파이썬 프로그래밍

PORTFOLIO

20192612 김정혁

목차

01 파이썬의 개요
- 파이썬이란

02 파이썬 설치 & 실행
- 개발 도구 설치와 파이썬 쉘의 실행 화면

03 단계별 함수학습
- 단원 순서로 알아보는 함수학습

03 응용 문제풀이
- 학습 내용 점검차원
- 자기주도 학습 목적

04 포트폴리오 소감
- 작성한 후의 느낀점

1. 파이썬의 개요

파이썬의 개요란?

파이썬은 배우기 쉬운 프로그래밍 언어

파이썬은 배우기 쉽고 누구나 무료로 사용할 수 있는 오픈 소스 프로그래밍 언어이다. 파이썬은 1991년 네덜란드의 귀도 반 로섬이 개발했으며, 현재는 비영리 단체인 파이썬 소프트웨워 재단이 관리하고 있다. 파이썬의 사전적 의미는 '비단뱀'으로, 그리스 신화에서 유래됐으며, 파이썬 로고가 비단뱀인 것은 바로 이 때문이다. 그런데 실제로는 개발자인 귀도가 애청하던 영국의 코미디 프로그램 'Monty Python's Flying Circus'의 주인공인 6인조 코미디 그룹의 이름 'Monty Python'에서 따온 것이라고 한다.

현재 파이썬의 인기는 최고다

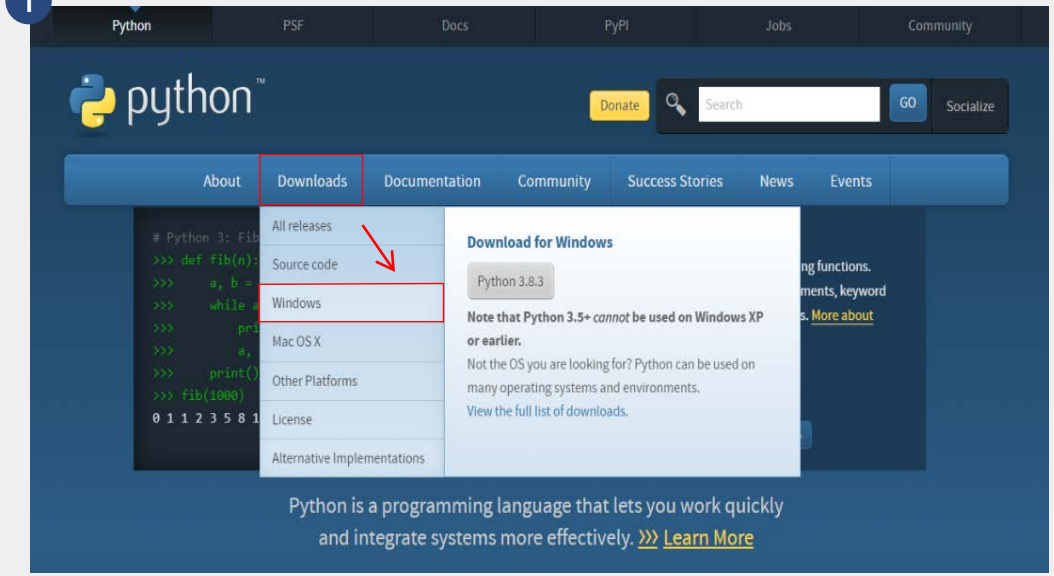
파이썬은 현재 미국과 우리나라의 대학 등 전 세계적으로 가장 많이 가르치는 프로그래밍 언어 중 하나다. 특히 비전공자의 컴퓨팅 사고력을 키우기 위한 프로그래밍 언어로도 많이 활용되고 있다. 파이썬은 배우기 쉽고 간결하며, 개발 속도가 빠르고 강력하기 때문이다. 또한 파이썬은 라이브러리가 풍부하고 다양한 개발환경을 제공하고 있어 개발자가 쉽고 빠르게 소프트웨어를 개발하는 데 도움을 준다. 파이썬은 프로그래밍 교육 분야뿐만 아니라 실무에서도 사용이 급증하고 있으며, 이를 증명하듯 유명한 '스택오버플로' 사이트에서 '가장 빠르게 성장하는 프로그래밍 언어'로 선택되기도 했다.

