파이썬 문법 기초
강의환경소개

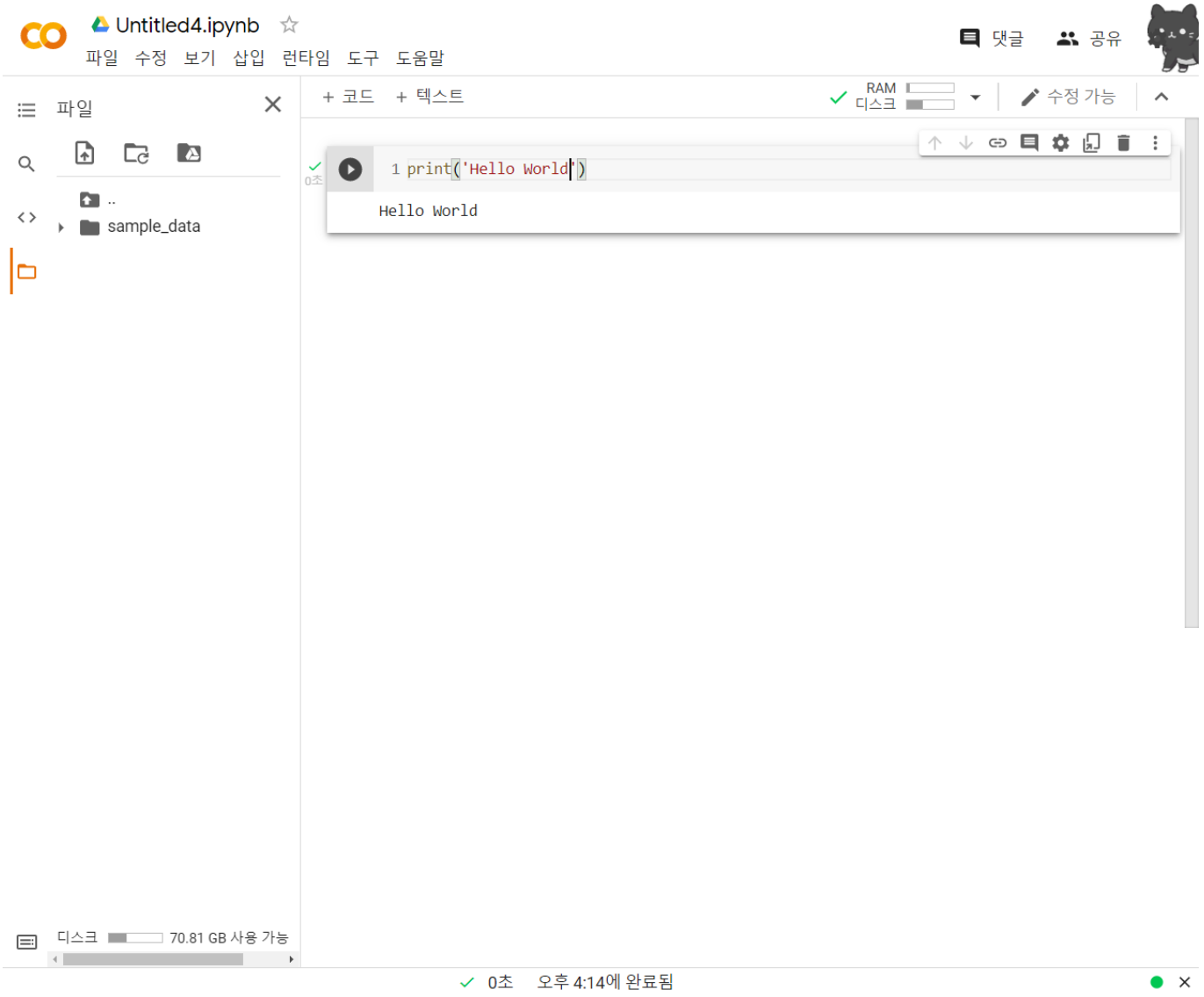
시스템경영공학부
이지환 교수

강의환경

- 강의환경 : Google Colaboratory (Colab)
  - 구글에서 데이터 과학 프로그래밍을 위해 제공한 클라우드 기반 코딩 환경
  - 무료
  - 로컬 컴퓨터에 별도 프로그램 설치 필요 없음

