

복습

- 관련 연산자 ... string에서 사용하는 연산 방법과 동일

연산자	사용 형태	의미	예	우선 순위
+	list + list	두 list 연결 시키기	[1, 2, 3] + ['a', 'b', 'c'] → [1, 2, 3, 'a', 'b', 'c']	2
*	list * 숫자	list를 숫자 만큼 반복 연결	[1, 2] * 4 → [1, 2, 1, 2, 1, 2, 1, 2]	1

- 곱셈(*) 시 숫자
 - 정수형만 가능
 - 0이나 음수일 경우 결과: [] (아무런 문자가 없는 비어있는 list)

- list 인덱싱(indexing)

- list의 item 하나를 선택
- 인덱싱
 - 왼쪽부터 0, 1, 2, ... 로 증가
 - 오른쪽에서 -1, -2, -3, ...로 감소

0	1	2	3	4
['a',	'b',	'c',	'd',	'e']
-5	-4	-3	-2	-1

```
In [ ] : a = [1, 2, 3]
          a[0]
```

```
Out [ ] : 1
```

```
In [ ] : a[1]
```

```
Out [ ] : 2
```

```
In [ ] : a[2]
```

```
Out [ ] : 3
```

```
In [ ] : a[-1]
```

```
Out [ ] : 3
```

```
In [ ] : a[-2]
```

```
Out [ ] : 2
```

```
In [ ] : a[-3]
```

```
Out [ ] : 1
```

- list 슬라이싱(slicing)
 - list의 일부분을 잘라 냄
 - 결과는 list ... 부분 집합과 비슷
 - 사용 방법: **변수**[시작:끝:증감]
 - 시작: 범위의 시작, 생략 시 0
 - 끝: 범위의 끝, 생략 시 list의 크기
 - 끝 인덱스는 미포함(직전 값까지)
 - 증감: 자료를 취하는 간격, 생략 시 1

```
In [ ] : a = [0, 1, 2, 3, 4, 5]
          a[0:2]
```

```
Out [ ] : [0, 1]
```

```
In [ ] : a[2:]
```

```
Out [ ] : [2, 3, 4, 5]
```

```
In [ ] : a[:4:2]
```

```
Out [ ] : [0, 2]
```

```
In [ ] : a[-4:-2]
```

```
Out [ ] : [2, 3]
```

```
In [ ] : a[-3:]
```

```
Out [ ] : [3, 4, 5]
```

```
In [ ] : a[:-3]
```

```
Out [ ] : [0, 1, 2]
```

- 리스트 생성

- 리스트 생성 관련 함수: list(), split(), range(끝), range(시작, 끝), range(시작, 끝, 스텝)

```
list('12345') → ['1', '2', '3', '4', '5']
```

```
a = '1 2 3 4 5'  
b = a.split() → b = ['1', '2', '3', '4', '5']
```

```
a = '1:2:3:4:5'  
b = a.split(':') → b = ['1', '2', '3', '4', '5']
```

```
list(range(4)) → [0, 1, 2, 3]  
list(range(3, 5)) → [3, 4]  
list(range(2, 11, 2)) → [2, 4, 6, 8, 10]  
list(range(9, 1, -2)) → [9, 7, 5, 3]
```

• 리스트 수정

사용 방법	의미	예시(a = [1, 2, 3]일 때)
리스트.append(요소)	리스트의 마지막에 요소를 추가	a.append(4) → a = [1, 2, 3, 4]
리스트.extend(리스트2)	리스트의 마지막에 리스트2를 추가	a.extend([4, 5]) → a = [1, 2, 3, 4, 5]
리스트.insert(index, 요소)	리스트의 index 위치에 요소를 추가	a.insert(1, 4) → a = [1, 4, 2, 3]
del 리스트[index]	리스트의 index에 위치한 요소를 삭제	del a[1] → a = [1, 3]
리스트.remove(요소)	리스트에서 첫 번째로 나오는 요소를 삭제	a.remove(1) → a = [2, 3]

- Dictionary (사전)
 - Key와 value 쌍들의 모음, {} 사용
 - 예: {'name':'gdhong', 'phone':'0222200001', 'addr':['Seoul', 'Wangsimni']}
 - Key: 중복 X, list형 불가(변경될 수 있는 데이터형 금지)
 - Value: 숫자, 문자열, list, dictionary 등 대부분의 자료형 가능

```
>>> a = {1:'a', 2:'b', 'three':'c'}
>>> a[1]
'a'
>>> a[2]
'b'
>>> a['three']
'c'
```

```
>>> a = {1:'a', 2:'b', 'three':'c'}
>>> a.get(1)
'a'
>>> a.get(2)
'b'
>>> a.get('three')
'c'
```

- Dictionary 요소 수정, 추가, 삭제
 - 수정: 기존에 있는 쌍에서 value만 수정
 - 추가: 기존에 없는 key에 대한 value를 대입
 - 삭제: `del dictionary[키]`
 - 모두 삭제: `clear()`

```
>>> a = {1:'a', 2:'b', 'three':'c'}
>>> a[1] = 'abc'
>>> a
{1: 'abc', 2: 'b', 'three': 'c'}
>>> a[4] = 'd'
>>> a
{1: 'abc', 2: 'b', 'three': 'c', 4: 'd'}
>>> del a['three']
>>> a
{1: 'abc', 2: 'b', 4: 'd'}
>>> a.clear()
>>> a
{}
```


- 문제
 - 학번을 입력 받아 역순으로 문자열을 출력하시오.
 - 문자열 슬라이싱 이용

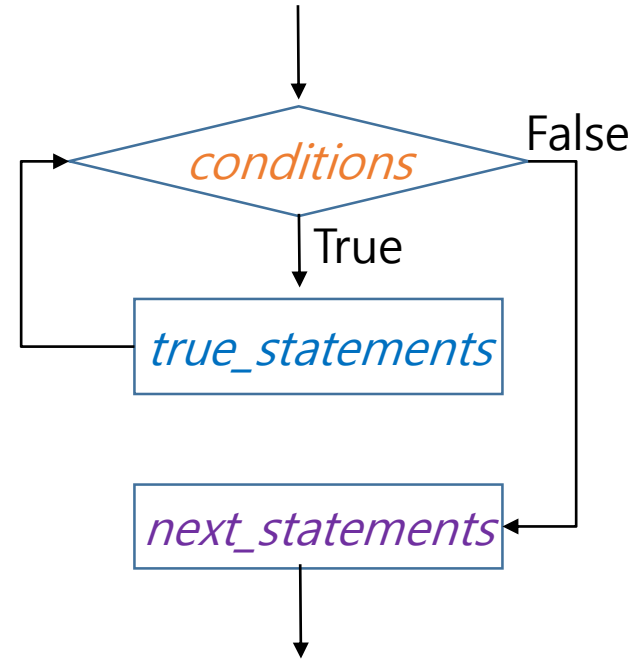
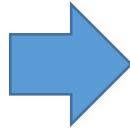
Input your student ID: 123456789
987654321

