리뷰

Day-2

print()

- **print( )**
  - print()는 모니터에 텍스트를 출력해주는 함수
  - 반드시 소문자만을 사용

• 소스코드

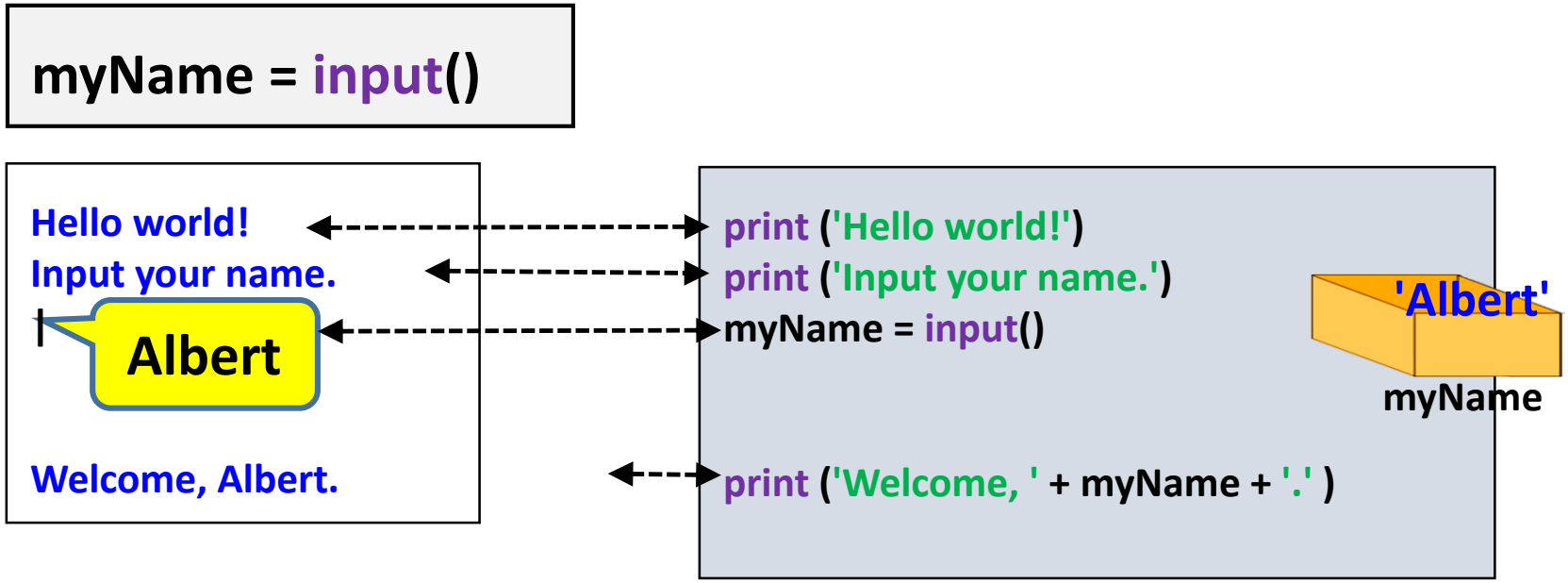
```
print ('Hello world!')
print ('Input your name.')
```

• 실행결과

```
Hello world!
Input your name.
>>>
```

input()

- input()
  - **프로그램 실행 중에** 키보드로부터 데이터를 입력 받고자 할 때 사용하는 함수
  - 프로그램에서 input()을 만나면 쉘 윈도우에서 커서가 깜박거리면서 입력을 기다림
  - 입력 받은 데이터는 일반적으로 변수에 저장함
    - ✓ 변수 없이 input()만 선언하면 데이터는 입력 받지만 저장하지는 않음



형 변환 함수

- **형 변환의 정의**
  - 데이터의 타입을 다른 타입으로 변환하는 것

- **종류**
  - int() 함수 : **문자를 정수로** 변환하거나, **실수를 정수로** 변환하는 함수
  - str() 함수 : **숫자(정수,실수)를 문자로** 변환하는 함수
  - float() 함수 : **정수를 실수로** 변환하는 함수

- **형 변환이 필요한 이유**
  - input() 함수로 입력 받은 데이터는 문자열 처리됨
    - ✓ 예)숫자 17를 입력하면 문자열 '17'이 됨
  - 연산을 위해, 문자열 <-> 숫자 간 형 변환이 필요할 때가 있음

int() 함수

- 형식

```
int(변수명 or '문자열' or 실수)
```

- Quiz

```
>>> int('42')
>>> int(42)
>>> int(' 42 ')
>>> 2 + int('2')
>>> int('hello')
>>> int('forty-two')
```

```
>>> number= int('42')
>>> number = '42'
>>> number = int(number)
>>> number = int(input())
100
>>> number
```

str() 함수

- 형식

```
str(변수명 or 정수 or 실수)
```

- Quiz

```
>>> str(42)

>>> str(3.5)
```

```
>>> age = 10

>>> newAge = str(age + int('2'))

>>> newAge + ' years old'
```

float() 함수

- 형식

```
float(변수명 or '문자열'  or 정수)
```

- Quiz

```
>>> float(42)

>>> float('3.5')
```

```
>>> temp = '10.4'
>>> newTemp = float(temp) + 2
>>> newTemp
```

조건문

- 조건문 형식 1

```
if 비교식:
    실행문장
```

조건문에는 세가지 형식이 있어요.