- 주소
  - https://colab.research.google.com
  - 구글 회원가입 필요

Colab 사용화면


Untitled4.ipynb
파일 수정 보기 삽입 런타임 도구 도움말
댓글
공유
파일
+ 코드
+ 텍스트
RAM
디스크
수정 가능
..
sample_data
1 print('Hello World')
Hello World
0초
디스크
70.81 GB 사용 가능
0초 오후 4:14에 완료됨

파이썬 문법 기초 (값, 변수, 연산자)

시스템경영공학부
이지환 교수

1. 값 (Value)

- 프로그램이 처리하는 기본단위
- 값의 종류
  - 숫자
  - 문자
  - 불리언

1. 값 (Value)

1-2. 숫자

- 정수형 숫자도 가능하다

```
[1]: 1
```

```
[1]: 1
```

- 음수도 가능하다

```
[2]: -2001
```

```
[2]: -2001
```

- 소숫점도 가능하다

```
[3]: 11.5
```

```
[3]: 11.5
```

1. 값 (Value)

1-2. 문자

- 숫자를 제외한 알파벳 문자를 바로 입력시 에러가 난다

```
[4]: x
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-6fcf9dfbd479> in <module>
----> 1 x

NameError: name 'x' is not defined
```

- 파이썬에서 문자열을 처리해주기 위해서는 따옴표로 문자를 감싸야 한다

```
[6]: 'x'
```

```
[6]: 'x'
```

- 문자열: 2개이상의 문자가 연달아 존재하는 경우 -> 역시 문자로 취급된다

```
[5]: 'String'
```

```
[5]: 'String'
```

- 공백도 문자열로 취급된다

```
[7]: ' '
```

```
[7]: ' '
```

- 숫자도 따옴표 안에 들어가면 문자열로 취급된다

```
[8]: '11111'
```

```
[8]: '11111'
```

- 다음처럼 여러 종류의 문자가 합쳐져서 문자열을 형성할 수 있다.

```
[9]: 'Hello World My name is 11111 abcde'
```

```
[9]: 'Hello World My name is 11111 abcde'
```

1. 값 (Value)

1-3. 불리언 값

- 불리언: 참/거짓 두가지 경우만을 나타내는 특수한 값
- 조건문과 함께 프로그램의 분기에 사용됨

```
[10]: True
```

```
[10]: True
```

```
[11]: False
```

```
[11]: False
```

```
[12]: true
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-12-724ba28f4a9a> in <module>
----> 1 true

NameError: name 'true' is not defined
```

```
[13]: false
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-13-b73d74fcede9> in <module>
----> 1 false

NameError: name 'false' is not defined
```

2. 변수(Variable)

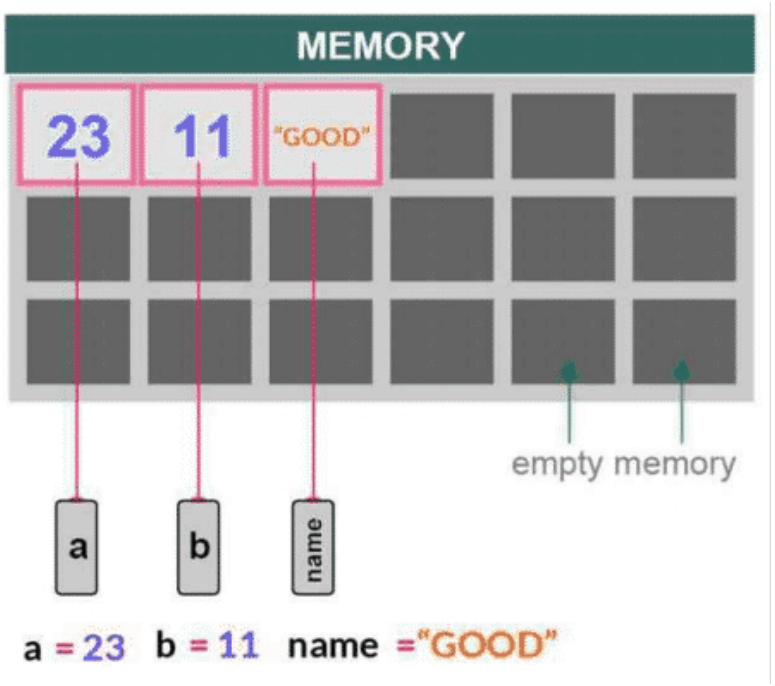
- 값: 프로그램이 처리하는 기본 단위
- 특정 값을 나중에 사용하려면? → 변수에 값을 대입한다.

