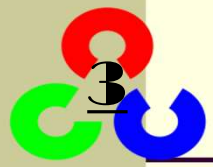
CHAPTER 03
파이썬 기초 실습2

김 찬, 윤 영 선

ckim.esw@gmail.com, ysyun@hnu.kr

정보통신공학과

목차



3.1 파이썬의 자료구조

3.0 print

출력 방법

- %

```
# solutuon1 (%)
print("배열에 있는 값은: %d %d %d 입니다." % (a[0], a[1], a[2]))
```

- .format

```
# solution2 (.format)
print("배열에 있는 값은: {} {} {} 입니다.".format(a[0], a[1], a[2]))
```

- f-string

```
# solution3 (f-string)
print(f"배열에 있는 값은: {a[0]} {a[1]} {a[2]} 입니다.")
```

- 입력
  - a = [10, 20, 30]

3.1 파이썬의 자료구조

3.1 for문

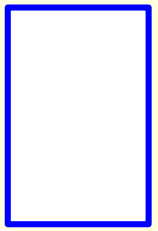
- list1 = [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’]
- list2 = [10, 20, 30, 40, 50, 60, 70]

enumerate

- Index, Value를 한번에 사용할 수 있게 해 주는 파이썬의 내장 함수

```
for idx, l1 in enumerate(list1):
    print(idx, l1)
```

- 출력 예시

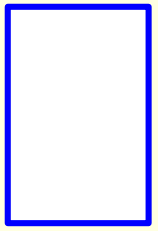


zip

- 두개 이상의 배열을 한번에 사용할 수 있게 해주는 파이썬의 내장 함수

```
for l1, l2 in zip(list1, list2):
    print(l1, l2)
```

- 출력 예시



3.1 파이썬의 자료구조

3.1 for문

조건

- 입력
  - names = [“ 홍길동”, “이순신”, “세종“, “아이언맨“, “슈퍼맨”]
  - scores = [30, 50, 100, 20, 70]

- 조건
  - names와 scores배열이 주어진다.
  - 주어진 배열을 이용하여 dict 형식으로 출력하여라.

- 출력 예시
  - {'홍길동': 30, '이순신': 50, '세종': 100, '아이언맨': 20, '슈퍼맨': 70}

3.1 파이썬의 자료구조

3.1 for문

- 입력
  - names = [“ 홍길동”, “이순신”, “세종“, “아이언맨“, “슈퍼맨”]
  - scores = [30, 50, 100, 20, 70]

range

```
# solution1 (range)
results = {}
for i in range(5):
    results[names[i]] = scores[i]
print(results)
```

for each

```
# solution2 (for each)
results = {}
for name in names:
    results[name] = scores[names.index(name)]
print(results)
```

3.1 파이썬의 자료구조

3.1 for문

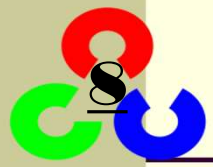
- 입력
  - names = [“ 홍길동”, “이순신”, “세종“, “아이언맨“, “슈퍼맨”]
  - scores = [30, 50, 100, 20, 70]

◆ enumerate

```
# solution3 (enumerate)
results = {}
for idx, name in enumerate(names):
    
print(results)
```

◆ zip

```
# solution4 (zip)
results = {}
for name, score in zip(names, scores):
    
print(results)
```



3.1 파이썬의 자료구조

3.2 함수와 라이브러리

◆ 함수 사용 방법

```
def 함수명(인수1, 인수2, ...):
  명령문 1
  명령문 2
  ...
    return 반환값
```

- **사용 예제**

```
def name(data):
  if data == "김찬":
    return "김찬이 맞습니다."

print(name("김찬"))
```

- **출력 예제**

```
김찬이 맞습니다.
```

3.1 파이썬의 자료구조

3.2 함수와 라이브러리

◆ 조건

- 입력
  - 자유
- 조건
  - 구구단을 1단부터 9단까지 출력하라.

- 출력 예시

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8

...

