파이썬 내장함수를 사용하여 리스트에서 원하는 값을 산출할 수 있다.
4. 리스트 컴프리헨션(표현식)을 활용하여 원하는 항목을 갖는 리스트를 생성할 수 있다.

1

반복문과 함수

1. 반복문 for

1) 반복 범위가 숫자인 경우

```
for i in range(start, stop, step):
```

- start 에서 시작하여 step 만큼 증가시켜 stop-1 까지의 값을 i에 할당하여 반복.

예 1)

```
for i in range(5):  
    print(i)
```

: 0 부터 1 만큼 증가시켜 4 까지의 수를 i 에 할당.

0
1
2
3
4

예 2)

```
for i in range(1, 10, 2):  
    print(i)
```

: 1 부터 2 만큼 증가시켜 9 까지의 수를 가져와 i에 할당.

1
3
5
7
9

2) 반복 범위가 문자인 경우

```
for x in 'abcde':  
    print(x)
```

: 문자열에 있는 문자를 하나씩 가져와 x 에 할당.

a
b
c
d
e

3) 반복 범위가 리스트인 경우

```
for x in ['금요일', '토요일', '일요일']:  
    print(x)
```

: 리스트에 있는 항목을 하나씩 가져와 x 에 할당.

금요일
토요일
일요일

2. 반복문 while

```
while 조건식:  
    실행문
```

- 조건식을 만족하는 동안 실행문으로 작성된 명령을 수행.

예)


```
while True:
    a=input("끝내려면 y, 아니면 n을 입력하시오")
    if a=='y':
        print("종료합니다")
        break
```

3. 반복문에서 사용하는 continue / break/ else

1) continue

반복문의 처음으로 돌아가서 다시 실행.

```
for i in range(11):
    if i%2 ==1:
        continue
    print(i)
```




0
2
4
6
8
10

2) break

반복문을 강제 종료한 후 반복문의 영역을 빠져나감.

```
for i in range(11):
    if i%2 == 1:
        break
    print(i)
```




0

3) else

반복문이 정상 종료되었을 때 실행하는 명령을 지정.

```
for i in range(11):
    if i%2 ==1:
        continue
    print(i)
else:
    print('for 문 종료')
```



0
2
4
6
8
10
for 문 종료

```
i=1
while i<=5:
    print(i)
    i+=1
else:
    print('while문 종료')
```



1
2
3
4
5
while문 종료

4. 함수

1) 함수 정의

```
def 함수명([매개변수1, 매개변수2, ...]):
    실행문1
    실행문2
    ...
    [return 결과값1, 결과값2, ...]
```

예) add() 함수 정의하여 사용하기

```
def add(x, y):
    r=x+y
    q=r/2
    return r, q

a,b = add(25, 30)
print(a, b)
```

→ 55 27.5

※ a, b = b, a : 두 변수 a, b 의 값을 맞교환

2) 지역 변수 vs. 전역 변수

- 지역 변수 : 변수의 영역이 함수 안으로 제한되는 변수.
- 전역 변수 : 함수 밖에서 생성된 변수로 함수 안/밖에서 사용하여 그 값을 공유하는 변수
- 함수의 매개변수는 지역 변수.

```
x = 15
y = 100

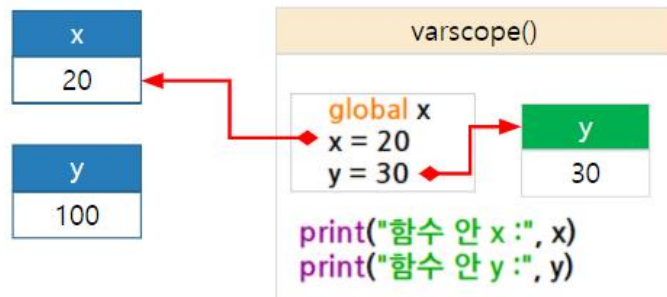
def varscope():
    global x
    x = 20
    y = 30
    print("함수 안 x:", x)
    print("함수 안 y:", y)

varscope()
print("함수 밖-메인 x:", x)
print("함수 밖-메인 y:", y)
```

- 전역 변수 x, y 생성.
- varscope() 내에서 전역 변수 x 사용을 선언.
- 전역 변수 x 에 20 저장.
- 지역 변수 y 를 생성하여 30 저장.
- 함수가 종료되면 지역 변수 y 도 소멸.