- 변수명 = 값

- 변수명: 저장하고싶은 값의 별명

- 프로그램 내부에서 변수명을 통해 값에 접근할 수 있다.

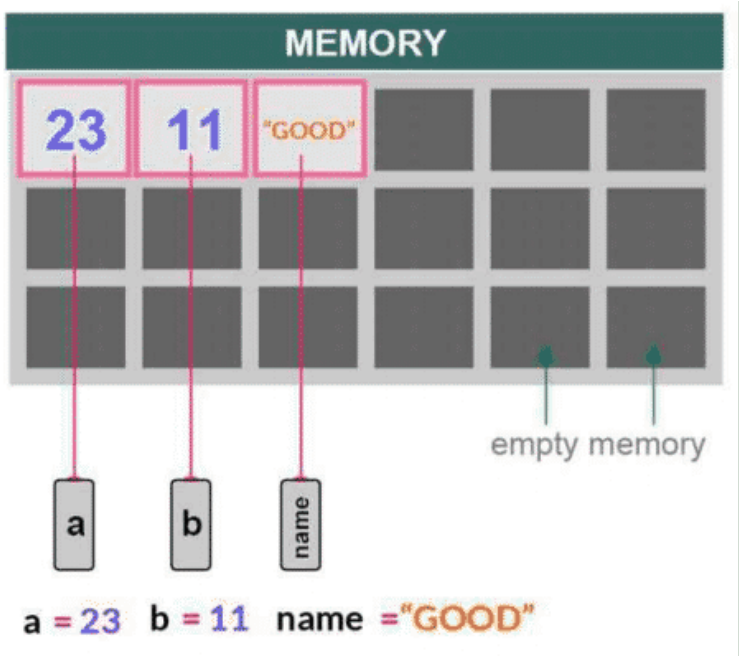
2. 변수(Variable)

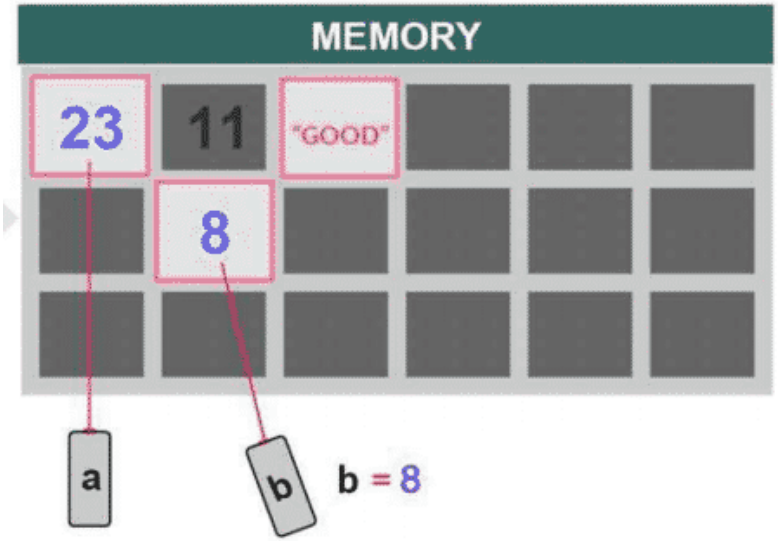


2. 변수(Variable)

- 변수로 하여금 다른 값을 가리키게 할 수 있다.
- 변수는 다양한 값을 가리킬 수 있다.(당연)

2. 변수(Variable)





명령어

- 파이썬 프로그램에서 사용하는 주요 구문
- 다음 명령어는 변수명으로 선언하면 안됨 (가능은 함)
  - print : 변수나 값을 출력
  - if : 조건문
  - for : 반복
  - while : 반복
  - def : 함수정의
  - input : 사용자 입력
  - class : 클래스
  - list : 리스트 자료구조
  - dict : 딕셔너리 자료구조
  - set : 세트 자료구조

2. 변수(Variable)

2. 변수(Variable)

- 값: 컴퓨터 메모리 안에 저장되어 있음
- 변수: 메모리안에 저장된 값을 가리키는 리모컨
- 변수를 통해 메모리안에 저장된 값에 접근할 수 있다.