02. 파이썬 설치

www.python.org

- 최신 버전 내려 받아 설치
- 현재 3.8

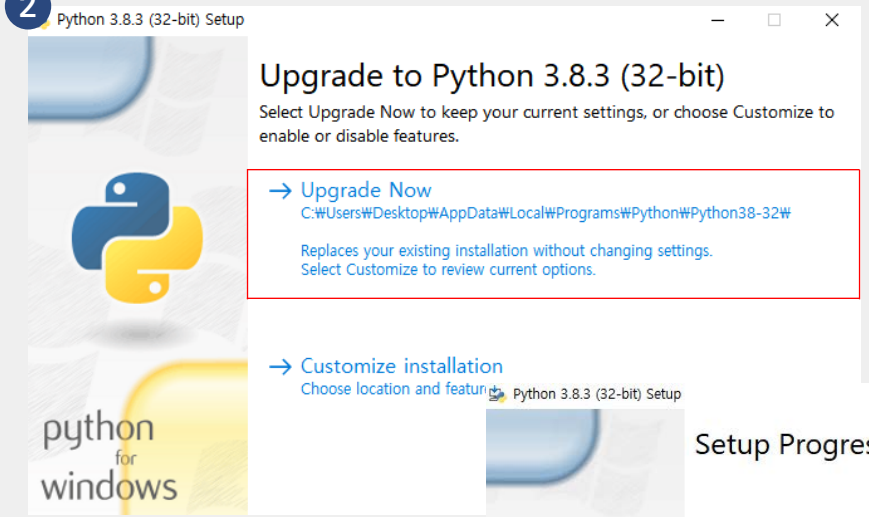
1

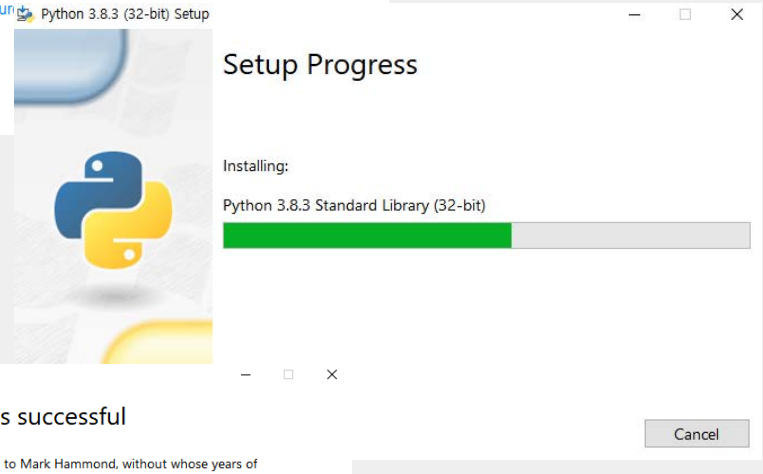


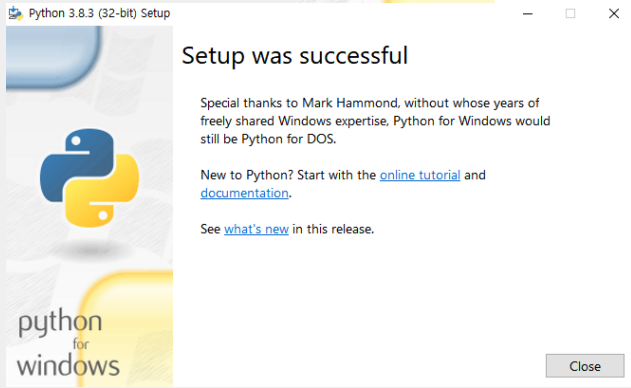
▲ Downloads – 자신에게 맞는 운영체제 클릭 후 다운로드

▼ 파이썬 설치 시작 화면

2



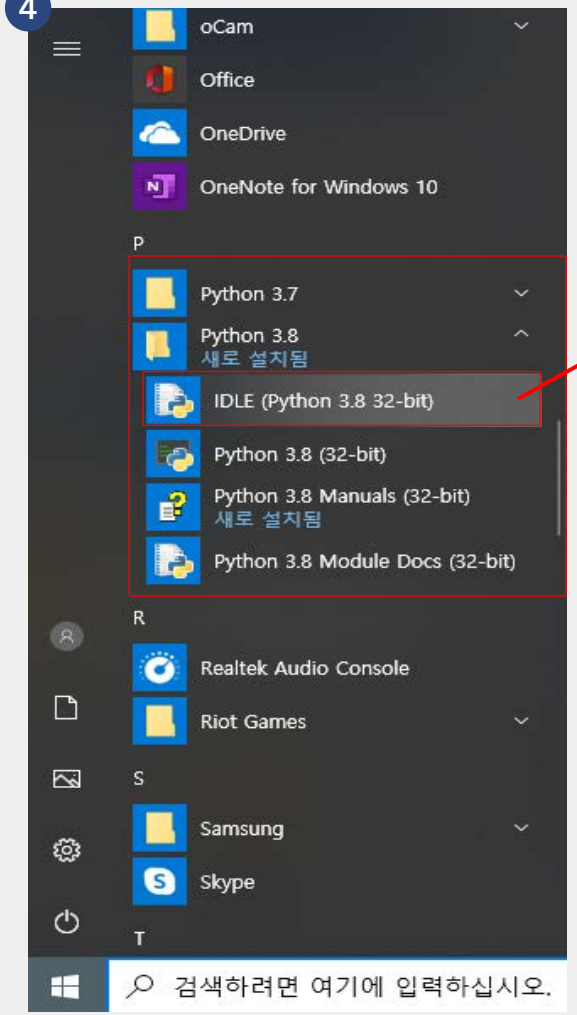




02. 파이썬 실행

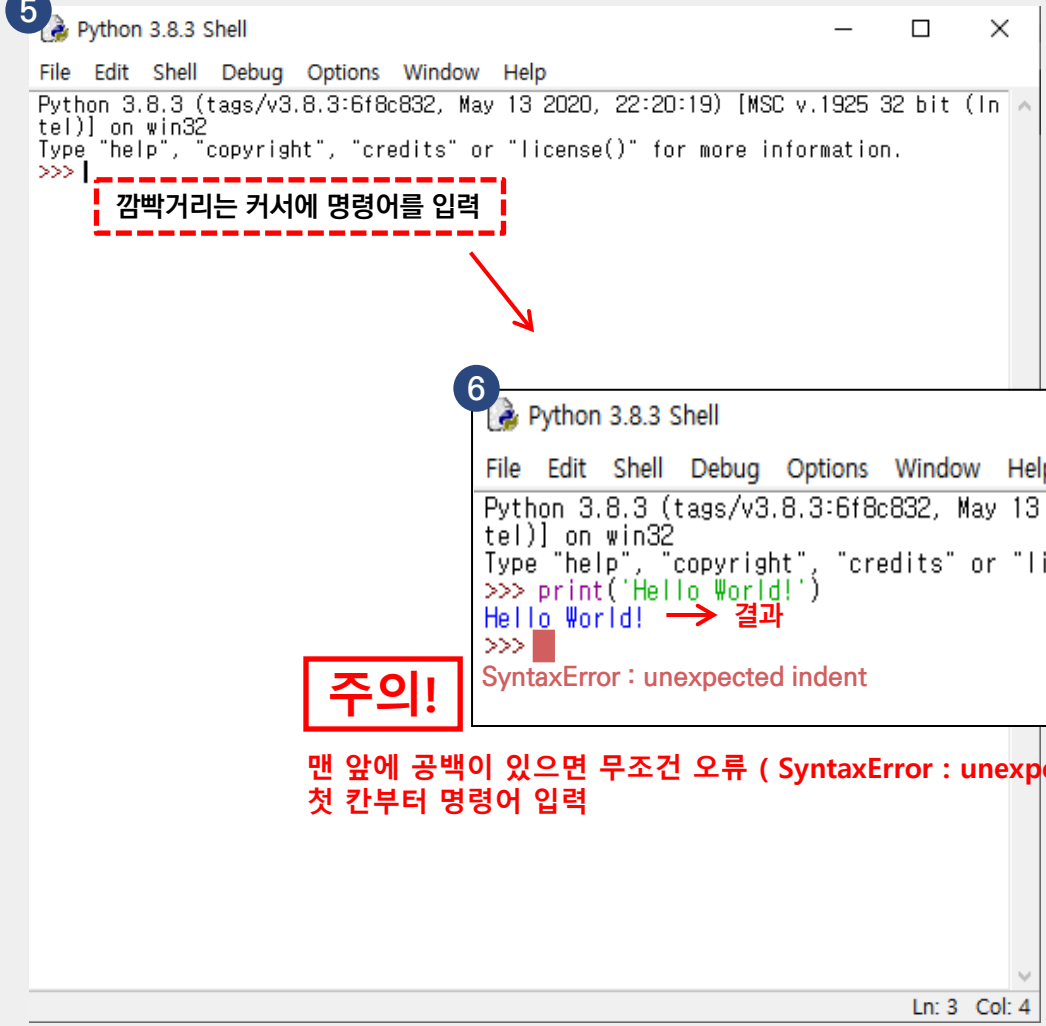
‘파이썬 쉘 IDLE’ 을 실행

4



– 설치된 [python 3.8] – [IDLE (python 3.8 32-bit)] 선택

5



6

주의!

맨 앞에 공백이 있으면 무조건 오류 (SyntaxError : unexpected indent)
첫 칸부터 명령어 입력

▲ 파이썬의 실행 메뉴와 ‘파이썬 쉘 IDLE’

Section 01

다양한 자료 : 문자열과 수

문자열의 연결 연산자 +와 반복 연산자 *

- 더하기 기호인 +는 문자열에서 문자열을 연결(concatenation)하는 역할을 한다.
- * 는 문자열에서 문자열을 지정된 수만큼 반복하는 연산은 수행한다.
- 문자열과 문자열 사이에 공백이 있어도 상관이 없으며, 두 문자열로만 반복 연산자를 사용하면 오류가 발생한다.

2 – 1 코딩 | 02-01stringop.py | 문자열 연결과 반복 연산자 +, * 의 활용 | 난이도 기본

8 lines (8 sloc) | 328 Bytes | Raw | Blame | History

```
>>> print("원의 원주율 " + '3.11592')
원의 원주율 3.141592
>>> print("python" 'programming ' + 'language')
python programming language
>>> print('파이썬 언어는' + " 강력하다")
파이썬 언어는 강력하다