<결과>

```
함수 안 x: 20
함수 안 y: 30
함수 밖-메인 x: 20
함수 밖-메인 y: 100
```



2

리스트

1. 리스트

1) 리스트 생성

리스트 변수명 = [값 1, 값 2, 값 3, ...]

예시)

```
a = [1, 2, 3, 4]
a = ['one', 'two', 'three', 'four']
a = [1, 2, 'three', 'four']
a = [1, 2, ['one', 'two', 'three', 'four']]
a = [ ]
a = list( )
```

➔ 리스트 항목을 가리키는 인덱스는 0 부터 시작.

2) 리스트 활용하기

① 리스트 항목 출력

```
x = [1, 2, 3, 4]
print(x)
print(x[0], x[2])
print(x[1:3])
```

- 리스트 x 전체 출력
- 리스트 x 의 인덱스 0 항목, 인덱스 2 항목 출력.
- 리스트 x 의 인덱스 1 항목부터 인덱스 2 항목까지 출력.

➔
[1, 2, 3, 4]
1 3
[2, 3]

② 리스트 항목 변경

```
y=["사과", "바나나"]
z=y*2
print(z)
y[0] = "복숭아"
print(y)
a = x+y
print(a)
```

- y 리스트를 두 번 반복한 것을 z에 저장.
- 리스트 y 의 인덱스 0 항목의 값을 '복숭아' 로 변경.
- 리스트 x와 리스트 y 를 병합한 것을 a에 저장.

➔
['사과', '바나나', '사과', '바나나']
['복숭아', '바나나']
[1, 2, 3, 4, '복숭아', '바나나']

③ if 문과 for 문에서 리스트 활용

```
a=['orange','apple','banana']
if 'apple' in a:
    print('사과가 있습니다')
for k in a:
    print(k)
```



사과가 있습니다
orange
apple
banana

- 리스트 a 에 'apple' 이 있으면 print() 함수 실행.
- for 문을 이용하여 리스트 a 의 항목들을 하나씩 출력.

3) 리스트 주소 복사, 값 복사

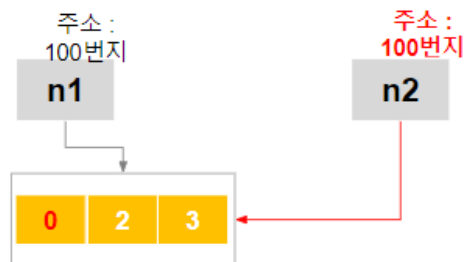
① 주소 복사

```
n1=[1,2,3]
n2=n1
n2[0]=0
print(n1, n2)
```

- [1,2,3] 이 위치한 주소를 n1 에 할당.
- n1 이 가지고 있는 주소를 n2 에 저장.
- n2 의 0 번째 항목을 0 으로 변경.

<결과>

[0, 2, 3] [0, 2, 3]



② 값 복사

```
n1=[1,2,3]
n2=n1.copy()
n2[0]=0
print(n1, n2)
```

- [1,2,3] 이 위치한 주소를 n1 에 할당.
- n1 이 가리키는 리스트를 복사하여 n2 에 그 주소를 저장.
- n2 의 0 번째 항목을 0 으로 변경.

<결과>

[1, 2, 3] [0, 2, 3]



2. 리스트 메소드

1) 리스트에 항목을 추가하는 메소드

메소드 명	기능
append(값)	해당 값을 리스트의 끝에 추가
insert(인덱스, 값)	해당 인덱스 번호에 해당하는 위치에 값을 추가
extend(리스트)	리스트 뒤에 인수로 전달된 리스트를 붙임

예)

```

a=list()
a.append(10)
print(a)
a.insert(1, 30)
print(a)
a.extend([50, 60])
print(a)

```



```

[10]
[10, 30]
[10, 30, 50, 60]

```

2) 기타 리스트 메소드

메소드 명	기능
index(값)	값에 해당하는 인덱스 번호를 리턴
count(값)	해당 값의 개수를 리턴
pop(인덱스)	해당 인덱스의 데이터를 삭제
remove(값)	해당 값을 삭제
clear()	리스트 전체를 삭제