[14]:
```
pi
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-14-f84ab820532c> in <module>
----> 1 pi

NameError: name 'pi' is not defined
```

- 변수에 값을 할당

[15]:
```
pi = 3.141592
```

- 변수가 가리키는 값을 출력

[16]:
```
print(pi)
```

```
3.141592
```

[17]:
```
pi
```

[17]:
```
3.141592
```

2. 변수(Variable)

- 변수가 가리키는 값을 출력

[16]:
```
print(pi)
```
```
3.141592
```

[17]:
```
pi
```

[17]:
```
3.141592
```

- 변수로 하여금 다른 값을 가리키게 할 수 있다.

[18]:
```
pi = 100
```

[19]:
```
print(pi)
```
```
100
```

- 다양한 종류의 값을 가리킬 수 있다.

[21]:
```
pi = 'hello world'
pi
```

[21]:
```
'hello world'
```

[22]:
```
pi = True
pi
```

[22]:
```
True
```

3. 연산자

- 두 개의 값을 결합하여 새로운 값을 만들어내는 기호

- 숫자사이의 연산자: 사칙연산
- 문자열 사이의 연산자: 문자열 붙이기
- 불리언값 사이의 연산자: and, not, or

- 비교 연산자

3-1. 숫자연산

[23]:
```
10+10
```

[23]:
```
20
```

[24]:
```
10*10
```

[24]:
```
100
```

[25]:
```
print(10+10)
print(2*3)
print(2/3)
print(2-50)
```

```
20
6
0.6666666666666666
-48
```

[26]:
```
print(4**3)
```

```
64
```

[27]:
```
print(15%3)
```

```
0
```

3-2. 문자연산

3.2 문자연산

```
[29]: 'Hello '+'World'

[29]: 'Hello World'

[31]: 'Hello '*5

[31]: 'Hello Hello Hello Hello Hello '
```

3-3 불리언 연산

[32]:
```
print(True and True)
print(True and False)
print(False and True)
print(False and False)
```

```
True
False
False
False
```

[33]:
```
print(True or True)
print(True or False)
print(False or True)
print(False or False)
```

```
True
True
True
False
```

[34]:
```
print(not True)
print(not False)
```

```
False
True
```

3-4. 비교연산자

- 비교 연산자
  - 두 값의 일치 여부를 비교하여 판단
  - 두 숫자의 수량의 차이를 비교 판단
  - ">, <, >=, <=, ==, !=“
- 비교 연산자의 결과
  - 불리언 값 (참 또는 거짓)

3-4. 비교연산자

```
[2]: 3>5
```

```
[2]: False
```

```
[3]: 3<5
```

```
[3]: True
```

```
[4]: 3==5
```

```
[4]: False
```

```
[7]: 3!=5
```

```
[7]: True
```

```
[5]: 3<=3
```

```
[5]: True
```

```
[6]: 3==3
```

```
[6]: True
```

연습문제

연습 문제 2.1.1

파이썬으로 계산기로 사용하여 다음 연산을 한다.

1. $3 \times 2 - 8 \div 4$
2. $25 \times 6 \div 3 + 17$
3. $39021 - 276920 \div 12040$
4. $2^6 - 10 \,\%\, 6$

위 식에서 %는 나머지를 구하는 연산이다.

연습 문제 2.1.2

파이썬을 계산기로 사용하여 다음 연산을 한다.

1. $12 - (5 \times 7 + 1)$
2. $5 \times \{8 + (10 - 6) \div 2\}$
3. $48320 - \{(365 - 5 \times 9) \div 16\} \times 987$
4. $((3^4 - 3 \times 7) \,\%\, 5 + 4)^2$

연습문제

연습문제 2.1.3

1. 파이썬으로 계산기로 사용하여 답이 True 인 부등식을 3개를 만든다.
2. 파이썬으로 계산기로 사용하여 답이 False 인 부등식을 3개를 만든다.

연습문제 2.1.5

변수들의 값을 바꾸어 가면서 다음 수식을 계산해 보자.