- 문제풀이
 - n 을 입력 받아 $n*1, n*2, \dots, n*9$ 를 요소로 갖는 list를 출력하시오.
 - range() 함수 사용

```
RESTART: C:/Users/user/Dropbox/SW/L  
Input a number: 3  
[3, 6, 9, 12, 15, 18, 21, 24, 27]  
>>>  
RESTART: C:/Users/user/Dropbox/SW/L  
Input a number: 5  
[5, 10, 15, 20, 25, 30, 35, 40, 45]
```

- 반복문 - while

```
while conditions:  
    true_statements  
    next_statements
```



- 사용 방법
 - break 사용하기

```
while conditions1:           # 1: conditions1의 결과가 거짓이면 7로
    true_statements1         # 2:
    if conditions2:          # 3: conditions2의 결과가 거짓이면 6으로
        if_statements        # 4:
        break                # 5: 7으로 (while문을 벗어남)
    true_statements2         # 6:
next_statements              # 7:
```

- continue 사용하기

```
while conditions1:           # 1: conditions1의 결과가 거짓이면 7로
    true_statements1         # 2:
    if conditions2:          # 3: conditions2의 결과가 거짓이면 6으로
        if_statements        # 4:
        continue             # 5: 1로 (condition1 검사 수행)
    true_statements2         # 6:
next_statements              # 7:
```

- 문제

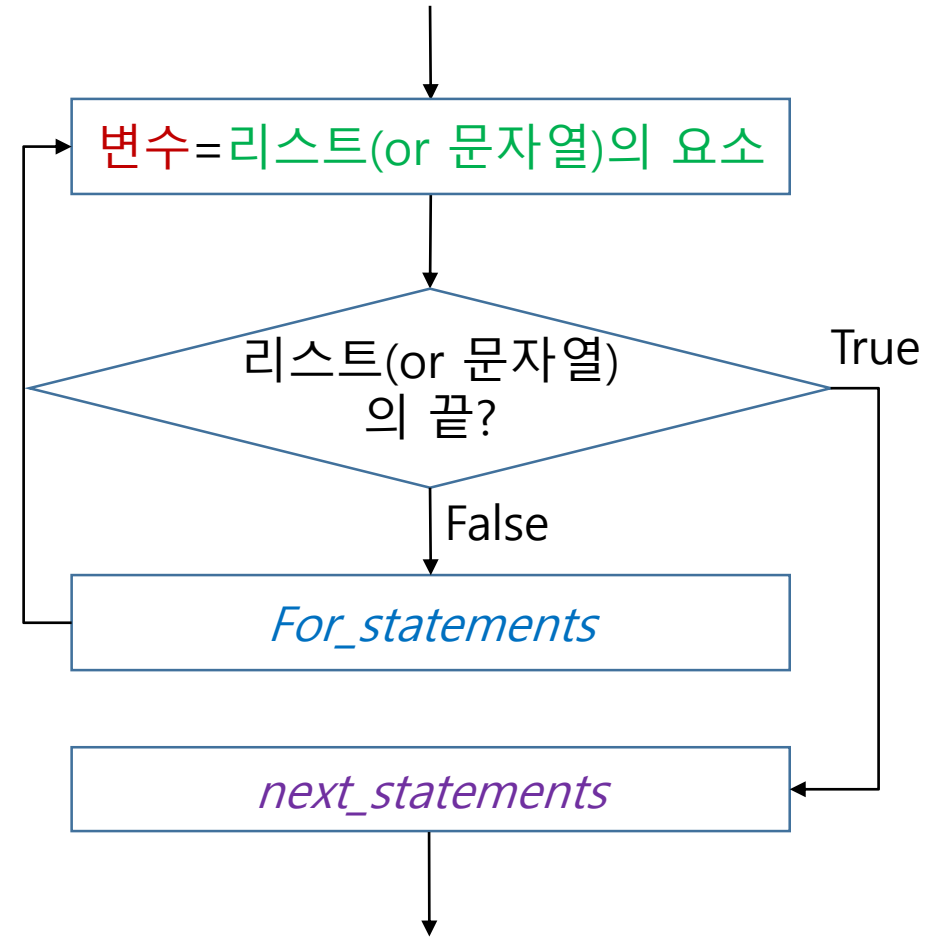
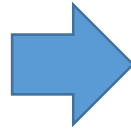
- n 을 입력받아 1부터 n 까지의 합을 구하여 출력하는 프로그램을 작성하십시오.
 - while문을 사용

```
===== RESTART: C:/Python36-
정수 입력 : 10
1부터 10까지의 합 : 55
>>>
===== RESTART: C:/Python36-
정수 입력 : 100
1부터 100까지의 합 : 5050
>>> |
```

- 반복문 – for

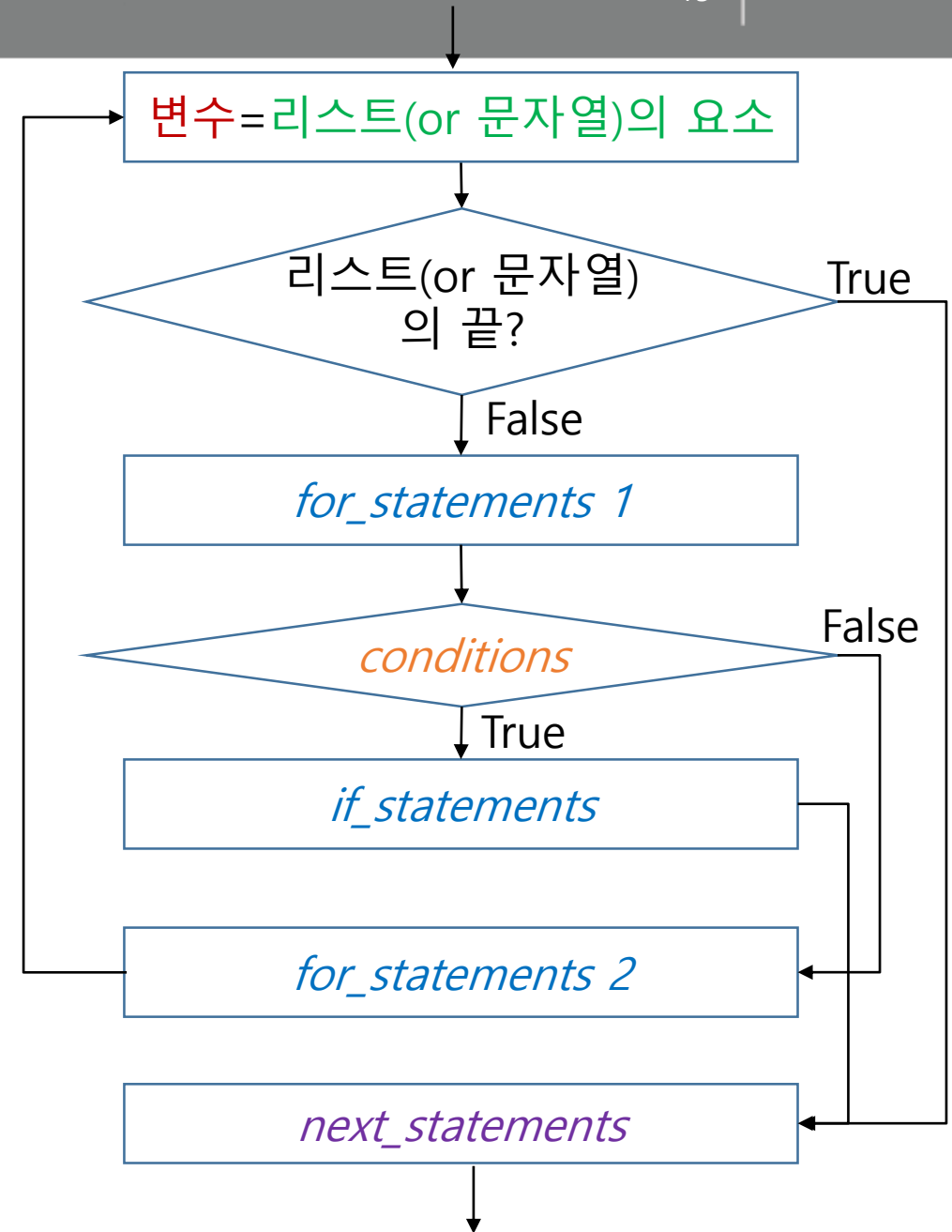
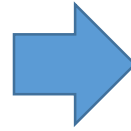
```
for 변수 in range(시작, 끝, 증감) :  
    for_statements  
next_statements
```

```
for 변수 in 리스트(or 문자열):  
    For_statements  
Next_statements
```