- 조건문 형식 1의 일반 예

파이썬으로 어떻게 작성할까요?

score가 60점 이상이면 조건

→ 합격입니다. 실행문장

- 조건문 형식1 예

```
if score >= 60:
    print ( '합격입니다.' )
```

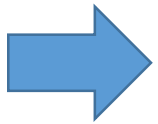
조건문

- 조건문 형식 2

```
if 비교식:
    실행문장
else :
    실행문장
```

- 조건문 형식 2의 일반 예

score가 60점 이상이면 조건
→ 합격입니다. 실행문장
그렇지 않으면 위의 조건이 아닐때
→ 불합격입니다. 실행문장



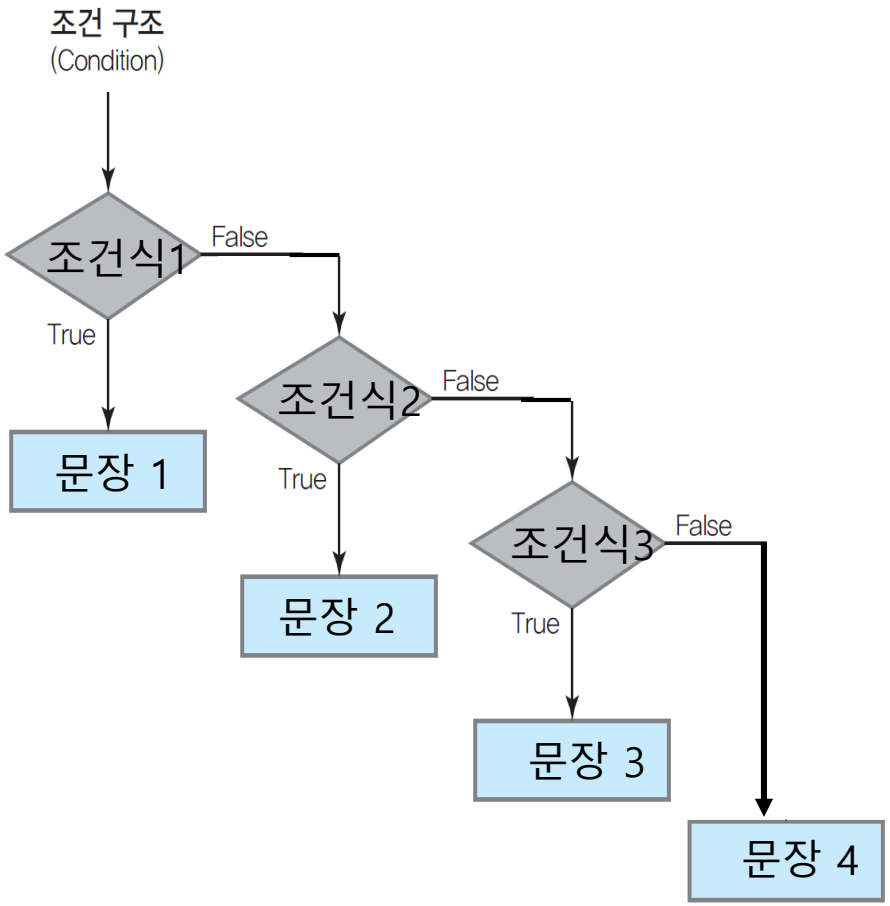
- 조건문 형식2 예

```
if score >= 60:
    print ('합격입니다. ' )
else :
    print ('불합격입니다.' )
```

조건문

- 조건문 형식 3

```
if 비교식1:
    실행문장1
elif 비교식2:
    실행문장2
elif 비교식3:
    실행문장3
else :
    실행문장4
```



- 조건문 형식3 예

```
if score >= 90:
    print ('A입니다. ' )
elif score >= 80 :
    print ('B입니다.' )
elif score >= 70 :
    print ('C입니다.' )
elif score >= 60 :
    print ('D입니다.' )
else:
    print ('F입니다.' )
```

Container

Data Type

- 숫자형 = numeric
- 문자, 문자열
- 불리언 = Boolean
- **Compound = Container = Collection**
  - 기본적인 데이터 타입을 조합하여, 여러 개의 값을 하나의 단위로 묶어서 다루는 데이터 타입
  - 논리적으로 이들은 데이터 타입인 동시에 데이터의 구조(흔히 말하는 자료 구조)의 한 종류. 보통 다른 데이터들을 원소로 하는 집합처럼 생각되는 타입들
- None

Data Type

- **Compound = Container = Collection**
  - Sequence
    - list : 순서가 있는 원소들의 묶음
    - tuple : 순서가 있는 원소들의 묶음. 리스트와 혼동하기 쉬운데 단순히 하나 이상의 값을 묶어서 하나로 취급하는 용도로 사용
    - range
  - Lookup
    - mapping
      - dictionary : 그룹내의 고유한 이름인 키와 그 키에 대응하는 값으로 이루어지는 키값 쌍(key-value pair)들의 집합.
    - set : 순서가 없는 고유한 원소들의 집합.

list

list

- **필요성**

학생 5명의 성적 데이터

```
s0 = 90
s1 = 93
s2 = 67
s3 = 82
s5 = 25
```

```
s5 = 80
s6 = 92
s7 = 57
s8 = 20
s9 = 95
```

백명의 학생 데이터를 저장하려면 백 개의 변수를 만들어야 할까요??

list

- 여러 개의 데이터를 하나의 변수에 저장하고자 할 때 매우 유용함
- 리스트에 들어있는 데이터를 아이템이라고 부름
- 리스트는 [ ]로 묶어주고, 리스트의 아이템은 ,(컴마)로 구분함
- 리스트의 아이템에 접근하고자 할 때에는 인덱스를 사용함
- 인덱스는 0부터 시작함!!

```
>>> shape= ['turtle', 'arrow', 'circle', 'square']
>>> shape[0]
'turtle'
>>> shape[1]
'arrow'
>>> shape[2]
'circle'
>>> shape[3]
'square'
```

Quiz

```
>>> shape[4]
```

```
>>> shape[-1]
```

list

- 인덱스번호를 잘못 준 경우, 에러메시지가 나옴

```
>>> shape= ['turtle', 'arrow', 'circle', 'square']
>>> shape[4]
IndexError: list index out of range
```