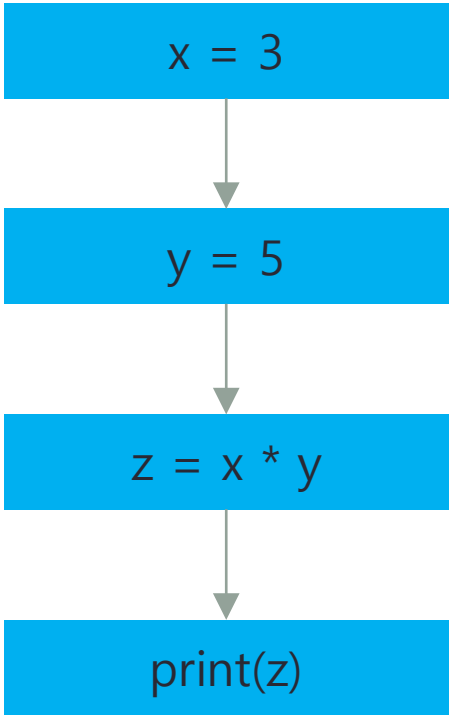
1. $$(2x-1)^2+1$$
2. $$x^{2y}\cdot(z+10)$$
3. $$((j=0)\ \&\ (0<k))\mid(i\leq 100)$$

파이썬 문법 기초 (조건문)