>>> print('파이썬 ' + "언어! " + 3 * "방가")
파이썬 언어! 방가 방가 방가
```

03 단계별 함수학습 Ch02

문법과 상관 없는 주석

- 주석은 소스 설명으로, 인터프리터는 주석을 무시한다.
- 파이썬 주석은 #으로 시작하고 그 줄의 끝까지 유효하다.
- 주석 #은 한 줄만 가능하므로 여러 줄의 주석이 필요하면 맨 앞에 #을 넣거나 삼중 따옴표를 사용한다.

2 – 2 코딩 | **02-02comments.py I 주석 #과 여러 줄 문자열에 삼중 따옴표 활용** | **난이도 기본**

7 lines (6 sloc) | 404 Bytes | Raw | Blame | History

```
1  ''' 01-02comments.py
2      2019 3. by Kang Hwan Soo '''
3
4      print('# 이후는 주석') # 한 줄에서 문장 이후에도 주석 사용 가능
5      print('string: "python"') # 큰 따옴표 내부에서 작은따옴표는 문자열
6      print("number: 1 5 3.14") # 문자열 내부에서 숫자도 문자열
7      print("string: 'python'") # 작은따옴표 내부에서 작은따옴표는 문자열
```

03 단계별 함수학습 Ch02

산술 연산자의 계산 우선순위

연산자	명칭	의미	우선순위	예
+	더하기(add)	두 피연산자를 더하거나 수의 부호	4, 2	4 + 5, +3
-	빼기(subtract)	두 피연산자를 빼거나 수의 부호	4, 2	9 – 5, -7
*	곱하기(multiply)	두 피연산자 곱하기	3	3 * 4
/	나누기(divide)	왼쪽을 오른쪽 피연산자로 나누기	3	10 / 4
%	나머지(moudulus)	왼쪽을 오른쪽 피연산자로 나눈 나머지	3	21 % 4
//	몫 나누기(floor division)	왼쪽을 오른쪽 피연산자로 나눈 결과에서 작거나 같은 정수	3	10 // 3
**	거듭제곱, 지수승(exponent)	왼쪽을 오른쪽 피연산자로 거듭제곱	1	2 ** 3