- 리스트에서 인덱스를 음수로 준 경우, 반대로 시작 하며 -1인 경우, 맨 마지막 위치에 있는 요소를 가져옴

```
>>> shape= ['turtle', 'arrow', 'circle', 'square']
>>> shape[-1]
'square'
```

Quiz

```
>>> numbers[-11]
```

list

- 아이템 수정

```
>>> shape= ['turtle', 'arrow', 'circle', 'square']
>>> shape[1] = 'ARROW'
>>> shape
['turtle', 'ARROW', 'circle', 'square']
```

list

- 아이템 추가

```
>>> shape= ['turtle', 'ARROW', 'circle', 'square']
>>> shape.append('heart')
>>> shape
['turtle', 'ARROW', 'circle', 'square', 'heart']
```

- Quiz: 문자열 리스트에 숫자도 추가할 수 있을까요?

```
>>> shape.append(27)
>>> shape
['turtle', 'ARROW', 'circle', 'square', 'heart', 27]
```

list

- 아이템 삽입

```
>>> shape= ['turtle', 'ARROW', 'circle', 'square', 'heart', 27]
>>> shape.insert(1, 'heart')
>>> shape
['turtle', 'heart', 'ARROW', 'circle', 'square', 'heart', 27]
>>> shape.insert(3, 'triangle')
['turtle', 'heart', 'ARROW', 'triangle', 'circle', 'square', 'heart', 27]
```

list

- **아이템 검색**
  - 리스트 안에 검색 값이 있으면 True
  -

```
>>> shape= ['turtle', 'heart',  'ARROW', 'triangle', 'circle', 'square', 'heart', 27]
>>> 'heart' in shape
True
>>> 'rectangle' in shape
False
```

list

- 아이템 삭제

```
>>> shape= ['turtle', 'heart',  'ARROW', 'triangle', 'circle', 'square', 'heart', 27]
>>> del shape[1]
>>> shape
['turtle',  'ARROW', 'triangle', 'circle', 'square', 'heart', 27]
>>> del shape[1]
>>> shape
['turtle', 'triangle', 'circle', 'square', 'heart', 27]
```

list

- 아이템의 인덱스 번호 찾기

```
>>> shape
['turtle', 'triangle', 'circle', 'square', 'heart', 27]
>>> shape.index('circle')
2
```

list

- 리스트 슬라이싱(리스트의 일부를 자를 때 사용)

```
>>> shape = ['turtle', 'triangle', 'circle', 'square', 'heart', 27]
>>> shape[0:3]  #0부터 3이전까지: 0~2
['turtle', 'triangle', 'circle']
>>> shape[2:4]  #2부터 4이전까지: 2~3
['circle', 'square']
>>> shape[:2]  #처음부터 2이전까지: 0~1
['turtle', 'triangle']
>>> shape[3:]  #3부터 끝까지: 3~4
['square', 'heart', 27]
```

list

- Quiz

```
>>> shape = ['turtle', 'arrow', 'circle', 'square' , 'heart']
>>> shape[0:0] #답이 나올까요? 에러일까요?

>>> shape[:] #답이 나올까요? 에러일까요?
```

list

- 리스트 슬라이싱

```
>>> a=[1,6,2,4,8,5,3,10,9,7]

>>> a[ : :2]   #2씩 건너뛴 요소를 슬라이싱한다.
[1,2,8,3,9]

>>> a[ : :3]   #3씩 건너뛴 요소를 슬라이싱한다.
[1,4,3,7]
```

- 빈 리스트 만들기

```
>>> blank_list =[ ]
```

list

- 리스트 아이템 더하기 연산

```
>>> shape = ['turtle', 'arrow', 'circle', 'square' , 'heart']
>>> shape[0]  + shape[2]
'turtlecircle'
```

- 리스트 연결

```
>>> [1, 2, 3, 4] + ['Seoul', 'Kwangju'] + ['Alice', 'Bob']
[1, 2, 3, 4, 'Seoul', 'Kwangju', 'Alice', Bob']
```

list

- 리스트 안에 리스트(list of list)

```
>>> ID = [1, 2]
>>> area = ['Seoul', 'Kwangju']
>>> name = ['Alice', 'Bob']
>>> student = [ID, area, name]
>>> student
[ [1, 2], ['Seoul', 'Kwangju'], ['Alice', Bob'] ]
```

list

- Quiz

```
>>> ID = [1, 2]
>>> area = ['Seoul', 'Kwangju']
>>> name = ['Alice', 'Bob']
>>> student = [1, ID, area, name, 'pass' ] #여러 데이터 타입이 하나의 리스트에 들어갈 수 있을까요?
```

list

- 아이템 정렬1(오름차순)

```
>>> shape = ['turtle', 'arrow', 'circle', 'square' , 'heart']
>>> shape.sort()
>>> shape
['arrow', 'circle', 'heart', 'square', 'turtle']
```

```
>>> numbers = [ 8, 6, 3, 10, 2, 9, 1, 4, 7, 5]
>>> numbers.sort()
>>> numbers
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

list

- **아이템 정렬2**
  - 원본 데이터는 그대로 유지

```
>>> numbers = [ 8, 6, 3, 10, 2, 9, 1, 4, 7, 5]
>>> sorted_numbers = sorted(numbers)
>>> sorted_numbers
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> numbers
[8, 6, 3, 10, 2, 9, 1, 4, 7, 5]
```

리스트를 생성하는 다양한 방법

- **split() 메소드**
  - 문자열의 단어를 분리하여 리스트 생성

사용 방법	의미	예시
split()	공백을 기준으로 문자열을 나눠 리스트 생성	a = '1 2 3 4 5' b = a.split() → b = ['1', '2', '3', '4', '5']
split(문자)	문자를 기준으로 문자열을 나눠 리스트 생성	a = '1:2:3:4:5' b = a.split(':') → b = ['1', '2', '3', '4', '5']

리스트를 생성하는 다양한 방법

- split() 메소드 응용