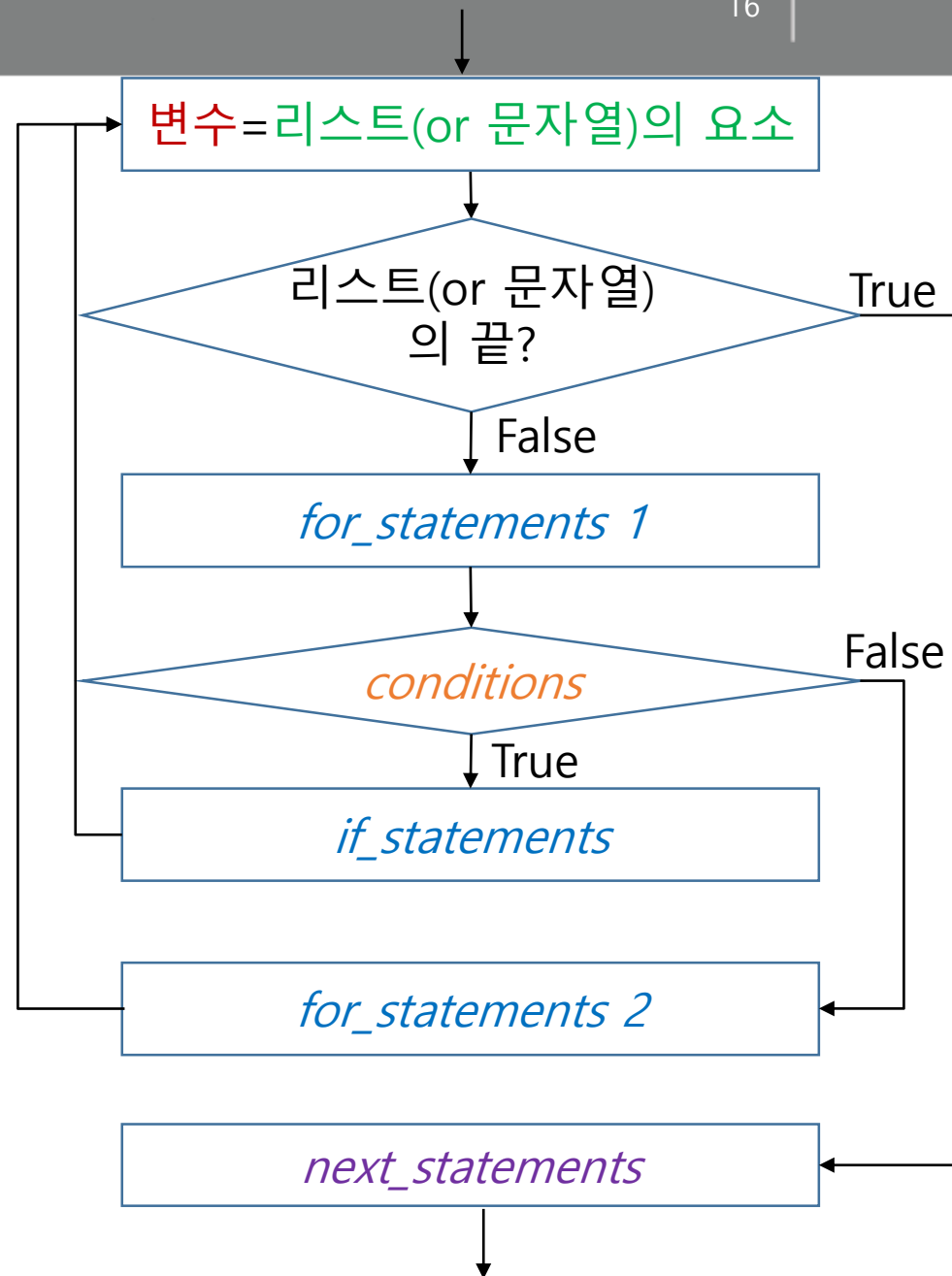
• 반복문 – for

```
for 변수 in 리스트(or 문자열):
    for_statements1
    if conditions:
        if_statements
        break
    for_statements2
next_statements
```