2 – 3 코딩 02-03twopdist.py I 두 점 사이의 거리 계산 난이도 응용

4 lines (3 sloc) 101 Bytes Raw Blame History

```
1  print(4 ** (1/2))
2  print( ((3**2 + 4**2)) ** (1/2) )
3  print( ((2-3.1)**2 + (5-4.8)**2) ** (1/2) )
```

Section 02

변수와 키워드, 대입 연산자

변수의 이해와 대입 연산자 =을 이용한 값의 저장

- 변수란, 말 그대로 '변하는 자료를 저장하는 메모리 공간'이다.
- 변수는 대소문자의 영문자, 숫자, 그리고 _로 구성되며, 대소문자는 구별된다.
- 변수의 맨 앞에 숫자는 올 수 없으며, import, True, False 등과 같은 키워드는 사용할 수 없다.

2 – 7 코딩 | 02-07variable.py I 변수 이름 규칙 | 난이도 기본

9 lines (8 sloc) | 140 Bytes | Raw | Blame | History

```
1  value = 10
2  Value = 200
3  value
4  Value
5  total_price = 100000
6  coffeePrice = 4500
7
8  _month = 11
9  2020year = 2020 # 오류: 숫자로 시작
```

03 단계별 함수학습 Ch02

동일한 변수에 값을 수정하는 다양한 대입 연산자

다양한 대입 연산자	형태	의미	의미
=	a = b	a = b	b의 결괏값을 변수 a에 저장
+=	a += b	a = a + b	a + b의 결괏값을 변수 a에 저장
-=	a -= b	a = a - b	a - b의 결괏값을 변수 a에 저장
*=	a *= b	a = a * b	a * b의 결괏값을 변수 a에 저장
/=	a /= b	a = a / b	a / b의 결괏값을 변수 a에 저장
%=	a %= b	a = a % b	a % b의 결괏값을 변수 a에 저장
//=	a //= b	a = a // b	a // b의 결괏값을 변수 a에 저장
**=	a **= b	a = a ** b	a ** b의 결괏값을 변수 a에 저장

2 - 8 코딩 02-08celsius.py I 섭씨 온도를 화씨 온도로 변환 난이도 기본

6 lines (6 sloc) | 259 Bytes Raw Blame History

```
1  celsius = 37
2  fahrenhite = celsius * 9/5 + 32 # 화씨로 변환
3  print('섭씨: ', celsius, ',', '화씨: ', fahrenhite)
4  celsius += 3 # 5도 증가
5  fahrenhite = celsius * 9/5 +32 # 화씨로 변환
6  print('섭씨: ', celsius, ',', '화씨: ', fahrenhite)
```

03 단계별 함수학습 Ch02

한 번에 여러 자료 대입

- 파이썬은 콤마로 구분된 여러 변수에 순서대로 값을 대입할 수 있다.

```
>>> a, b = 5, 9
>>> print(a, b)
5 9
```

divmod() 함수

- divmod(a, b)는 나누기 몫 연산 //와 나머지 연산 %를 함께 수행해 2개의 결과를 반환한다.

2 – 10 코딩 02-10moondist.py | 지구와 달까지의 거리를 만 단위로 출력 난이도 기본

4 lines (4 sloc) 166 Bytes Raw Blame History

```
1  distance = 384400
2  unit = 10000
3  manUnit, remainder = divmod(distance, unit)
4  print('지구에서 달까지의 거리: ', manUnit, '만', remainder, '킬로미터')
```

Section 03

자료의 표준 입력과 자료 변환 함수

함수 input()으로 문자열 표준 입력

- 함수 input()은 입력되는 표준 입력을 문자열로 읽어 반환하는 함수
- Input()에서 반환된 입력 문자열을 변수에 저장하려면 대입 연산자 =을 사용해 변수에 저장해야 한다.
- 변수의 맨 앞에 숫자는 올 수 없으며, import, True, False 등과 같은 키워드는 사용할 수 없다.

2 – 11 코딩 02-11input.py | 학교와 이름을 입력받아 출력 난이도 기본

3 lines (3 sloc) 101 Bytes Raw Blame History

```
1  univ = input('대학은? ')
2  name = input('이름은? ')
3  print('대학: ', univ, '이름: ', name)
```

03 단계별 함수학습 Ch02

10진수의 변환 함수 bin(), oct(), hex()

- 10진수를 바로 16진수, 8진수, 2진수로 표현되는 문자열로 변환하려면

↓

- 각각 함수 bin(), oct(), hex()를 사용해야 한다.

2 – 15 코딩 02-15base.py | 10진수 정수를 입력받아 2진수, 8진수, 10진수, 16진수 출력 | 난이도 응용

6 lines (5 sloc) 245 Bytes Raw Blame History

```
1  data = int(input('정수 입력 >> '))
2
3  print('2진수: ', bin(data)) # 2진수로 출력
4  print('8진수: ', oct(data)) # 8진수로 출력
5  print('10진수: ', data) # 10진수로 출력
6  print('16진수: ', hex(data)) # 16진수로 출력
```

Section 01

문자열 다루기

함수 len()으로 문자열의 길이 참조

- 함수 len(문자열)으로 문자열의 길이를 나타낸다.