예)

```

a=[10, 30, 50, 60]
print(a.index(30))
print(a.count(10))
a.remove(10)
print(a)
a.pop(2)
print(a)
a.clear()
print(a)

```



```

1
1
[30, 50, 60]
[30, 50]
[]

```

3. 리스트를 활용하는 파이썬 내장함수

1) 리스트 항목들을 집계하는 함수

함수명	기능
min(리스트)	리스트 항목 값 중 가장 작은 값을 리턴

max(리스트)	리스트 항목 값 중 가장 큰 값을 리턴
sum(리스트)	리스트 항목 값의 합을 리턴
len(리스트)	리스트 항목의 개수를 리턴

예)

```
a=[10, 20, 30]
print(min(a), max(a), sum(a), len(a))
```

→ 10 30 60 3

2) 리스트 항목에 일괄 적용할 함수를 지정하는 함수

함수명	기능
map(함수, 리스트)	<ul style="list-style-type: none"> - 리스트 개개의 항목에 함수를 일괄 적용시켜 그 결과를 돌려주는 함수. - 돌려받는 값이 리스트가 아니므로 리스트로 변환시켜야 함.

예)

```
b = ['11', '12', '13']
b2=map(int,b)
print(b2)
b2=list(b2)
print(b2)
```

→ <map object at 0x00000207925ED4F0>
[11, 12, 13]

4. 리스트 컴프리헨션(표현식)

1) 값을 이용한 표현식

```
[값 for 변수 in range(반복 범위) ]
list(값 for 변수 in range(반복
```

→ 반복 범위의 결과만큼 중복된 값을 갖는 리스트가 생성됨.

예)

```
a = [0 for _ in range(5)]
print(a)
a=[0 for i in range(5)]
print(a)
```

<결과>
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]

→ range(5) 에 의해 5 개의 0 을 갖는 리스트를 생성하여 a 에 할당.

2) 식을 이용한 표현식

```
[식 for 변수 in range(반복 범위) ]
list(식 for 변수 in range(반복
```

→ 반복 범위만큼 식을 계산한 결과 값을 갖는 리스트가 생성됨.

예)

```
a=[i for i in range(5)]
print(a)

a=[str(i+10) for i in range(5)]
print(a)

a=list(i for i in range(3,10,2))
print(a)
```

range(5) 에 의해 0 부터 4 까지의 값을 변수 i 로 받아와 i 를 항목으로 하는 리스트를 생성

range(5) 에 의해 0 부터 4 까지의 값을 변수 i 로 받아온 후 str(i+10)의 결과를 항목으로 하는 리스트를 생성

range(3, 10, 2)에 의해 3 부터 9 까지 2 만큼 증가하는 값을 i 로 받아와 i 를 항목으로 하는 리스트 생성.

<결과>

```
[0, 1, 2, 3, 4]
['10', '11', '12', '13', '14']
[3, 5, 7, 9]
```

3) 조건식이 추가된 표현식

[변수 for 변수 in range(반복 범위) if 조건식]
list(변수 for 변수 in range(반복 범위) if

→ 리스트를 생성할 때, 조건식을 만족하는 값만 항목으로 추가하여 생성.

예)

```
a=[i for i in range(10) if i%3==0]
b=list(i for i in range(10) if i%3==0)
print(a)
print(b)
```

range(10) 에서 가져온 값 중 3 으로 나눈 나머지가 0 인 항목만 추가하여 리스트로 생성.

<결과>

```
[0, 3, 6, 9]
[0, 3, 6, 9]
```

4) 두 개의 for 문을 사용하는 표현식

리스트 a 에 있는 항목과 리스트 b 에 있는 항목을 결합한 값을 항목으로 하는 리스트 생성의 예

```
a=['즐거운', '행복한']
b=['파이썬', '나']

c=[]
for i in a:
    for j in b:
        c.append(i+j)
print(c)

d=[i+j for i in a for j in b]
print(d)
```

중첩 for 문을 이용하여 리스트 a 의 항목과 리스트 b 의 항목을 결합한 값을 리스트 c 에 추가.

연달아 사용한 for 문으로 리스트 a, 리스트 b 의 항목을 하나씩 가져와 리스트를 생성.

<결과>

```
{'즐거운파이썬', '즐거운나', '행복한파이썬', '행복한나'}
```