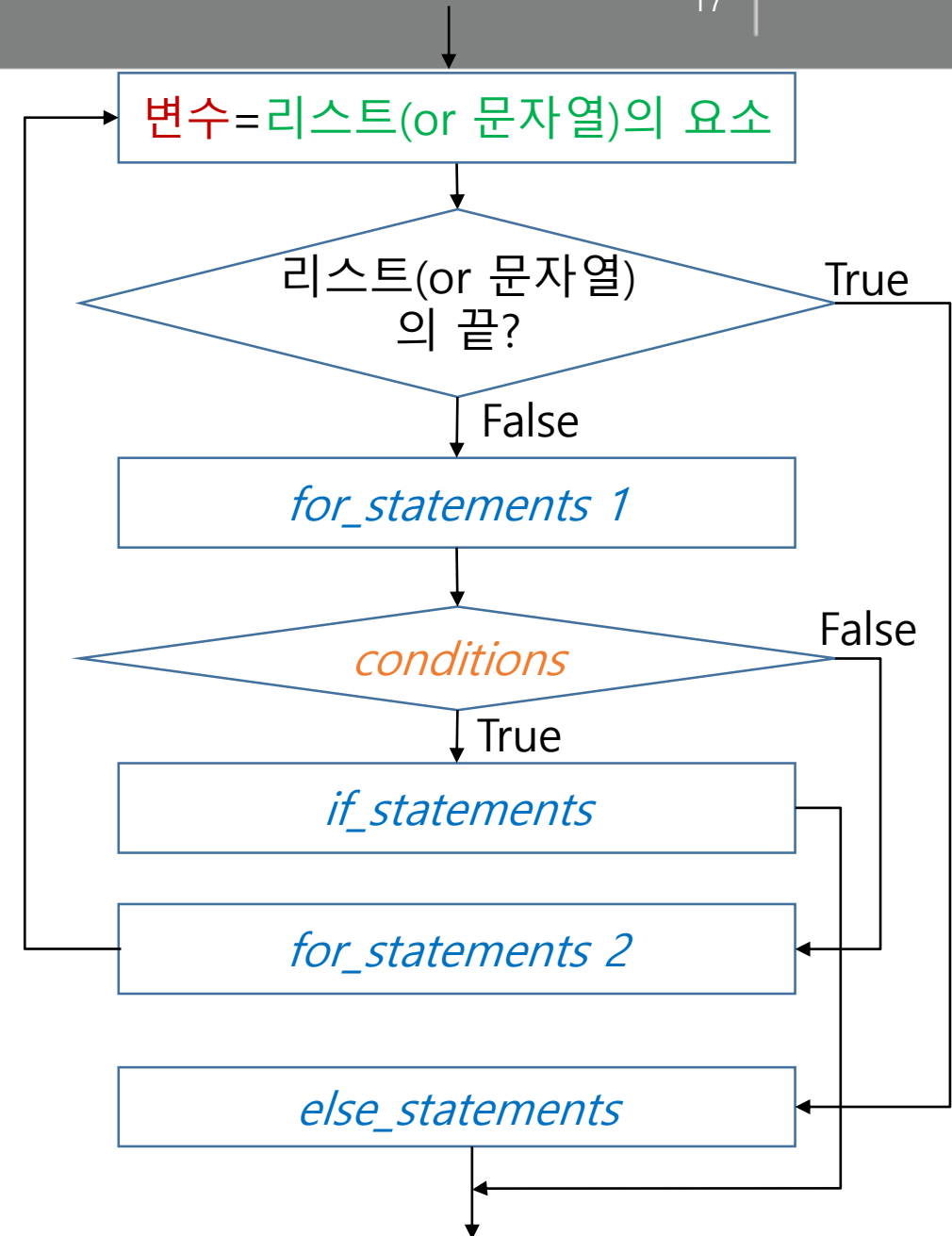
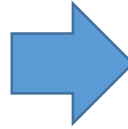
• 반복문 – for

```
for 변수 in 리스트(or 문자열):
    for_statements1
    if conditions:
        if_statements
        continue
    for_statements2
next_statements
```



- 반복문 – for

```
for 변수 in 리스트(or 문자열):  
    for_statements1  
    if conditions:  
        if_statements  
        break  
    for_statements2  
else:  
    else_statements
```



- 문제

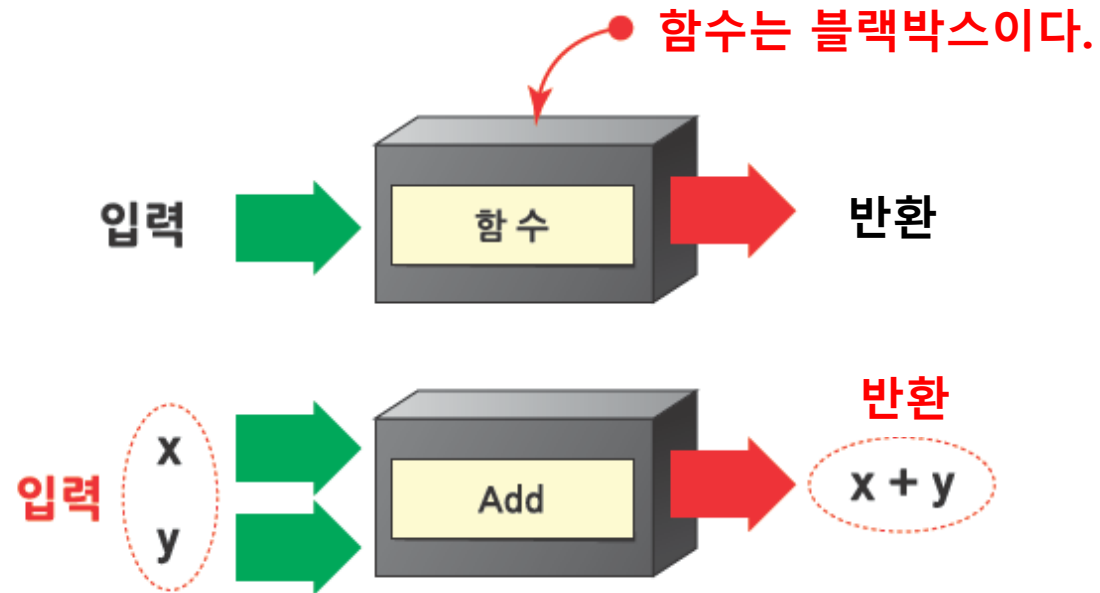
- n 을 입력받아 1부터 n 까지의 합을 구하여 출력하는 프로그램을 작성하십시오.
 - for문을 사용

```
===== RESTART: C:/Python36-
정수 입력: 10
1부터 10까지의 합: 55
>>>
===== RESTART: C:/Python36-
정수 입력: 100
1부터 100까지의 합: 5050
>>> |
```

함수

- 함수

- 프로그램에서 자주 사용되는 코드를 따로 만들어 두고 필요할 때마다 불러서(호출해서) 사용하는 기능



- 지금까지 프로그래밍에서 사용한 함수에는 어떤 것들이 있었을까?

- 입력함수

- `input('Input the number :')`
- `input()`

- 출력함수

- `print('Total number' + total)`
- `print()`

- 형변환함수

- `int('12')`
- `str(31)`
- `float(15)`

- 랜덤함수

- `random.randint(1,20)`

내장함수:
프로그래밍에서 자주 사용하는 기능을 미리 만들어놓은 함수



- 필요한 기능이 내장함수에 없으면 어떻게 하지?
 - 예:
 - “섭씨온도를 입력하면 화씨온도를 계산해주는 기능이 있으면 좋겠어”
 - “dino를 클릭할 때마다 공룡그림을 출력해주는 기능이 필요해”

```
def process_c_f(c):  
    f=9.0/5.0*c+32.0  
    print (f)
```

```
def dino( ):  
    print ('  
    <img alt="A green dashed outline of a dinosaur, specifically a T-Rex, standing and facing left." data-bbox="175 695 365 910"/>  
    ...')
```

사용자정의함수:
프로그래머가 필요에 의해서 만든
함수

- 함수를 사용하지 않은 예

```
num = 1
total = 0
while num <= 10 :
    total = total + num
    num = num + 1
print ('1부터 10까지의 합은' + str(total) + '입니다.')
```

