UNISTUDY

파이썬

10강. 제어문(반복문2)

1. for문

- for 명령어를 이용한 반복문은 별도의 조건식이 없는 대신에 열거형객체를 이용한다. 열거형객체란 하나의 메모리 영역에 여러 개의 자료가 나열된 객체를 의미한다.
- 다시 말해서 하나의 참조변수를 이용 해서 여러 개의 자료를 참조할 수 있는 객체를 말한다.
- 문자열은 일정한 순서를 갖는 여러 개의 문자로 나열되어 있으며, 참조변수의 색인을 이용해서 각 문자로 접근할 수 있다.
- for문의 in 뒤쪽에는 여러 개의 값을 가지고 있는 열거형객체를 지정하고, in 앞쪽에는 열거형객체의 값을 하나씩 넘겨받는 변수를 지정한다. 한편 for 반복문에서 반복 횟수는 열 거형객체에 포함된 원소길이에 의해서 결정된다.

```
for 변수 in 열거형객체 :
        실행문1
        :
        실행문n
```

1. for문

chapter03.lecture.step03_for.py

```
# (1) 문자열 열거형객체 이용

string = "홍길동"
print(len(string)) # 문자 길이 : 3
for s in string : # 1문자 -> 변수 넘김 : 3회
    print(s)


# (2) list 열거형객체 이용
lstset = [1, 2, 3, 4, 5] # 5개 원소를 갖는 열거형객체

for e in lstset :
    print('원소 : ', e)
```

Python Console

```
3
홍
길
동

원소 :  1
원소 :  2
원소 :  3
원소 :  4
원소 :  5
```

1. for문

- (1) 문자열 열거형객체 이용
- string 변수는 3개의 문자로 구성된 열거형객체이다. for문에서 string을 열거형객체로 사용하면 s 변수에 단일 문자가 1개씩 할당되면서 3회 반복된다. 반복문의 수행 결과는 단일 문자가 줄 단위로 출력된다.
- (2) list 열거형객체 이용
- lstset 변수는 1에서 5까지 5개의 원소를 갖는 열거형객체이다. 5개의 원소가 순서대로 e 변수에 넘어오면서 5회 반복되면서 원소가 출력된다.

2. for & range

- 파이썬의 모듈 파일(*.py)은 여러 가지 함수를 제공하기도 하지만 다양한 클래스를 제공하기도 한다.
- 여기서는 모듈에서 제공되는 range 클래스 사용법에 초점을 두고 실습을 진행한다. 클래스는 함수와 모양이 매우 비슷하다.
- range 클래스의 도움말은 프롬프트(>>>) 다음에 help(range) 명령문을 이용하여 확인할 수 있다.

2. for & range

- 가장 첫 번째 'class range in module builtins' 문장에서 range 클래스는 builtins 모듈에 포함된 클래스임을 나타내고 있다. 다음 문장부터는 클래스의 인수에 대한 설명이다. 인수가 한 개인 경우, 인 수가 두 개인 경우 range 객체(object)가 만들어지는 사례를 설명하고 있다.

```
>>>help(range)
Help on class range in module builtins:

class range(object)
 |  range(stop) -> range object
 |  range(start, stop[, step]) -> range object
 |
 |  Return an object that produces a sequence of integers from start (inclusive)
 |  to stop (exclusive) by step.  range(i, j) produces i, i+1, i+2, ..., j-1.
 |  start defaults to 0, and stop is omitted!  range(4) produces 0, 1, 2, 3.
 |  These are exactly the valid indices for a list of 4 elements.
 |  When step is given, it specifies the increment (or decrement).
```

2. for & range

chapter03.lecture.step03_for.py

```
# (1) range 객체 생성
num1 = range(10) # range(start)
print('num1 : ', num1)
num2 = range(1, 10) # range(start, stop)
print('num2 : ', num2)
num3 = range(1, 10, 2) # range(start, stop,
step)
print('num3 :', num3)

# (2) range 객체 활용
for n in num1 :
    print(n, end = ' ')
print()
for n in num2 :
    print(n, end = ' ')
print()
for n in num3 :
    print(n, end = ' ')
```

Python Console

```
num1 :  range(0, 10)

num2 :  range(1, 10)

num3 : range(1, 10, 2)


0 1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

1 3 5 7 9
```

2. for & range

- range 클래스 예
- (1) range 객체 생성
- range 클래스를 이용하여 3가지 유형의 객체를 생성하고 있다. num1 변수는 start 인수를 이용하여 0~9까지 숫자 객체가 만들어지고, num2 변수는 start, stop 인수를 이용하여 1~9까지 숫자 객체가 만들어진다. 끝으로 3개의 인수를 갖는 num3 변수는 step의 인수에 의해서 2씩 증가하는 숫자 객체가 만들어진다..
- (2) range 객체 활용
- range 클래스에 의해서 생성된 객체는 열거형객체이다. 따라서 for의 in 뒤에 열거형객체로 이용할 수 있다. for문에서 사용되는 print 실행문에 의해서 num1, num2, num3 객체들의 값을 확인할 수 있다.