[start:end]

- 0과 양수일 때거나 음수일 때, 첨자방식대로 start 첨자에서 end-1 첨자까지의 문자열을 반환
- start와 end를 비우면 '처음부터'와 '끝까지'를 의미

3 – 3 코딩 | 03-03strstep.py I 다양한 문자열 슬라이싱 | 난이도 기본

8 lines (8 sloc) | 365 Bytes | Raw | Blame | History

```
1  str = '일요일 기러기'
2  print(str[:3], str[4:]) # 양수 이용
3  print(str[:-4], str[-3:]) # 음수 이용
4  print(str[:], str[::], str[::1]) # 모든 문자열 참조
5  print(str[::2]) # 한 문자씩 건너뛰기
6  print(str[::3]) # 두 문자씩 건너뛰기
7  print(str[::-2]) # 역순으로 한 문자씩 건너뛰기
8  print(str[::-1]) # 역순으로 출력
```

Section 02

문자열 관련 메소드

문자열 바꿔 반환하는 메소드 replace()

- 함수 len(문자열)으로 문자열의 길이를 나타낸다.

```
>>> str = '안녕하세요'
>>> str.replace('하세요', '히가세요')
안녕히가세요
```

부분 문자열 출현 횟수를 반환하는 메소드 count()

- 문자열에서 문자나 부분 문자열의 출현 횟수를 알려준다.

```
>>> str = '안녕하세요 안녕하세요 안녕안녕안녕'
>>> str.count('안녕')
5
```

문자와 문자 사이에 문자열을 삽입하는 메소드 join()

- 문자열의 문자와 문자 사이에 원하는 문자열을 삽입할 때 사용한다.

```
>>> num = '12345'
>>> '->'.join(num)
'1->2->3->4->5'
```

03 단계별 함수학습 Ch03

문자열을 찾는 메소드 find()와 index()

- 맨 처음에 위치한 첨자를 반환받을 때 사용
- 오류가 발생 시 find는 -1을 반환 index는 ValueError를 발생시킨다.

```
>>> str = '자바 C 파이썬 코틀린'
>>> str.find ('파이썬')
5
```

문자열을 여러 문자열로 나누는 split() 메소드

- 문자열 메소드 split()는 문자열에서 공백을 기준으로 문자열을 나눠준다.

```
>>> '사과 배 딸기 복숭아 포도'.split()
['사과', '배', '딸기', '복숭아', '포도']
```

영문자 알파벳의 다양한 변환 메소드

```
'python'.upper() # 모두 대문자로 변환해 반환
'PYTHON'.lower() # 모두 소문자로 변환해 반환
'python lecture'.capitalize() # 첫 문자만 대문자로 변환해 반환
'python lecture'.title() # 단어마다 첫 문자를 대문자로 변환해 반환
'PyThOn'.swapcase() # 소문자와 대문자를 서로 변환해 반환
```

```
'PYTHON'
'python'
'Python lecture'
'Python Lecture'
'pYtHoN'
```

03 단계별 함수학습 Ch03

문자열의 format() 메소드를 이용해 간결한 출력 처리

- 문자열 '{} + {} = {}' 내부에서 중괄호 {}가 위치한 부분에 format(3, 4, 3+4) 인자인 3, 4, 7이 순서대로 출력된다.
- 문자열에서 {}을 제외한 나머지 부분인 '+='은 쓰여 있는 그대로 출력된다.

```
>>> str = '{} + {} = {}' .format(3, 4, 3 + 4)
>>> print(str)
3 + 4 = 7
```

형식 유형	의미
d, n	10진수 정수이며, n은 국가에 맞는 구분자를 추가
c	유니코드 수에 대응하는 문자 출력
f, F	기본적으로 소수점 여섯 자리까지 실수로 출력하며, F인 경우는 'inf', 'nan' 표시를 대문자 'INF', 'NAN'로 표시
b	2진수 정수
o	8진수 정수
x, X	16진수 표현으로 a~f까지 소문자와 대문자로 각각 표시
e, E	지수 형식 3.141592e + 00으로 지수 표현이 각각 소문자 e와 대문자 E
g, G	실수를 일반적으로는 소수점 형식으로 출력하지만 커지면 지수승으로 표시
%	퍼센트 형식, 인자의 100배를 소수점으로 출력하고 맨 뒤에 %를 출력
s	문자열 형식이며, 기본적으로 왼쪽 정렬, 그러나 수 형식은 모두 기본이 오른쪽 정렬

Section 01

조건에 따른 선택 if ··· else

조건에 따른 선택을 결정하는 if문

- If문에서 논리 표현식 이후에는 반드시 콜론이 있어야 한다.
- 콜론 이후 다음 줄부터 시작되는 블록은 반드시 들여쓰기를 해야 한다. 그렇지 않으면 오류 발생

If 논리 표현식 :

    문장 1
    문장 2

If문은 조건인 논리 표현식의 결과가 True이면 이후에 구성된 블록 구문인 문장1과 문장2를 실행한다.
그러나 결과가 False이면 실행하지 않는다.

4 – 1 코딩 | 04-01rideif.py I 키가 140이상이면 놀이 기구 타기 | 난이도 기본

3 lines (3 sloc) | 124 Bytes | Raw | Blame | History

```
1  height = 152 # 탑승을 체크할 키를 대입
2  if 140 <= height:
3      print('롤러코스터 T-Express, 즐기세요!')
```

03 단계별 함수학습 Ch04

논리 표현식 결과인 True와 False에 따라 나뉘는 if ··· else문

- If문에서 논리 표현식 결과가 True이면 논리 표현식 콜론 이후 블록을 실행하고 False이면 else: 이후의 블록을 실행한다.

```
If 논리 표현식 :
    문장 1
    문장 2
else :
    문장 3
    문장 4
```