```
>>> wordlist = '''ant baboon badger bat bear beaver camel cat
clam cobra cougar coyote crow deer dog donkey duck eagle
ferret fox frog goat goose hawk lion lizard llama mole
monkey moose mouse mule newt otter owl panda parrot pigeon
python rabbit ram rat raven rhino salmon seal shark sheep
skunk sloth snake spider stork swan tiger toad trout turkey
turtle weasel whale wolf zebra'''.split()
```

```
>>> wordlist
['ant', 'baboon', 'badger', 'bat'.....'zebra' ]
```

list() 함수

- **list() 함수**
  - list 함수: list형으로 형변환

사용 방법	의미	예시
list(문자열)	문자열을 한 글자씩 분리하여 리스트 생성	list('12345') → ['1', '2', '3', '4', '5']

```
>>> list('hello')
['h', 'e', 'l', 'l', 'o']
```

range

range() 함수

- range 함수: 규칙적인 숫자열로 구성된 범위 생성

사용 방법	의미	리스트생성
range(끝)	[0, 1, 2, 3, …, 끝-1]	list(range(4)) → [0, 1, 2, 3]
range(시작, 끝)	[시작, 시작+1, 시작+2, …, 끝-1]	list(range (3, 5)) → [3, 4]
range(시작, 끝, 스텝)	[시작, 시작+스텝, 시작+스텝×2, …, 시작+스텝×k]	list(range(2, 11, 2)) → [2, 4, 6, 8, 10]
range(시작, 끝, -스텝)	[시작, 시작-스텝, 시작-스텝×2, …, 시작-스텝×k]	list(range(9, 1, -2)) → [9, 7, 5, 3]

```
>>> range(10)
range(0,10)     => 의미 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
>>> range(10, 20)
range(10,20)    => 의미 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
>>> list(range(10, 20))
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

range() 함수

```
>>> a= range(1,100)
>>> a[2:5]  # 인덱스2부터 5까지 슬라이싱으로 range를 만듦
range(3,6)
>>> b=a[2:5]
[3,4,5]
>>> b[0]=5   #range는 불변객체이므로 재할당할 수 없음
에러
```

len() 함수

- len() 함수
  - 리스트의 개수, 문자열의 길이를 반환하는 함수

```
>>> len(wordlist)
63
>>> len('hello')
5
```

문자 리스트 생성 예제

- 예제

```
>>> a = 'Hello world'
>>> b = list(a)
>>> c = a.split()
>>> d = a.split('o')
>>> a
'Hello world'
>>> b
['H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd']
>>> c
['Hello', 'world']
>>> d
['Hell', ' w', 'rld']
```

숫자 list 생성 예제

- 예제

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(3, 7))
[3, 4, 5, 6]
>>> list(range(10, 1, -2))
[10, 8, 6, 4, 2]
>>> list(range(1, 7, -1))
[]
```

random

random

- **random 모듈**
  - 난수를 발생시키는 것과 관련된 함수들을 모아놓은 집합
  - random 모듈을 불러오는 형식

- **형식**

```
import random

random.randint(범위의 시작 값, 범위의 끝 값)
```

random 모듈의 randint( ) 함수를 이용하여 범위의 값중에서 하나의 난수를 발생시킴

random

- random의 다양한 예

```
>>> import random
>>> random.randint(1, 20)
12
>>> random.randint(1, 20)
18
>>> random.randint(1, 20)
3
>>> random.randint(1, 4)
3
>>> random.randint(1000, 2000)
1294
```

random

- Quiz

```
>>> random.randint(100, 100)

>>> random.randint(5.0, 10.0)

>>> random.randint(5.5, 10.0)

>>> random.randint(1)
```

```
>>> random.randint(1, 2, 3)

>>> num=input()
100
>>> random.randint(1, num)

>>> random.randint(100, 20)
```

random

- 아래의 단어들 가운데 랜덤한 단어를 하나 나타내도록 코드를 작성해보세요.

```
'''ant baboon badger bat bear beaver camel cat
clam cobra cougar coyote crow deer dog donkey duck eagle
ferret fox frog goat goose hawk lion lizard llama mole'''
```

```
>>> 선택된 secret word는 baboon입니다.
>>> 선택된 secret word는 crow입니다.
```

실습

실습1. 오늘의 명언

- 리스트에 여러개의 속담을 저장한 후, 속담중에서 하나를 랜덤하게 골라서 제공하는 프로그램을 작성하시오.

```
---------------------------------
        오늘의 명언
---------------------------------
고생없이 얻을 수 있는 진실로 귀중한 것은 하나도 없다.
>>>
```

- 속담리스트

```
꿈을 지녀라. 그러면 어려운 현실을 이길 수 있다.
고생없이 얻을 수 있는 진실로 귀중한 것은 하나도 없다.
사람은 사랑할 때 누구나 시인이 된다.
시작이 반이다.
나는 사랑으로 내가 이해하는 모든 것들을 이해한다.
```

실습2-1. 상품재고관리

- **리스트에 상품목록을 저장한 후, 상품을 삽입,삭제,추가,검색하는 프로그램을 작성하시오.**
  - 상품리스트

```
커피, 우유, 바나나, 감귤, 화장지, 장갑, 이불, 베개, 장난감, 음료수, 빵
```

  - 상품 삽입

```
-----------------------------------
삽입하고자 하는 상품을 입력하세요. 사과
삽입하고자 하는 위치를 입력하세요. 4
-----------------------------------
[커피, 우유, 바나나, 감귤, 사과, 화장지, 장갑, 이불, 베개, 장난감, 음료수, 빵]
>>>
```

실습2-2. 상품재고관리

- 상품 삭제

```
---------------------------------------
삭제하고자 하는 상품을 입력하세요. 장갑
---------------------------------------
[커피, 우유, 바나나, 감귤, 사과, 화장지, 이불, 베개, 장난감, 음료수, 빵]
>>>
```

실습2-3. 상품재고관리

- 상품 추가