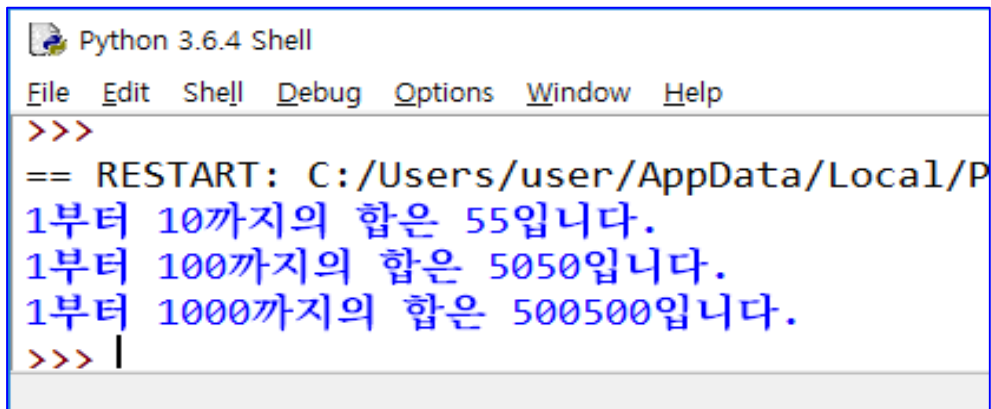
```
num = 1
total = 0
while num <= 100 :
    total = total + num
    num = num + 1
print ('1부터 100까지의 합은' + str(total) + '입니다.')
```

```
num = 1
total = 0
while num <= 1000 :
    total = total + num
    num = num + 1
print ('1부터 1000까지의 합은' + str(total) + '입니다.')
```

- 함수를 사용한 예

```
def summation(y) :  
    num = 1  
    total = 0  
    while num <= y :  
        total = total + num  
        num = num + 1  
    print ('1부터 ' + str(y) + '까지의 합은 ' + str(total) + '입니다.')
```

```
summation(10)  
summation(100)  
summation(1000)
```



Python 3.6.4 Shell

File Edit Shell Debug Options Window Help

```
>>>  
== RESTART: C:/Users/user/AppData/Local/P  
1부터 10까지의 합은 55입니다.  
1부터 100까지의 합은 5050입니다.  
1부터 1000까지의 합은 500500입니다.  
>>> |
```


- 코드의 간결성
 - 코드가 중복되지 않고 간결해진다.
- 코드의 재사용성
 - 한 번 작성해둔 코드를 여러 번 사용하므로 코드를 재사용할 수 있다.
- 프로그램의 모듈화
 - 기능별로 함수를 작성하므로 프로그램 모듈화가 증대된다.
- 코드 수정의 용이성
 - 프로그램 기능을 함수로 나누어 묶기 때문에 코드 수정이 쉽다.

- 함수는 어떻게 만들지?
 - 함수 정의 파트와 함수 호출 파트로 이루어진다.
 - 함수 정의 파트가 함수 호출 파트보다 먼저 작성되어 있어야 한다.
 - 함수 호출 파트(외부) 에서 함수 정의 파트(내부)로 값을 전달할 때 전달인자를 사용한다.
 - 함수 정의 파트에서는 값을 받을 때 매개변수로 받는다.
 - 전달인자나 매개변수를 파라미터라고도 부름
 - 함수 정의 파트에서 함수 호출 파트로 값을 내 보낼 때에는 반환(return)을 사용한다.
 - 전달인자, 매개변수, 반환이 없는 함수도 있다.
 - 함수안에서 사용하는 지역변수와 함수 밖에서 사용하는 전역변수가 있다.

- 함수 정의와 호출 방법
 - 함수 정의 파트와 함수 호출 파트로 이루어진다.
 - 함수 정의 파트가 함수 호출 파트보다 먼저 작성되어 있어야 한다.

함수 정의 방법

keyword 매개변수

```
def bird():
```

↑
함수명

↑
colon

함수 호출 방법

함수명() 을 적는다.

함수 정의와 호출의 예

```
def bird():  
    print (''  
        {.,.}  
        |_)_  
        _"_"  
    ... )  
bird()
```

들여쓰기

함수 정의 (내부)

함수 호출 (외부)

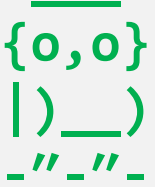

- 함수 정의 위치
 - 함수 정의 파트가 함수 호출 파트보다 먼저 작성되어 있어야 한다.


```
sayGoodBye ()
```

```
def sayGoodBye () :  
    print 'Good bye! '
```

```
Traceback (most recent call last):  
  File "<pyshell#3>", line 2, in <module>  
    sayGoodBye ()  
NameError: name 'sayGoodBye' is not defined
```

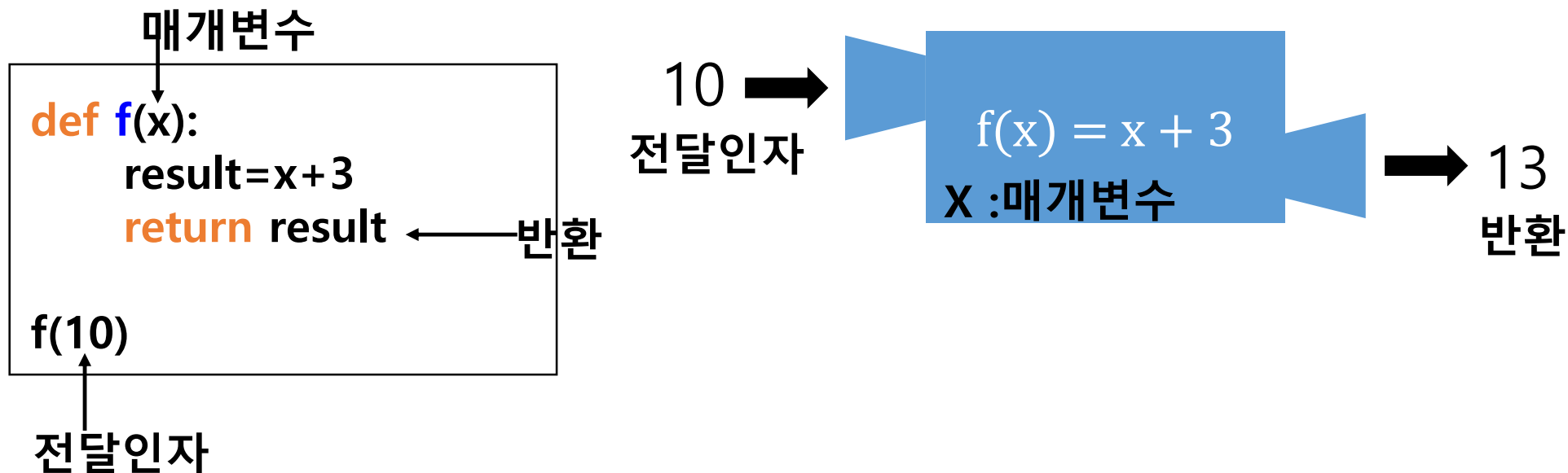
- 실습해 보기

```
def bird1():  
    print (''  
          
        '' )  
def bird2():  
    print (''  
          
        '' )
```

```
def bird3():  
    print (''  
          
        '' )  
  
    count = 1  
    while count <= 10:  
        bird1()  
        bird2()  
        bird3()  
        count = count + 1
```

- 함수를 10번
호출해 보기

- 매개변수 & 전달인자 & 반환
 - 함수 호출 파트(외부) 에서 함수 정의 파트(내부)로 값을 전달할 때 **전달인자**를 사용한다.
 - 함수 정의 파트에서는 값을 받을 때 **매개변수**로 받는다.
 - 함수 정의 파트에서 함수 호출 파트로 값을 내 보낼 때에는 **반환(return)**을 사용한다.