9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
```

3.1 파이썬의 자료구조

3.2 함수와 라이브러리

◆ 함수(function)

- 코드의 재사용 용이

```
# solution1 (for)
for i in range(1, 10):
    print(f"{2} x {i} = {2*i}")
for i in range(1, 10):
    print(f"{3} x {i} = {3*i}")

...

for i in range(1, 10):
    print(f"{8} x {i} = {8*i}")
for i in range(1, 10):
    print(f"{9} x {i} = {9*i}")
```

```
# solution2 (function)
def gogodan(num):
```

3.1 파이썬의 자료구조

- 3.2 함수와 라이브러리
  - 내장함수
    - Python 개발자들이 미리 작성해 둔 함수
      - len() – 문자의 개수, 배열의 개수 등..
      - enumerate() – index와 value로 값을 한번에 출력
      - zip() – 2개 이상의 배열을 한번에 출력
      - Min() – 배열에서 가장 작은 값 출력
      - max() – 배열에서 가장 큰 값 출력
      - print() – 출력

        …

3.1 파이썬의 자료구조

- 3.2 함수와 라이브러리
  - 모듈, 라이브러리, 패키지
    - 모듈
      - 클래스, 함수 등을 모아 둔 파일
    - 라이브러리, 패키지
      - 같은 뜻이라고 생각하면 편함
      - 모듈을 모아 둔 폴더

3.1 파이썬의 자료구조

3.2 함수와 라이브러리

모듈, 라이브러리, 패키지...

- 라이브러리 불러오기

```
# c
#include <timer.h>


# java
import java.util.Scanner;


# python
import numpy
```

- 불러온 라이브러리 사용하기

```
# c
time_t t;
t = time(null);

# java
Scanner in = new Scanner(System.in);


# python
numpy.array([1, 2, 3])
```

3.1 파이썬의 자료구조

- 3.2 함수와 라이브러리
  - 모듈, 라이브러리, 패키지...
    - 간편하게 사용하기

```
# original
import numpy
```

```
# original
numpy.array([1, 2, 3])
```

```
# used as
import numpy as np
```

```
# used as
np.array([1, 2, 3])
```

3.1 파이썬의 자료구조

3.3 numpy

◆ 행렬 원소로 지정값 생성하기

```
# example1
import numpy as np

zeros = np.zeros((2, 5), np.int32)      # 0으로 된 2행 5열 행렬 생성
ones = np.ones((3, 1), np.uint8)        # 1로 된 3행 1열 행렬 생성
emptys = np.empty((1, 5), np.float64)   # 비어있는 1행 5열 행렬 생성
fulls = np.full(5, 15, np.float32)      # 15로 된 1차원 행렬 생성

print(zeros)
print(ones)
print(emptys)
print(fulls)
```

```
[[0 0 0 0 0]
 [0 0 0 0 0]]
[[1]
 [1]
 [1]]
[[3.9155e-313 1.8399e+222 1.6758e+243 8.8241e+199 4.3385e-313]]
[15. 15. 15. 15. 15.]
```

3.1 파이썬의 자료구조

3.3 numpy

◆ 행렬 원소로 임의값 생성하기

```
# example2
import numpy as np

np.random.seed(10)                          # 랜덤시드 10으로 설정
a = np.random.rand(2, 3)                    # 2행 3열로 된 랜덤 행렬 생성
b = np.random.randint(1, 100, 6)            # 1 ~ 100 랜덤 행렬 6개 생성
print(a)
print(b)
b = b.reshape(2,-1)                         # 1차원 행렬을 2행으로 변경
print(b)
```

```
[[0.77132064 0.02075195 0.63364823]
 [0.74880388 0.49850701 0.22479665]]
[ 9 74  1 41 37 17]
[[ 9 74  1]
 [41 37 17]]
```

3.1 파이썬의 자료구조

3.3 numpy 실습1

조건

- 입력
  - list1 = np.array([1, 2, 3, 4, 5, 6, 7, 8])
  - list2 = np.array([10, 20, 30, 40, 50, 60, 70, 80])

- 조건
  - 위 두개의 배열을 (2, 4) 형태로 바꾸고, 두 배열의 합과 곱을 출력하라.

- 출력 예시

```
[[11 22 33 44]
 [55 66 77 88]]
[[ 10  40  90 160]
 [250 360 490 640]]
```

3.1 파이썬의 자료구조

3.3 numpy 실습1

- 1. 위 두개의 배열을 (2, 4) 형태로 바꾸기.
- 2. 두 배열의 합과 곱을 출력하기.