3. for & list

- 배열(Array)은 자료를 순차적으로 저장하는 자료구조를 말한다. 자료구조에 관한 자세한 설명은 제4장 에서 다룬다. 여기서는 대괄호([])에 의해서 생성된 list 자료구조에 초점을 두고 실습을 진행한다.
- list 자료구조는 배열의 자료구조를 갖는다. 따라서 순차적으로 자료를 저장하고, 저장된 자료를 색인으로 참조할 수 있다.

3. for & list

chapter03.lecture.step03_for.py

```
# (1) list에 자료 저장하기
lst = [] # 빈 list 만들기
for i in range(10) : # 0~9
    r = random.randint(1,10) # 난수 발생
    lst.append(r) # 난수 저장

print('lst=',lst) # 난수 출력

# (2) list에 자료 참조하기
for i in range(10) : # 0~9
    print(lst[i] * 0.25) # 난수 * 0.25
```

Python Console

```
lst=[2, 4, 5, 4, 3, 9, 1, 10,
9, 5]

0.5
1.0
1.25
1.0
0.75
2.25
0.25
2.5
2.25
1.25
```

3. for & list

- list 자료구조 예
- (1) list에 자료 저장하기
- range(10)에 의해서 0~9까지 10회 반복되면서 난수 정수 10개 생성된다. 이렇게 생성된 난수는 순차적으로 lst 변수에 저장된다. list 자료구조는 자체 제공되는 append() 함수에 의해서 첫 번째 위치부터 자료가 저장된다. lst 변수의 첫 번째 위치는 lst[0]와 같이 색인으로 표현한다.
- (2) range 객체 활용
- lst 변수는 현재 10개의 난수 정수를 저장하고 있다. 순차적으로 저장되기 때문에 색인을 이용하여 순차적으로 배열의 원소를 참조할 수 있다. 위 예문은 range(10)에 의해서 0~9까지 색인을 순서대로 발생시켜 10개의 난수에 각각 0.25를 곱해서 출력한 결과이다.

4. 중첩 반복문

- 중첩 반복문(이중 반복문)은 반복문 안에 또 다른 반복문이 포함된 명령문을 말한다.
- for와 while 명령어를 이용한 중첩 반복문의 형식이다.

4. 중첩 반복문

- 중첩 반복문은 for와 while 명령어 모두 사용이 가능하다. 바깥쪽 반복문과 안쪽 반복문의 영역은 들여쓰기에 의해서 구분한다. 이중 반복문의 동작 과정을 단계적으로 살펴보면 다음과 같다.

- 단계1 : 바깥쪽 반복문의 조건이 참(True)이면 1회 수행한 후 안쪽 반복문으로 진입한다. 단계2 : 안쪽 반복문의 조건이 참(True)인 동안 반복한다.
- 단계3 : 안쪽 반복문의 조건이 거짓(False)이면 안쪽 반복문을 탈출하고, 바깥쪽 반복문의 시작으로 이동하여 단계1~단계3을 반복 수행한다.

- 만약 단계1에서 바깥쪽 반복문의 조건이 거짓(False)이면 중첩 반복문을 완전히 탈출한다.

4. 중첩 반복문

- (1) 구구단
- 구구단은 단(2~9)과 곱해지는 수(1~9)에 의해서 계산된다. 먼저 2단은 2를 1~9까지 순서대로 곱해서 만들어진다. 이러한 원리는 중첩 반복문의 수행과정과 일치한다. 따라서 단은 바깥쪽 반복문에서 만들 고, 곱해지는 수는 안쪽 반복문에서 만든 다음 단과 곱해지는 수를 곱해서 구구단을 만들 수 있다.