- Quiz

```
def f_a(x, y, z):  
    result=x+y+z  
    return result
```

```
result = f_a(10, 20, 30)  
print (result)
```

```
def f_b( ):  
    for count in range(100):  
        result+=count  
    print ('result:', result)
```

```
result = f_b( )  
print (result)
```

전달인자:

매개변수:

반환:

전달인자:

매개변수:

반환:

- None은 타입 자체가 None이다.

```
>>> None == False
False
>>> type(None)
NoneType
```


- Quiz

```
def f_a( ):
    result=10+20+30
    return result
```

```
result = f a( )
print ('result:', result)
```

```
def f_b( ):
    for count in range(100):
        result+=count
    return result
```

```
print ('result:', f_b( ) )
```

전달인자:

매개변수:

반환:

전달인자:

매개변수:

반환:

- 지역변수
 - 함수 정의 파트(내부)에서만 유효한 변수
 - 함수가 호출되었을 때 생성되며 함수를 벗어날 때 사라짐
- 전역변수
 - 프로그램 전체에서 유효한 변수
 - 프로그램이 시작될 때 생성되며, 프로그램이 끝날 때 사라짐

전역변수와 지역변수는 변수 이름이 같아도 각각 별개의 변수입니다.



```
def add ():  
    number = 5  
    number = number + 3  
    print (number)
```

지역변수
number

```
number = 5  
add()  
print (number)
```

전역변수
number

• 실행결과

```
8  
5  
>>>
```

- 지역변수, 전역변수의 사용
 - 전역변수를 함수 내부에서 수정 또는 선언하고자 할 때에는 **global** 키워드를 사용함

```
def add():  
    global number  
    number = number + 3  
    print(number)
```

전역변수
number

```
number = 5  
add()  
print(number)
```

전역변수
number

• 실행결과

```
8  
8  
>>>
```

- 지역변수, 전역변수의 사용



```
number = 5
def add ():
    global number
    number = number + 3

add()
print (number)
```



```
number = 5
def add ():
    number = number + 3

add()
print (number)
```

- 지역변수, 전역변수의 사용

```
number = 5  
def add():  
    print (number)  
  
add()
```

- Tip

전역변수의 값을 변경하는 것 없이, 단순히 전역변수의 값을 참조만 하는 경우에는 함수 내부에서 global로 선언하지 않아도 함수에 정의된 지역변수가 없다면 전역변수를 찾아 참조한다.

에러가 나올까요?
안나올까요?



- Quiz

```
def spam(myName):  
    print ('Hello, ' + myName)  
    myName = 'Waffles'  
    print('Your new name is ' + myName)  
    return myName  
  
myName = 'Albert'  
spam(myName)  
print ('Howdy, ' + myName)
```

- Intro 출력하기
 - `print()`를 이용하여 출력

```
print ('You are in a land full of dragons.')
```

```
print ('In front of you, you see two caves. ')
```

```
print ('In one cave, the dragon is friendly, ')
```

```
print (' and will share his treasure with you. ')
```

```
print ('The other dragon is greedy and hungry, ')
```

```
print ('and will eat you on sight. ')
```

- 으سس~효과를 주기
 - 맨 윗줄에 **import time**을 추가
 - 시간 지연을 넣고자 하는 곳에 **time.sleep(1)**을 추가

```
import time
print ('You are in a land full of dragons.')
time.sleep(1)
print ('In front of you, you see two caves. ')
time.sleep(1)
print ('In one cave, the dragon is friendly, ')
time.sleep(1)
print (' and will share his treasure with you. ')
time.sleep(1)
print ('The other dragon is greedy and hungry, ')
time.sleep(1)
print ('and will eat you on sight. ')
```


- 동굴번호 입력받기

- 동굴번호를 입력받아 저장할 변수 : cave

```
cave = input ('Which cave will you go into? (1 or 2) ')
```

- 입력받은 동굴번호가 1도 아니고, 2도 아니면 계속 입력해달라고 반복해야 함

- 소스코드 작성

```
while cave != '1' and cave != '2':  
    cave = input ('Which cave will you go into? (1 or 2) ')
```

- 동굴번호 입력 후 메시지
 - 메시지를 출력하는 소스코드를 작성하세요.
 - 시간 지연을 넣고자 하는 곳에 **time.sleep(1)**을 추가하세요.

- 소스코드 작성

You approach the cave...
It is dark and spooky...
A large dragon jumps out in front of you!
He opens his jaws and...

- 랜덤값 발생

- ❖ 컴퓨터도 랜덤값 발생, 랜덤값을 저장할 변수이름 : friendlycave

- 소스코드 작성

```
#랜덤모듈 import
#랜덤값 발생
if friendlycave == int(cave):
    print ('Gives you his treasure!')
else :
    print ('Gobbles you down in one bite!')
```

- 게임 계속 여부 질문 넣기

```
playagain = input ('Do you want to play again?(yes or no)')
```

- 'yes' or 'y'이면 게임이 계속되도록 반복문을 넣어보세요.
 - 반복해야 할 문장은? 문장 전체

• 소스코드 작성

```
.....  
:   
print ('You are in a land full of dragons.')  
time.sleep(1)  
print ('In front of you, you see two caves. ' )  
.....  
playagain = input ('Do you want to play again?(yes or no)')
```

- 함수로 바꾸어보기
 - displayIntro() 함수 : 게임 시작에 나오는 인트로 메시지 부분
 - chooseCave() 함수 : 동굴번호 선택 코드 부분
 - checkCave() 함수 : 동굴번호와 랜덤번호 비교하는 코드 부분

- 다음과 같이 두수를 입력 받아 계산하여 출력하는 프로그램을 addition(x,y) 함수를 정의하여 작성하시오.

```
=====
첫번째 정수를 입력하세요:
```

```
3
```

```
두번째 정수를 입력하세요:
```

```
6
```

```
addition result 9
```

- subtraction(x,y) 함수를 추가하여 아래와 같이 출력되도록 작성하시오.

```
===== RESTART:
```

```
첫번째 정수를 입력하세요:
```

```
6
```

```
두번째 정수를 입력하세요:
```

```
3
```

```
subtraction result: 3
```

```
>>>
```

- multiplication(x,y)함수를 추가하여 아래와 같이 출력되는 프로그램을 작성하시오.

첫번째 정수를 입력하세요:

10

두번째 정수를 입력하세요:

20

multiplication result: 200

- division(x,y)함수 추가하여 아래와 같이 출력되는 프로그램을 작성하시오.

첫번째 정수를 입력하세요:

10

두번째 정수를 입력하세요:

20

division result: 0

- 연산자를 함께 입력 받도록 수정한 후, 연산자에 따라 필요한 함수를 호출하도록 프로그램을 수정하시오.