4 – 3 코딩 | 04-03earlybirddiscount.py I 영화 조조 할인 판정 | 난이도 응용

8 lines (7 sloc) | 267 Bytes | Raw | Blame | History

```
1  from time import localtime
2  hour = localtime().tm_hour
3  mnt = localtime().tm_min
4
5  if hour < 10:
6      print ('지금 시각 : %d시 %d분, 조조 할인 된다. '%(hour,mnt))
7  else:
8       print ('지금 시각 : %d시 %d분, 조조 할인 안 된다. '%(hour,mnt))
```

Section 02

반복을 제어하는 for문과 while문

◆ 정해져 있는 시퀀스의 항목 값으로 반복을 실행하는 for문

- 여러 개의 값을 갖는 시퀀스에서 변수에 하나의 값을 순서대로 할당해 블록의 문장들을 순차적으로 실행한다.
- 반복 몸체인 문장 1, 문장 2에서 변수를 사용할 수 있다.

```
for 변수 in 시퀀스 :
    문장 1
    문장 2
else :
    문장 3
```

else 이후인 문장 3 은 반복이 종료된 마지막에 실행된다.

4 – 7 코딩 | 04-07numseq.py | 수의 나열에서 합과 평균 구하기 | 난이도 기본

6 lines (6 sloc) | 127 Bytes | Raw | Blame | History

```
1  sum = 0
2  for i in 1.1,2.5,3.6,4.2,5.4:
3      sum += i
4      print(i,sum)
5  else:
6      print('합:%.2f,평균:%.2f' %(sum,sum/5))
```

03 단계별 함수학습 Ch04

반복 구조가 간단한 while 반복

- 논리 표현식이 True이면 반복 몸체인 문장 1, 2를 실행한 후 다시 논리 표현식을 검사해 실행한다.
- 논리 표현식이 False이면 반복 몸체를 실행하지 않고, 선택 사항인 else: 이후의 블록을 실행한 후 반복을 종료한다.

```
while 논리 표현식 :
    문장 1
    문장 2
else :
    문장 3
```

4 – 10 코딩 04-10checkrides.py | 어린이를 위한 놀이 기구 탑승 검사 — 난이도 응용

15 lines (14 sloc) | 384 Bytes — Raw | Blame | History

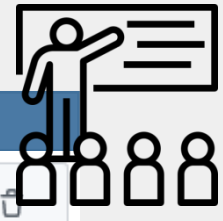
```
1  MAXNUM = 4
2  MAXHEIGHT = 130
3
4  more = True
5  cnt = 0
6  while more:
7      height = float(input("키는? "))
8      if height < MAXHEIGHT:
9          cnt+=1
10         print('들어가세요.','%d명' %cnt)
11     else : print('커서 못 들어갑니다.')
12     if cnt == MAXNUM :
13         more = False
14     else :
15         print('%d명 모두 찼습니다. 다음 번에 이용하세요.'%cnt)
```

Section 03

임의의 수인 난수와 반복을 제어하는 break문, continue문

임의의 수를 발생하는 난수

- 함수 random(시작, 끝)을 사용해 정수 시작과 끝 수 사이에서 임의의 정수를 얻을 수 있다.
- 여기서는 시작과 끝을 모두 포함한다.



4 – 13 코딩 04-13lotto.py | 1에서 45까지의 6개 수를 맞히는 로또 — 난이도 응용

16 lines (16 sloc) | 398 Bytes — Raw | Blame | History

```
winnumber = 11, 17, 28, 30, 33, 35
print(' 모의 로또 당첨 번호 '.center(28, '='))
print(winnumber)
print()
print(' 내 번호 확인 '.center(30, '-'))
cnt = 0
import random
for i in range(6):
    n = random.randint(1, 45)
    if n in winnumber:
        print(n, 'O ', end = ' ')
        cnt += 1
    else:
        print(n, 'X ', end = ' ')
print()
print(cnt, '개 맞음')
```

03 단계별 함수학습 Ch04

반복을 종료하는 break문

- for나 while 반복 내부에서 문장 break는 else: 블록을 실행시키지 않고 반복을 무조건 종료한다.

무한 반복

```
While True :
        반복 문장들
```

반복을 종료

```
While True :
        …
        break
        …
```

4 – 14 코딩 04-14menubreak.py | 1을 입력하면 계속하고 0을 입력하면 반복 종료 난이도 기본

5 lines (5 sloc) | 110 Bytes — Raw | Blame | History

```
1  while True:
2      menu = input('[0]종료 [1]계속 ? ')
3      if menu == '0':
4          break
5  print('종료')
```

03 단계별 함수학습 Ch04

continue 이후의 반복 몸체를 실행하지 않고 다음 반복을 계속 실행

- 반복 for와 while문 내부에서 continue 문장은 이후의 반복 몸체를 실행하지 않고 다음 반복을 위해 논리 조건을 수행한다.

for에서 continue

```
for 변수 in 시퀀스 :
    ...
        continue
    ...
```

while에서 continue

```
while 논리 표현식 :
    ...
        continue
    ...
```

4 – 16 코딩 04-16dayspelltest.py I 월, 화, 수 중 영어 철자 하나 검사 난이도 응용

11 lines (9 sloc) | 319 Bytes — Raw | Blame | History

```
1  days = ['monday', 'tuesday' 'wednesday']
2
3  while True:
4      user = input('월, 화, 수 중 하나 영어 단어 입력 >> ')
5      if user not in days:
6          print('잘못 입력했어요!')
7          continue
8      print('입력: %s, 철자가 맞습니다.' % user)
9      break
10
11 print(' 종료 '.center(15, '*'))
```

Section 01

여러 자료 값을 편리하게 처리하는 리스트

관련된 나열 항목을 관리하는 리스트

- 리스트는 대괄호(square brackets) [ ] 사이에 항목을 기술한다.