```
---------------------------------------
추가하고자 하는 상품을 입력하세요. 물
---------------------------------------
[커피, 우유, 바나나, 감귤, 사과, 화장지, 이불, 베개, 장난감, 음료수, 빵, 물]
>>>
```

실습2-4. 상품재고관리

- 상품 검색

```
---------------------------------------
검색하고자 하는 상품을 입력하세요. 화장지
---------------------------------------
화장지가 검색되었습니다.
>>>
```

```
---------------------------------------
검색하고자 하는 상품을 입력하세요. 자동차
---------------------------------------
자동차가 검색되지 않았습니다.
>>>
```

논리연산자

논리연산자 and, or, not

- and 연산자

```
>>> True and True
True
>>> True and False
False
>>> False and True
False
>>> False and False
False
```

- or 연산자

```
>>> True or True
True
>>> True or False
True
>>> False or True
True
>>> False or False
False
```

- not 연산자

```
>>> not True
False
>>> not False
True
>>> True not
SyntaxError: invalid syntax
```

논리연산자 and, or, not

```
>>> 10 > 5

>>> 'python' != 'python '

>>> True or False

>>> False and True

>>> False or True and False

>>> 10 > 20 or 20 < 10

>>> python = 'programming'

>>> 5 > 20 and python == 'Hello'
```

반복문

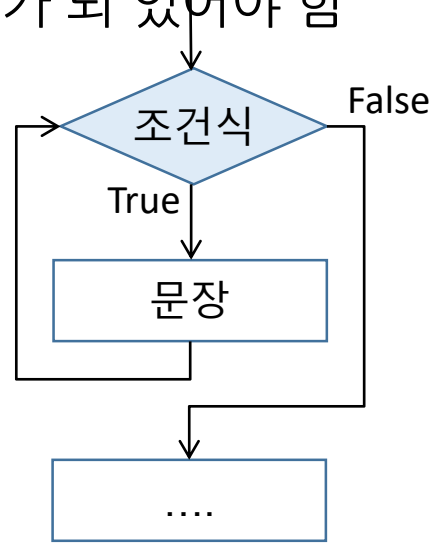
반복문

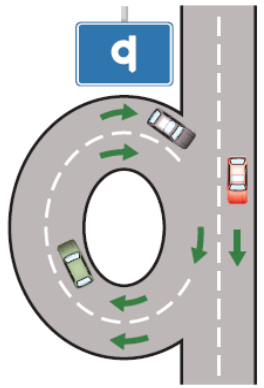
- while
- for

반복문

반복문

- 반복(iteration)은 동일한 문장을 여러 번 반복시키는 구조
- 조건식이 **True**인 동안 실행문장을 반복함
- if문과 마찬가지로 반복실행 되어야 할 문장은 들여쓰기가 되 있어야 함
- 컴퓨터는 반복작업을 빠르게 실행 할 수 있음





- **반복문 형식1**

```
while 조건식:
    실행문장
```

- **반복문 형식2**

```
while 1:
    ...
    break
```

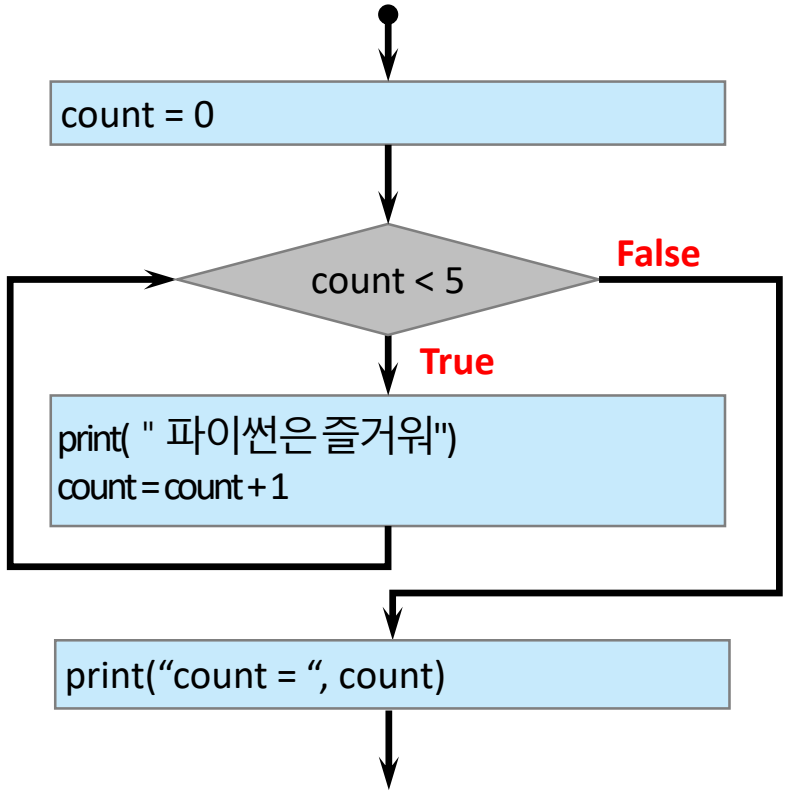
조건이 항상 True인 반복문
1은 True의 의미를 가짐
이 반복문을 빠져나오기 위해 break를 꼭 써야함

반복문

- **반복 조건식에 사용할 변수**
  - while 조건식에 사용할 변수는 반드시 먼저 정의되어 있어야 함

```
count = 0

while count < 5:
    print ('파이썬은 즐거워')
    count = count + 1
print ('count= ', count)
```



반복문

- **반복문**

```
for 변수 in 범위:
    실행문장1
    실행문장2
    실행문장3
```

- 범위의 처음부터 끝까지 반복함
- while과의 차이점
  - ✓ while: 조건비교결과가 참인 동안 실행문장들이 반복해서 실행됨
  - ✓ for : 범위의 처음부터 끝까지 반복해서 실행됨

반복문

- 범위에 숫자 리스트를 준 예제