첫번째 정수를 입력하세요:

10

두번째 정수를 입력하세요:

5

연산자를 입력하세요 (+, -, *, /):

+

addition result : 15

>>>

===== RESTART: C:,

첫번째 정수를 입력하세요:

50

두번째 정수를 입력하세요:

10

연산자를 입력하세요 (+, -, *, /):

-

subtraction result: 40

- 연산자에 all을 추가하고 all을 입력 받았을 때 모든 결과의 합을 출력하도록 수정해보세요.(all이라는 연산자는 실제 없음!)

첫번째 정수를 입력하세요:

30

두번째 정수를 입력하세요:

10

연산자를 입력하세요 (+, -, *, /, all)

all

addition result : 40

subtraction result: 20

multiplication result: 300

division result: 3

모든 결과의 합은 363 입니다.

예외처리

- 오류 (error)
 - 구문 오류 (syntax error)
 - 프로그램 실행 전에 발생하는 오류
 - 런타임 오류 (runtime error) / 예외 (exception)
 - 프로그램 실행 중에 발생하는 오류

- 구문 오류

```
In [1]: print('예외를 발생시켜 봅시다.')
```

```
File "<ipython-input-1-36d4aac6f87c>", line 1
    print('예외를 발생시켜 봅시다.')
      ^
```

```
SyntaxError: EOL while scanning string literal
```

- 예외 / 런타임 오류
 - 실행 중 발생하는 오류

```
In [2]: print('예외를 발생시켜 봅시다.')  
listE[1]
```

예외를 발생시켜 봅시다.

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-2-722f373717e0> in <module>  
      1 print('예외를 발생시켜 봅시다.')  
----> 2 listE[1]
```

NameError: name 'listE' is not defined

```
In [3]: print('예외를 발생시켜 봅시다.')  
listE = [1, 2, 3, 4]  
listE[1]
```

예외를 발생시켜 봅시다.

Out[3]: 2

- 기본 예외 처리(조건문)
 - 정수 입력

```
In [4]: number = int(input('정수 입력: '))  
  
print('원의 반지름:', number)  
print('원의 둘레:', 2*3.14*number)  
print('원의 넓이:', 3.14*number**2)
```

```
정수 입력: 10  
원의 반지름: 10  
원의 둘레: 62.800000000000004  
원의 넓이: 314.0
```

- 기본 예외 처리(조건문)
 - 실수 입력

```
In [5]: number = int(input('정수 입력: '))

print('원의 반지름:', number)
print('원의 둘레:', 2*3.14*number)
print('원의 넓이:', 3.14*number**2)
```

정수 입력: 10.12

```
ValueError                                Traceback (most recent call last)
<ipython-input-5-eb3db3caf3c8> in <module>
----> 1 number = int(input('정수 입력: '))
      2
      3 print('원의 반지름:', number)
      4 print('원의 둘레:', 2*3.14*number)
      5 print('원의 넓이:', 3.14*number**2)
```

ValueError: invalid literal for int() with base 10: '10.12'

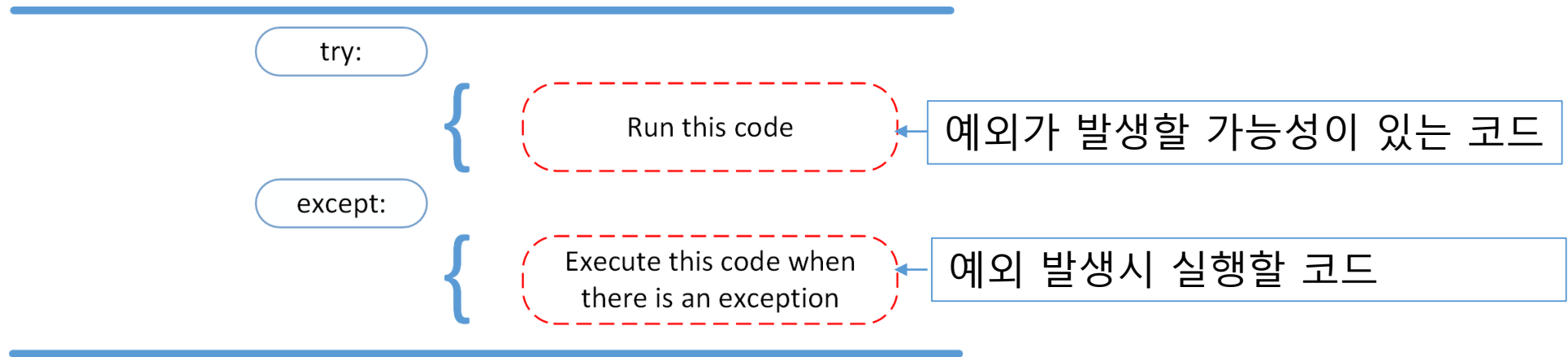
- 기본 예외 처리(조건문)
 - isdigit() 함수 : 숫자로만 구성된 글자인지 확인

```
In [7]: number = input('정수 입력: ')

if number.isdigit():
    number = int(number)
    print('원의 반지름:', number)
    print('원의 둘레:', 2*3.14*number)
    print('원의 넓이:', 3.14*number**2)
else:
    print('정수를 입력하지 않았습니다.')
```

정수 입력: 10.12
정수를 입력하지 않았습니다.

- try except 구문



- 모든 예외 상황을 예측하고, 조건문으로 만들기는 매우 힘든 작업
- 모든 예외 상황을 몰라도, 프로그램이 강제로 다운되는 상황을 피할 수 있음

- 위의 조건문을 바꾼 예

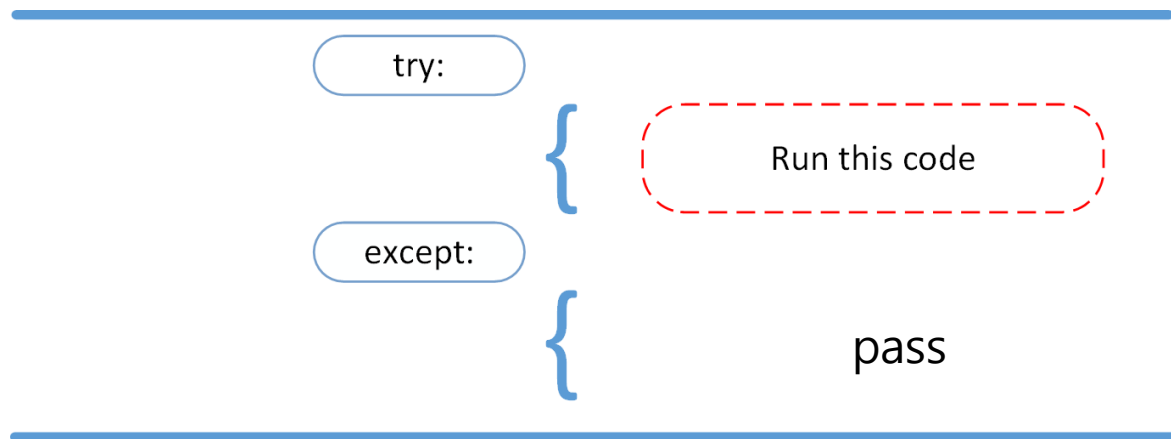
```
In [8]: try:
        number = int(input('정수 입력: '))
        print('원의 반지름:', number)
        print('원의 둘레:', 2*3.14*number)
        print('원의 넓이:', 3.14*number**2)
    except:
        print('어딘가 예외가 발생했습니다.')
```

← 예외가 발생할 가능성이 있는 코드

← 예외 발생시 실행할 코드

정수 입력: hi~
어딘가 예외가 발생했습니다.

- pass 키워드
 - 예외 발생시 처리해야 되지만, 중요한 부분이 아니면 except 에 아무것도 넣지 않을 수도 있음
- except 는 비워 놓으면 오류가 발생하므로 pass 키워드 사용



- 예