[항목 1, 항목 2, 항목 3, ...] # 리스트

5 – 1 코딩 | 05-01coffee.py | 다양한 커피 종류가 저장된 리스트 | 난이도 기본

9 lines (7 sloc) | 198 Bytes | Raw | Blame | History

```
1  coffee = ['에스프레소', '아메리카노', '카페라테', '카페모카']
2  print(coffee)
3  print(type(coffee))
4
5  num = 0
6  for s in coffee:
7      num += 1
8      print('%d. %s' % (num, s))
9
```

03 단계별 함수학습 Ch05

리스트의 메소드 count()와 index()

- 리스트의 메소드 count(값) - 값을 갖는 항목의 수를 반환
- index(값) – 인자인 값의 항목이 위치한 첨자를 반환
- 리스트의 항목 수정이 가능하다.

5 – 6 코딩 05-06foodorder.py | 중국집에서 음식 주문하기 난이도 응용

12 lines (11 sloc) | 307 Bytes Raw Blame History

```
1  food = ['짜장면', '짬뽕', '우동', '울면']
2  print(food)
3  # 탕수육 주문 추가
4  food.append('탕수육')
5  print(food)
6  # 짬뽕을 굴짬뽕으로 주문 변경
7  food[1] = '굴짬뽕'
8  print(food)
9
10 # 우동을 물만두로 주문 변경
11 food[food.index('우동')] = '물만두'
12 print(food)
```

Section 02

리스트의 부분 참조와 항목의 삽입과 삭제

첨자 3개로 리스트 일부분을 참조하는 슬라이싱

- 0에서 시작하는 오름차순, 마지막 요소 -1에서 시작하는 내림차순 첨자도 가능하며, 두 순차의 첨자를 함께 사용가능

리스트[start : stop : stop]

5 – 8 코딩 | 05-08swordslice.py ㅣ 한 글자의 한국 단어로 이해하는 리스트 슬라이싱 | 난이도 기본

14 lines (14 sloc) | 393 Bytes | Raw | Blame | History

```
1  wlist = ['밥', '삶', '길', '죽', '꿈', '차', '떡', '복', '말']
2  print('wlist[:] = ', wlist[:])
3  print('wlist[::] = ', wlist[::])
4  print('wlist[::-1] = ', wlist[::-1])
5  # 오름차순
6  print(wlist[::3])
7  print(wlist[1::3])
8  print(wlist[2::3])
9  # 내림차순
10 print(wlist[::-2])
11 print(wlist[-1:-8:-3])
12 # 오름과 내림차순의 혼재
13 print(wlist[1:-1:])
14 print(wlist[-2:-9:-3])
```

Section 03

항목의 순서나 내용을 수정할 수 없는 튜플

수정할 수 없는 항목의 나열인 튜플

- 튜플은 리스트와 달리 **항목의 순서나 내용의 수정이 불가능**하다.
- 튜플은 괄호 (…) 사이에 항목을 기술하며 괄호는 생략 가능하다.
- 빈 튜플은 ( )로 만들며, 튜플의 자료형 이름은 클래스 tuple이다.

(항목 1, 항목 2, 항목 3, …) **# 튜플**

5 – 12 코딩 | 05-12kpoptuble.py ㅣ K – pop 가수와 곡으로 구성된 튜플의 참조 | 난이도 기본

13 lines (10 sloc) | 389 Bytes | Raw | Blame | History

```
1  singer = ('BTS', '불빨간사춘기', 'BTS', '블랙핑크', '태연')
2  song = ('작은 것들을 위한 시', '나만, 봄', '소우주', 'Kill This Love', '사계')
3  print(singer)
4  print(song)
5
6  print(singer.count('BTS'))
7  print(singer.index('불빨간사춘기'))
8  print(singer.index('BTS'))
9  print()
10
11 for _ in range(len(singer)):
12     print('%s: %s' % (singer[_], song[_]))
13
```

Section 01

키와 값인 쌍의 나열인 딕셔너리

키와 값의 쌍을 항목으로 관리하는 딕셔너리

- 딕셔너리는 중괄호 {…} 사이에 키와 값의 항목을 기술한다.
- 딕셔너리의 항목순서는 의미가 없으며, 키는 중복될 수 있다.
- 키는 수정될 수 없지만, 값은 수정될 수 있다.
- 값은 키로 참조된다.

```
dict = { <key>: <value>, <key>: <value>, … , <key>: <value>, }  # 딕셔너리
```

6 – 1 코딩 | 06-01groupnumber.py | K – pop 그룹의 인원수 | 난이도 기본

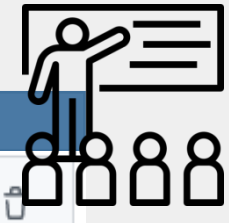
3 lines (3 sloc) | 138 Bytes | Raw | Blame | History

```
1  groupnumber = {'잔나비': 5, '트와이스': 9, '블랙핑크': 4, '방탄소년단': 7}
2  print(groupnumber)
3  print(type(groupnumber))
```

03 단계별 함수학습 Ch06

리스트 또는 튜플로 구성된 키- 값을 인자로 사용

- 내장함수 dict() 함수에서 인자로 리스트나 튜플 1개를 사용해 딕셔너리를 만들 수 있다.
  - >>> day = dict([ ]) # 빈 딕셔너리 또는 >>> day = dict(( )) # 빈 딕셔너리
- 키가 문자열이면 키 = 값 항목의 나열로도 딕셔너리 생성이 가능하다.