```
for count  in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
    print ('나무를', count, '번 찍었습니다.', '딱! ' *count)
print ('넘어가즈아~~~')
```

- 동일한 표현

```
for count  in range(1, 11):
```

```
나무를 1 번 찍었습니다. 딱!
나무를 2 번 찍었습니다. 딱! 딱!
나무를 3 번 찍었습니다. 딱! 딱! 딱!
나무를 4 번 찍었습니다. 딱! 딱! 딱! 딱!
나무를 5 번 찍었습니다. 딱! 딱! 딱! 딱! 딱!
나무를 6 번 찍었습니다. 딱! 딱! 딱! 딱! 딱! 딱!
나무를 7 번 찍었습니다. 딱! 딱! 딱! 딱! 딱! 딱! 딱!
나무를 8 번 찍었습니다. 딱! 딱! 딱! 딱! 딱! 딱! 딱! 딱!
나무를 9 번 찍었습니다. 딱! 딱! 딱! 딱! 딱! 딱! 딱! 딱! 딱!
나무를 10 번 찍었습니다. 딱! 딱! 딱! 딱! 딱! 딱! 딱! 딱! 딱! 딱!
넘어가즈아~~~
>>>
```

반복문

- 구구단 예제

```
for i  in range
    for j  in range
        print
```

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
....

9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
>>>
```

반복문

- 범위에 문자열 리스트를 준 예제

```
for thing  in ['programming', 'pasta', 'spam', 'play park']:
    print ('I really like ' + thing + '.')
```

- 실행결과

```
I really like programming.
I really like pasta.
I really like spam.
I really like play park.
>>>
```

반복문

- 범위에 변수를 준 예제

```
stuff = ['programming', 'pasta', 'spam', 'play park']
for thing  in stuff:
    print ('I really like ' + thing + '.')
```

- 실행결과

```
I really like programming.
I really like pasta.
I really like spam.
I really like play park.
>>>
```

반복문

- 범위에 문자열을 준 예제

```
for i  in 'Hello world!':
      print (i, end=' ')
```

```
Hello world!
>>>
```

```
for i  in 'Hello world!':
      print (i)
```

```
H
e
l
l
o

w
o
r
l
d
!
>>>
```

반복문 실습

while문/for문 이용하여 나무 찍기

```
열번 찍어 안 넘어가는 나무없다!!!

나무를 1번 찍었습니다.
나무를 2번 찍었습니다.
나무를 3번 찍었습니다.
나무를 4번 찍었습니다.
나무를 5번 찍었습니다.
나무를 6번 찍었습니다.
나무를 7번 찍었습니다.
나무를 8번 찍었습니다.
나무를 9번 찍었습니다.
나무를 10번 찍었습니다.

간다~간다~넘어간다!!!
```

while문/for문 이용하여 동전모으기

- Scenario of “티끌 모아 태산”

```
티끌 모아 태산!!!

땡그랑, 동전 100원을 모았습니다.
땡그랑, 땡그랑, 동전 200원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 동전 300원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전 400원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전 500원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전 600원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전 700원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전 800원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전 900원을 모았습니다.
땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 땡그랑, 동전 1000원을 모았습니다.

산다. 산다. 집 산다!!!
```

while문/for문 로또번호 생성기

- 로또번호는 1~45사이 random.randint()를 사용
- 6개의 번호가 필요함.
- 중복번호가 나오지 않게 해보세요.

```
로또번호 생성기 프로그램
1번째 번호는 1입니다.
2번째 번호는 8입니다.
3번째 번호는 19입니다.
4번째 번호는 41입니다.
5번째 번호는 14입니다.
6번째 번호는 32입니다.
```

로그인 프로그램

- **올바른 암호를 입력할 때까지 암호를 입력받는 프로그램을 작성하세요. 올바른 암호를 입력하면 로그인 성공을 출력하세요.**
  - 알고리즘
    - ✓ 암호가 “pythonisfun”이 아니면 반복한다.
      - ❖ 암호를 입력하세요.
      - ❖ password=input()
    - ✓ 암호가 올바르면 “로그인 성공”을 출력한다.

```
암호를 입력하세요.
pythonis
암호를 입력하세요.
funfunfun
암호를 입력하세요.
pythonisfun
로그인 성공

>>>
```

구구단 출력

- 원하는 단을 입력 받아 그 단의 구구단을 출력하는 프로그램을 작성하세요.

```
원하는 단은: 9
9*1 = 9
9*2 = 18
9*3 = 27
9*4 = 36
9*5 = 45
9*6 = 54
9*7 = 63
9*8 = 72
9*9 = 81
>>>
```

random.randint()와 반복문 이용

- 입력한 값이 1보다 작은 경우 “잘못 입력하셨습니다.” 메시지를 출력한 후 다시 입력 받도록 작성해보세요.
- 1이상의 숫자를 입력했을 시에는 1부터 입력 받은 숫자까지를 범위로 하여 랜덤넘버 하나를 발생시켜 출력

```
1이상의 숫자를 입력해주세요.
-3
잘못 입력하셨습니다.
1이상의 숫자를 입력해주세요.
100
시작 범위 1부터 끝 범위 100 가운데 선택된 랜덤 넘버는 78입니다.
>>>
```

1~100까지 숫자 더하기

- 1~100까지 수를 더해 합을 출력하는 프로그램을 while/for문을 사용하여 작성하시오.