```
In [11]: listDigit = ['10', '20', '삼십', '40']

listTrueDigit = []
for i in listDigit:
    try:
        float(i)
        listTrueDigit.append(i)
    except:
        pass

print('{}는 원래 리스트 요소입니다.'.format(listDigit))
print('{}는 숫자로 이루어진 리스트 요소입니다.'.format(listTrueDigit))
```

['10', '20', '삼십', '40']는 원래 리스트 요소입니다.
['10', '20', '40']는 숫자로 이루어진 리스트 요소입니다.

- try except else
 - try except 구문 뒤에 else 는 예외가 발생하지 않으면 실행할 코드

try :

예외가 발생할 가능성이 있는 코드

except :

예외 발생시 실행할 코드

else :

예외가 발생하지 않으면 실행할 코드

- 예외가 발생할 가능성 있는 코드만 try
나머지는 else

• 예

```
In [12]: try:
          number = int(input('정수 입력: '))

          except:
              print('어딘가 예외가 발생했습니다.')

          else:
              print('원의 반지름:', number)
              print('원의 둘레:', 2*3.14*number)
              print('원의 넓이:', 3.14*number**2)
```

정수 입력: 10
원의 반지름: 10
원의 둘레: 62.800000000000004
원의 넓이: 314.0

정수 입력: hello
어딘가 예외가 발생했습니다.

- finally
 - 예외 처리에서 마지막에 사용
 - 무조건 실행 되는 코드

try :

예외가 발생할 가능성이 있는 코드

except :

예외 발생시 실행할 코드

else :

예외가 발생하지 않으면 실행할 코드

finally :

무조건 실행 되는 코드

• 예

```
In [15]: try:
          number = int(input('정수 입력: '))

          except:
              print('어딘가 예외가 발생했습니다.')

          else:
              print('원의 반지름:', number)
              print('원의 둘레:', 2*3.14*number)
              print('원의 넓이:', 3.14*number**2)

          finally:
              print('무조건 실행하자!')
```

정수 입력: 10
원의 반지름: 10
원의 둘레: 62.800000000000004
원의 넓이: 314.0
무조건 실행하자!

정수 입력: 10.12
어딘가 예외가 발생했습니다.
무조건 실행하자!

- 예외 객체
 - 예외 발생시 예외 정보 저장

try :

예외가 발생할 가능성이 있는 코드

except 예외 종류 as 변수이름 :

예외 발생시 실행할 코드

- Exception 은 모든 예외의 어머니

```
In [18]: try:
          number = int(input('정수 입력: '))
          print('원의 반지름:', number)
          print('원의 둘레:', 2*3.14*number)
          print('원의 넓이:', 3.14*number**2)

        except Exception as exception:
          print('type(excetion):', type(exception))
          print('exception:', exception)
```

정수 입력: hi

type(excetion): <class 'ValueError'>

exception: invalid literal for int() with base 10: 'hi'

- 예외 구분하기
 - if ~ elif 처럼 예외를 구분

try :

예외가 발생할 가능성이 있는 코드

except 예외 종류 A :

예외A 발생시 실행할 코드

except 예외 종류 B :

예외B 발생시 실행할 코드

except 예외 종류 C :

예외C 발생시 실행할 코드

• 예

```
In [19]: listNum = [1, 2, 3, 4, 5]

try:
    number = int(input('정수 입력: '))
    print(number, '번째 요소:', listNum[number])

except ValueError:
    print('정수를 입력해주세요.')

except IndexError:
    print('인덱스 범위를 벗어났어요')
```

정수 입력: 2
2 번째 요소: 3

정수 입력: hi
정수를 입력해주세요.

정수 입력: 10
인덱스 범위를 벗어났어요

- as 키워드 사용

```
In [26]: listNum = [1, 2, 3, 4, 5]

try:
    number = int(input('정수 입력: '))
    print(number, '번째 요소:', listNum[number])

except ValueError as exception:
    print('정수를 입력해주세요.')
    print('exception:', exception)

except IndexError as exception:
    print('인덱스 범위를 벗어났어요')
    print('exception:', exception)
```

```
정수 입력: hi
정수를 입력해주세요.
exception: invalid literal for int() with base 10: 'hi'
```

```
정수 입력: 10
인덱스 범위를 벗어났어요
exception: list index out of range
```

- 모든 예외 잡기
 - 마지막에 Exception 추가

```
In [28]: listNum = [1, 2, 3, 4, 5]

try:
    number = int(input('정수 입력: '))
    print(number, '번째 요소:', listNum[number])
    예외.예외발생()

except ValueError as exception:
    print('정수를 입력해주세요.')
    print('exception:', exception)

except IndexError as exception:
    print('인덱스 범위를 벗어났어요')
    print('exception:', exception)

except Exception as exception:
    print('예상하지 못한 예외입니다.')
    print(type(exception), exception)
```

정수 입력: 1

1 번째 요소: 2

예상하지 못한 예외입니다.

<class 'NameError'> name '예외' is not defined

- BMI 계산하기
 - 기본 코드

```
In [ ] : def getInt(prompt):  
         inNum = int(input(prompt))  
         return inNum  
  
         H = getInt('Height: ')  
         W = getInt('Weight: ')  
  
         BMI = W/(H/100)**2  
         print("BMI:", BMI)
```

- 정수가 아닌 수를 입력할 경우 예외 처리가 되도록 try~ except~ else 로 변경해 보자

- BMI 계산하기
 - int 함수에서 발생하는 오류 대처하기

```
In [ ] : def getInt(prompt):  
        while True:  
            try: inNum = int(input(prompt))  
            except ValueError as e: print(e)  
            else: return inNum  
  
        H = getInt('Height: ')  
        W = getInt('Weight: ')  
  
        BMI = W/(H/100)**2  
        print("BMI:", BMI)
```

- H = 0인 경우와 모든 예외에서 오류가 발생하지 않도록 코드를 수정하시오.

- 코드

In [] :

```
def getInt(prompt):  
    while True  
        try: inNum = int(input(prompt))  
        except ValueError as e: print(e)  
        else: return inNum  
  
H = getInt('Height: ')  
W = getInt('Weight: ')  
  
try: BMI = W/(H/100)**2  
except ZeroDivisionError as e: print('키 입력이 0입니다.')  
except Exception as e: pass  
else: print("BMI:", BMI)
```