6 – 3 코딩 | **06-03btsdict.py I 방탄소년단 정보를 저장하는 다양한 딕셔너리 생성과 참조** | **난이도 기본**

15 lines (12 sloc) | 627 Bytes — Raw | Blame | History

```
1  bts1 = {'그룹명': '방탄소년단', '인원수': 7, '리더': '김남준'}
2  bts1['소속사'] = '빅히트 엔터테인먼트';
3  print(bts1)
4  bts2 = dict([['그룹명', '방탄소년단'], ['인원수', 7]])
5  print(bts2)
6  bts3 = dict((('리더', '김남준'), ('소속사', '빅히트 엔터테인먼트')))
7  print(bts3)
8
9  bts = dict(그룹명 = '방탄소년단', 인원수=7, 리더='김남준', 소속사='빅히트 엔터테인먼트')
10 # 구성원 추가
11 bts['구성원'] =  ['RM', '진', '슈가', '제이홉', '지민', '뷔', '정국']
12
13 print(bts) # 전체 출력
14 print(bts['구성원']) # 구성원 출력
```

03 단계별 함수학습 Ch06

딕셔너리 메소드 keys(), items(), values()

- 딕셔너리 메소드 keys()는 키로만 구성된 리스트를 반환한다.
- 딕셔너리 메소드 items()는 (키, 값) 쌍의 튜플이 들어 있는 리스트를 반환한다.
- 딕셔너리 메소드 values()는 값으로 구성된 리스트를 반환한다.

반복 for문에서 딕셔너리 변수로만 모든 키 순회

6 – 5 코딩 | 06-05seasondict.py ㅣ 사계절의 영어 사전 생성과 항목 순회 | 난이도 응용

16 lines (14 sloc) | 540 Bytes | Raw | Blame | History

```
1  season = {'봄': 'spring', '여름': 'summer', '가을': 'autumn', '겨울': 'winter'}
2  print(season.keys())
3  print(season.items())
4  print(season.values())
5
6  # 메소드 keys()로 항목 순회
7  for key in season.keys():
8      print('%s %s ' % (key, season[key]))
9
10 for item in season.items():
11     print('{} {} '.format(item[0], item[1]), end= ' ')
12 print()
13 # 메소드 items()의 반환 값인 튜플을 한 변수에 저장한 경우, 항목 순회 2
14 for item in season.items():
15     print('{} {} '.format(*item), end= ' ')
16 print()
```

Section 02

중복과 순서가 없는 집합

원소는 유일하고 순서는 의미 없는 집합

- 원소는 불변 값으로 중복될 수 없으며 서로 다른(unique) 값이어야 한다.
- 즉, 원소는 **중복을 허용하지 않으며** 원소의 **순서는 의미가 없다**.

```
{ 원소 1, 원소 2, … 원소 n }
```

집합을 만드는 내당 함수 set()

- 인자가 없으면 빈 집합인 공집합이 생성된다.
- 인자가 있으면 하나이며, 리스트와 튜플, 문자열 등이 올 수 있다.

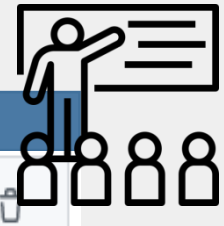
```
set( 원소로 구성된 리스트_or_튜플_or_문자열 )
```

{원소 1, 원소 2, … }로 생성

- 중괄호 { } 안에 직접 원소를 콤마로 구분해 나열하는 방법이다.
- 집합의 원소는 문자, 문자열, 숫자, 튜플 등과 같이 변할 수 없는 것이어야 한다.
- 리스트나 딕셔너리는 원소로 사용할 수 없다.

03 단계별 함수학습 Ch06

전 슬라이드에서 배운 집합을 이용해 한 글자 단어로 구성된
집합 만들기를 출력해보자
(함수 set( ) 이용하기)



6 – 7 코딩 | 06-07onecharset.py ㅣ 한 글자 단어의 집합 만들어보기 | 난이도 기본

10 lines (9 sloc) | 265 Bytes | Raw | Blame | History

```
1  planets = set('해달별')
2  fruits = set(['감', '귤'])
3  nuts = {'밤', '잣'}
4  things = {('밤', '잣'), ('감' '귤'), '해달'}
5  # things = {['밤', '잣'], ('감', '귤'), '해달'} # 오류 발생
6
7  print(planets)
8  print(fruits)
9  print(nuts)
10 print(things)
```

4. 포트폴리오 소감

01 진행 시간

2틀 소요

대략 포트폴리오만 완성 시키기 까지는
2틀 정도 소요가 된 것 같다.
다른 용무도 딱히 없었으며 오로지
포트폴리오에만 몰두했지만 꽤 많은 시간이
소요됐다고 생각한다.

02 구성

목차 구성에 대해

우선 어떤식으로 구성을 진행해야 할 지를 가장
신중하고 오래 생각했던 것 같다.
결과적으로 내가 제일 잘 알아볼 수 있는 방식인
내용과 그것에 기반한 코딩풀이로 나만의
요점정리 방식인 포트폴리오를 작성했다.

04 느낀점

소감

분량과 그만큼의 시간을 투자해 작성한다는 건
꽤 빠듯한 시간이었지만, 한 번 작성하고
더 이상 안보고 과제만 끝났다고 생각하기 보단
작성하면서 그 동안의 재 복습과 놓쳤던 부분들을
다시 한 번 짚고 넘어가는 계기가 된 것 같아
의미있는 중간고사라고 생각이 들었다.

04 마지막으로

하고싶은 말

끝으로 학생 교과목 포트폴리오를 마치며,
대면으로 진행하고 있진 않지만 항상
교과목을 위해 열심히 가르쳐주셔서
감사의 말씀 드립니다.
그럼 코로나 조심하시고 다음에 뵙겠습니다.

THANK YOU!

MADE BY Jeonghyeok