4. 중첩 반복문

chapter03.lecture.step03_for.py

```
# 구구단 출력 : range() 함수 이용

# (1) 바깥쪽 반복문
for i in range(2, 10) :
    print('~~~ {}단 ~~~'.format(i))

    # (2) 안쪽 반복문
    for j in range(1, 10) :

        print('%d * %d = %d'%(i, j, i*j))
```

Python Console

```
~~~ 2단 ~~~
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
생략
 :
```

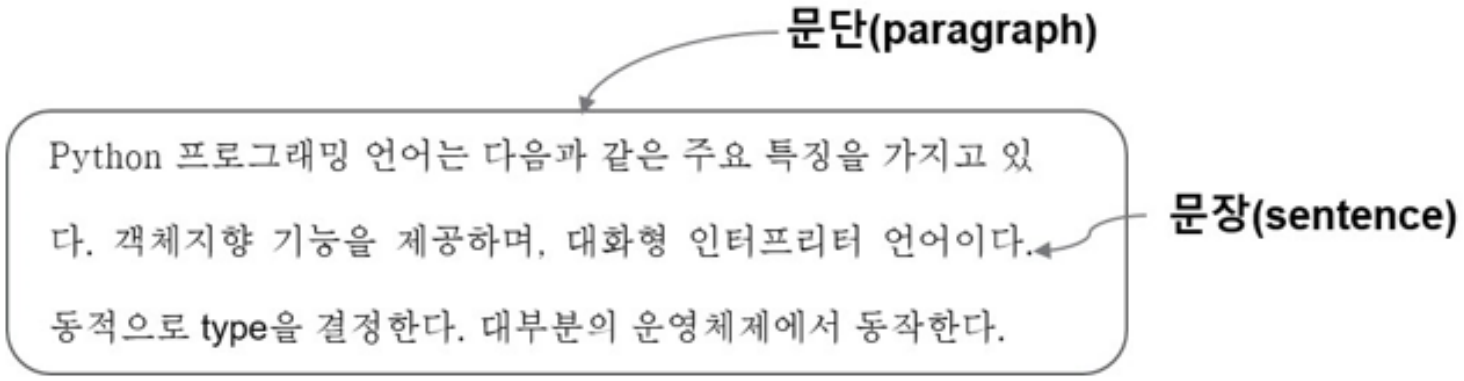
4. 중첩 반복문

- 구구단 예
- (1) 바깥쪽 반복문
- range(2, 10) 함수를 이용하여 2~9까지 단을 만들고, format() 함수를 이용하여 단을 출력한다.
- (2) 안쪽 반복문
- range(1, 10) 함수를 이용하여 1~9까지 곱해지는 수를 만들고, 단(i 변수)과 곱해지는 수(j 변수)를 곱해서 구구단을 계산하고, 출력한다.

4. 중첩 반복문

- (2) 문장과 단어 추출

- 일반적으로 문단은 여러 개의 문장을 포함하고, 문장은 여러 개의 단어로 구성된다. 마침표(.)를 기준으 로 문장을 구분할 경우는 4개의 문장으로 구성된 문단의 구조이다. 하나의 문단으로 부터 문장과 단어를 추출하는 과정도 중첩 반복문을 이용할 수 있다.



4. 중첩 반복문

chapter03.lecture.step03_for.py

```
string = """나는 홍길동 입니다.
주소는 서울시 입니다.
나이는 35세 입니다."""

sents = [] # 문장 저장
words = [] # 단어 저장

# (1) 문단 -> 문장
for sen in string.split(sep = "\n") :


    sents.append(sen)

    # (2) 문장 -> 단어
    for word in sen.split() :
        words.append(word)


print('문장 :', sents)
print('문장수 :', len(sents))
print('단어 :', words)
print('단어수 :', len(words))
```

Python Console

```
문장 : ['나는 홍길동 입니다.', '주소는 서울시 입니다.', '나이는 35세 입니다.']
문장수 : 3
단어 : ['나는', '홍길동', '입니다.', '주소는', '서울시', '입니다.', '나이는', '35세', '입니다.']
단어수 : 9
```

4. 중첩 반복문

- 문장과 단어 추출 예
- (1) 문단 -> 문장
- string 변수는 3줄의 문장으로 구성되어 있다. 따라서 줄 단위로 문장을 분리하기 위해서 string.split(sep = "₩n") 명령문을 이용한다. "₩n" 기호에 의해서 줄 단위로 문자열이 분리되고, 순서대로 sents 변수에 추가된다.
- (2) 문장 -> 단어
- 바깥쪽 반복문에서 한 개의 문장(sen)를 받아서 sen.split() 명령문으로 단어를 추출한다. split()함수에서 sep 인자를 생략하면 기본적으로 공백을 기준으로 문자열이 분리된다. 분리된 단어7)는 words 변수에 순서대로 추가된다.

THANK YOU