Summation / Average / Max / Min

- 몇개의 숫자가 입력될지 알 수 없다.
- 0을 입력하면 반복문을 빠져 나오도록 한다.
- 합, 평균, 최대값, 최소값을 출력한다.
- 평균을 실수타입으로 구한다.

```
Input numbers
910
11
9
15
3
0
The total is 48
The average is 9.6
The max number is 15
The min number is 3
>>>
```

tuple

tuple

- **튜플(tuple)**
  - 순서가 있는 원소들의 묶음. 리스트와 혼동하기 쉬운데 단순히 하나 이상의 값을 묶어서 하나로 취급하는 용도로 사용
  - 리스트와 동일하지만, 튜플은 불변객체이다는 차이점이 있음

```
>>> a=(1,2,3,4,5)
>>> a
(1,2,3,4,5)
>>> a[0]=0  #불변객체이므로 재할당이 되지 않는다.
```

tuple

- tuple()함수를 이용하여 튜플 생성

```
>>> b=tuple(range(1,10))
>>> b
(1,2,3,4,5,6,7,8,9)
```

- count() 메서드를 이용하여 아이템 개수 확인, index() 메서드로 인덱스 찾기

```
>>> b.count(7) # 아이템 7이 몇개인지 개수를 세어준다.
1
>>> b.index(7)  # 아이템 7의 인덱스를 알려준다.
6
```

tuple

- 다양한 tuple 값 대입 예

```
>>> x=1,
>>> type(x)
tuple
>>> x=1,2
>>> type(x)
tuple
>>> x,y=1,2
>>> type(x)
```

```
>>> (x,y) =(1,2)
>>> type(x)

>>> type((x,y))
```

```
>>> x=1, (2,3)
>>> x=(1,)
>>> x=(1, (2,3))
```

dictionary

dictionary

- **딕셔너리**
  - 키-값 쌍(key-value pair)으로 이루어진 자료구조
  - 중괄호{ }로 묶음

```
>>> a={'a' : 1, 'b': 2}
>>>  a
{'a' : 1, 'b': 2}
```

dictionary

- 딕셔너리 생성

```
>>> phone_book={'신사임당': '010-1234-5678', '홍길동': '010-4532-4532'}
>>> phone_book
{'신사임당': '010-1234-5678', '홍길동': '010-4532-4532'}
```

- 딕셔너리 키-값 추가

```
>>> phone_book['이순신']= '010-0987-6543'
>>> phone_book
{'신사임당': '010-1234-5678', '홍길동': '010-4532-4532', '이순신': '010-0987-
6543'}
```

dictionary

- 딕셔너리 키로 검색

```
>>> phone_book={'신사임당': '010-1234-5678', '홍길동': '010-4532-4532',
'이순신': '010-0987-6543'}
>>> phone_book['이순신']
'010-0987-6543'
>>> '이순신' in phone_book
True
>>> '강감찬' in phone_book
False
>>> phone_book.get('이순신')
'010-0987-6543'
```

dictionary

- keys() 메서드 : 딕셔너리의 모든 keys 출력

```
>>> phone_book.keys()
dict_keys(['신사임당', '홍길동', 이순신'])
```

- values() 메서드 : 딕셔너리의 모든 values 출력

```
>>> phone_book.values()
dict_values(['010-1234-5678', '010-4532-4532', '010-0987-6543'])
```

dictionary

- 딕셔너리 항목 삭제

```
>>> del phone_book['홍길동']
>>> phone_book
{'신사임당': '010-1234-5678', '이순신': '010-0987-6543'}
>>> phone_book.clear() #모든 항목 삭제
>>> phone_book
{}
```

dictionary

```
>>> >>> phone_book={'신사임당': '010-1234-5678', '홍길동': '010-4532-4532',
'이순신': '010-0987-6543'}
>>> for i in phone_book:
  print(i)

신사임당
홍길동
이순신
>>> for i in phone_book:
  print(i)
      else:
         print('end')
신사임당
홍길동
이순신
end
```

dictionary

```
>>> for i in phone_book.keys():
  print(i)
신사임당
홍길동
이순신
>>> for i in phone_book.values():
  print(i)
010-1234-5678
010-4532-4532
010-0987-6543
>>> for i in phone_book.items():
  print(i)
('신사임당', '010-1234-5678')
('홍길동', '010-4532-4532')
('이순신', '010-0987-6543')
```

```
>>> for i, j in phone_book.items():
  print(i, j)
신사임당 010-1234-5678
홍길동 010-4532-4532
이순신 010-0987-6543
```

dictionary

```
>>> dishes = {'eggs': 2, 'sausage': 1, 'bacon': 1, 'spam': 500}

>>> keys = dishes.keys()
>>> values = dishes.values()

>>> list(keys)
['eggs', 'bacon', 'sausage', 'spam']

>>> list(values)
[2, 1, 1, 500]
```

```
n = 0
for val in values:
    n += val
print(n)
```

dictionary

Operation	Result
len(d)	Return the number of items in the dictionary *d*.
d[key]	Return the item of *d* with key *key*.
d[key] = value	Set d[key] to *value*.
del d[key]	Remove d[key] from *d*.
key in d	Return True if *d* has a key *key*, else False.
key not in d	Equivalent to not key in d.
clear()	Remove all items from the dictionary.
copy()	Return a shallow copy of the dictionary.
items()	Return a new view of the dictionary's items ((key, value) pairs).
keys()	Return a new view of the dictionary's keys.
pop(*key*)	If *key* is in the dictionary, remove it and return its value, else return *default*.
get(*key*)	Return the value for *key* if *key* is in the dictionary, else *default*.
popitem()	Remove and return an arbitrary (key, value) pair from the dictionary.
setdefault(*key*)	If *key* is in the dictionary, return its value. If not, insert *key* with a value of *default* and return *default*.
update([*other*])	Update the dictionary with the key/value pairs from *other*, overwriting existing keys. ex. d.update(red=1, blue=2).
values()	Return a new view of the dictionary's values.