```
list1 = np.array([1, 2, 3, 4, 5, 6, 7, 8])
list2 = np.array([10, 20, 30, 40, 50, 60, 70, 80])
```

```

```

```
print(          )
print(          )
```

- *출력 예시*

```
[[11 22 33 44]
 [55 66 77 88]]
[[ 10  40  90 160]
 [250 360 490 640]]
```

3.1 파이썬의 자료구조

3.3 numpy 실습2

◆ 조건

- 입력
  - 자유

- 조건
  - 실수형 원소 10개를 갖는 ndarray 행렬을 랜덤으로 선언해서 전체 원소와 합과 평균을 구하라.

- 출력 예시

```
[[0.77132064 0.02075195 0.63364823 0.74880388 0.49850701 0.22479665
  0.19806286 0.76053071 0.16911084 0.08833981]]

4.113872595619601

0.41138725956196015
```

3.1 파이썬의 자료구조

3.3 numpy 실습2

numpy + for

```
# solution1
import numpy as np

np.random.seed(10)
np_a = np.random.rand(10)
sum_a = 0
for a in np_a:
    sum_a += a
print(np_a)
print(sum_a)
print(sum_a/10)
```

numpy + 내장함수

```
# solution2
import numpy as np

np.random.seed(10)
np_a = np.random.rand(1, 10)
print(np_a)
print(np_a.sum())
print(np_a.mean())
```

```
[[0.77132064 0.02075195 0.63364823 0.74880388 0.49850701 0.22479665
  0.19806286 0.76053071 0.16911084 0.08833981]]

4.113872595619601

0.41138725956196015
```

3.1 파이썬의 자료구조

3.3 numpy 실습3

◆ 조건

- 입력
  - 자유

- 조건
  - 0~50 사이의 임의의 원소(정수형, 중복가능)를 500개 만들어서 가장 많이 나온 **원소값**과 **중복횟수**로 출력하라.

- 출력 예시

```
가장 많이 나온 원소: 102
중복횟수: 4회
```

3.1 파이썬의 자료구조

3.3 numpy 실습3

◆ 가장 많이 중복 된 원소값 + 중복횟수 출력

```
import numpy as np

np.random.seed(10)
result = {}
arr = np.random.randint(0, 50, 500)

# 딕셔너리에 중복값을 담는 과정
```

3.1 파이썬의 자료구조

3.3 numpy 실습3

◆ 가장 많이 중복 된 원소값 + 중복횟수 출력

- **For-loop로 가장 많이 나온 원소 값 찾기**

```
# solution1
## 중복이 가장 많은 값만 변수에 저장
max_key, max_value = 0, 0
for k, v in result.items():
    if v > max_value:
        max_value = v
        max_key = k

print(f'가장 많이 나온 원소: {max_key}')
print(f'중복횟수: {max_value}회')
```

- ***출력 예시***

```
가장 많이 나온 원소: 102
중복횟수: 4회
```

3.1 파이썬의 자료구조

3.3 numpy 실습3

가장 많이 중복 된 원소값 + 중복횟수 출력

- 익명함수(람다, lambda)

```
# solution2
## 중복이 많은 횟수대로 정렬
result = sorted(result.items(), key=lambda x:x[1], reverse=True)
print(result)

print(f'가장 많이 나온 원소: {result[0][0]}')
print(f'중복횟수: {result[0][1]}회')
```

- *출력 예시*

```
가장 많이 나온 원소: 102
중복횟수: 4회
```

- 2번째, 3번째로 많이 중복 된 원소값도 찾기 쉬워짐