편의점 재고관리 프로그램

- **편의점 아이템**
  - "커피": 7
  - "펜": 3
  - "종이컵": 2
  - "우유": 1
  - "콜라": 4
  - "책": 5

```
--------------------메뉴 ------------
1. 재고확인     2. 판매     3. 종료
-----------------------------------
메뉴를 선택하세요: 1
재고 확인을 위한 물건명을 입력하세요. 커피
커피는 7개 남았습니다.

--------------------메뉴 ------------
1. 재고확인     2. 판매     3. 종료
-----------------------------------
메뉴를 선택하세요: 2
판매한 물건명을 입력하세요. 콜라
콜라의 판매 개수를 입력하세요. 2
콜라의 재고는 2개입니다.

--------------------메뉴 ------------
1. 재고확인     2. 판매     3. 종료
-----------------------------------
3
>>>
```

한영사전 프로그램

```
------------------메뉴 -----------------
1. 단어 추가    2. 사전 보기    3. 종료
----------------------------------------
메뉴를 선택하세요 : 1
추가할 한글 단어를 입력하세요: 사과
추가할 영어 단어를 입력하세요: apple

메뉴를 선택하세요 : 1
추가할 한글 단어를 입력하세요: 기차
추가할 영어 단어를 입력하세요: train

메뉴를 선택하세요 : 2
찾고자 하는 단어를 입력하세요: 기차
train

메뉴를 선택하세요 : 3
프로그램을 종료합니다.
>>>
```

set

set

- **집합 (set)**
  - set은 순서가 없으며, 수정이 가능하고, 값이 중복되지 않음

```
>>> a={1,2,3,4,5}
>>> a
{1,2,3,4,5}
```

  - 중복된

```
>>> b=set([1,1,4,2,2,6,6,6,6,7,8,10,4,2,8,6,3])
>>> b
{1,2,3,4,6,7,8,10}
```

set

- set 관련 자주 사용되는 메서드

```
>>> a={1,2,3,4,5}
>>> a
{1,2,3,4,5}
>>> a.add(6)  #아이템을 추가한다.
>>> a.remove(6) #아이템을 제거한다.
>>> a.pop() #가장 먼저 있는 아이템을 제거한다.
>>> a.clear() #아이템을 모두 제거한다.
```

set

- set은 집합이므로, 합집합, 차집합, 교집합, 대칭차집합을 표현할 수 있음

```
>>> a={1,4,7}
>>> b={1,3,6,8,10}
>>> a | b   #합집합
{1, 3, 4, 6, 7, 8, 10}
>>> a - b   #차집합
{4, 7}
>>> a & b   #교집합
{1}
>>> a ^ b   #대칭차집합    #(합집합)-(교집합)
{3, 4, 6, 7, 8, 10}
```

set

Operation	Result
s.add(item)	Add item to s (mutates s)
s.clear()	Remove elements from s (mutates s)
s.copy()	Shallow copy
s.difference(s2)	Return set with elements from s and not s2
s.difference_update(s2)	Remove s2 items from s (mutates s)
s.discard(item)	Remove item from s (mutates s). No error on missing item
s.intersection(s2)	Return set with elements from both sets
s.intersection_update(s2)	Update s with members of s2 (mutates s)
s.isdisjoint(s2)	True if there is no intersection of these two sets
s.issubset(s2)	True if all elements of s are in s2
s.issuperset(s2)	True if all elements of s2 are in s
s.pop()	Remove arbitrary item from s (mutates s). KeyError on missing item
s.remove(item)	Remove item from s (mutates s). KeyError on missing item
s.symmetric_difference(s2)	Return set with elements only in one of the sets
s.symmetric_difference_update(s2)	Update s with elements only in one of the sets (mutates s)
s.union(s2)	Return all elements of both sets
s.update(s2)	Update s with all elements of both sets (mutates s)

- Container 타입 값 할당의 여러 가지 예
  - *는 나머지 값의 의미로 사용한다.

```
>>> a, *b, c = 1,2,3,4,5
>>> a
1
>>> b
[2,3,4]
>>> c
5
>>> a, b= {1,2}
>>> a
1
>>> {3,1,2,3,1}
{1,2,3}
```

```
>>> *a, b, c = "안녕하세요"
>>> a
['안','녕','하']
>>> b
['세']
>>> c
['요']
```

멤버십연산자

- in : 해당 요소가 있으면 True, 없으면 False를 반환함
- not in : 해당 요소가 없으면 True, 있으면 False를 반환함

```
>>> a='Hello'
>>> 'h' in a
True
>>> 'b' in a
False
>>> b={1,3,6,8,10}
>>> 1 in b
True
>>> c = [1,2,3,4,5]
>>> 3 in c
True
```

```
>>> d=[[4,6], '1', '2']
>>> [4] in d
False
>>> [4,6] in d
True
```

- 0이외의 값은 모두 True이다.

```
>>> if 2:
        print('True')
True
>>> if [2]:
        print('True')
True
```

```
>>> if [ ]:
        print('True')
    else:
        print('False')
False
```

```
>>> a=3

>>> if 1 < a < 5:
        print('True')
    else:
        print('False')
True
```

```
>>> a=-3
>>> b=3 if a>0 else 8
>>> a
-3
>>> b
8
```

비교연산자 is

- 비교연산자 is는 값 비교가 아닌 레퍼런스 비교를 수행함

```
>>> 1 is 3
False
>>> a= '1'
>>> b= 1
>>> a is b
False
>>> a= 10
>>> b= 10
>>> a is b
True
>>> id(a), id(b)
(1516360656, 1516360656)
```

```
>>> a= -5
>>> b= -5
>>> a is b
True
>>> a= -6
>>> b= -6
>>> a is b
False
>>> a= 256
>>> b= 256
>>> a is b
True
>>> a= 257
>>> b= 257
>>> a is b
False
```

파이썬에서는 -5~256까지는 메모리에 캐시하여 동일한 주소를 참조하도록 하고 있다.

Thank you!

Beyond The Engine of Korea

HANYANG UNIVERSITY

한양대학교
HANYANG UNIVERSITY