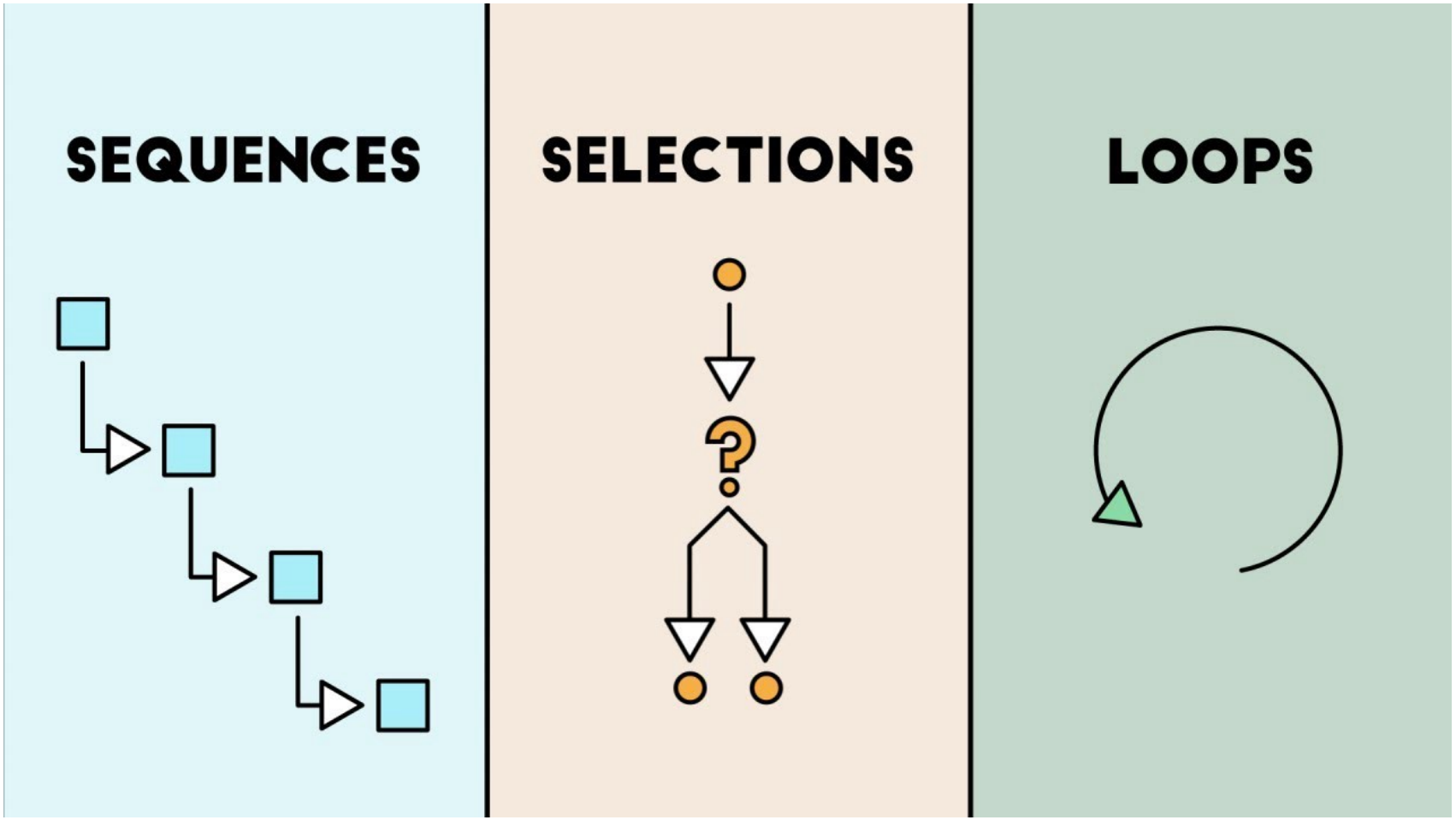
시스템경영공학부
이지환 교수

지금까지 프로그램의 흐름 (sequence)

```
1   x=3
2   y=5
3   z=x*y
4   print(z)
```

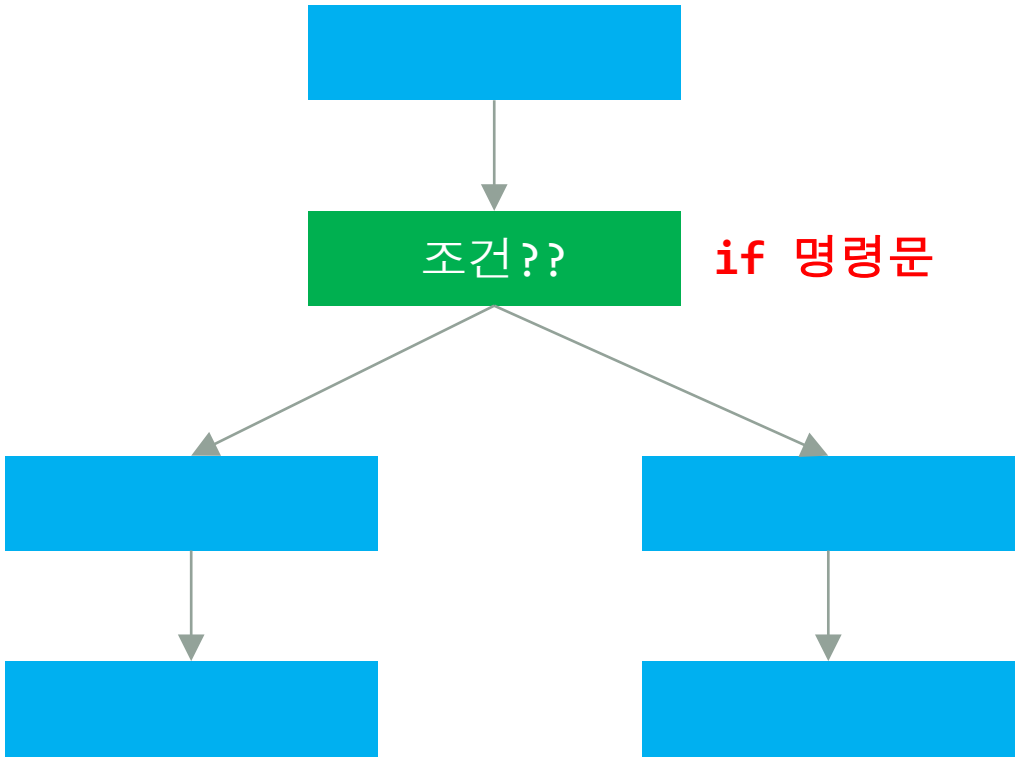


프로그램 코드의 진행



조건에 따른 흐름의 선택 (Selection)

- 조건에 따라 프로그램의 방향을 바꾸어 진행



4. 조건문

- 조건에 따라 프로그램의 흐름을 바꾸고 싶을 때 사용

```
if 참 또는 거짓을 가지는 값:
    조건이 참일 때 실행되는 명령들
else:
    조건이 거짓일 때 실행되는 명령들
```

- 조건이 2개 일때
- 조건이 2개 이상일때

4-1. If문의 기본 구조

[3]:
```
if True:
    print('Hello World')
```

```
Hello World
```

[4]:
```
if False:
    print('Hello World')
```

4-1 비교연산자 복습

- 비교연산자의 결과물 → '불리언 값'

[1]:
```
a=3
b=2
print(a>b)
print(a<b)
print(a==b)
print(a!=b)
```

```
True
False
False
True
```

[2]:
```
a='hello'
b='hello'
c='heRRp'
print(a==b)
print(a==c)
```

```
True
False
```

4-2 비교연산자와 함께 조건문 사용

[5]:
```
x = 3
if x<4:
    print('Hello World')
```

```
Hello World
```

[6]:
```
x = 5
if x != 10:
    print('Hello World')
```

```
Hello World
```

4-3 if와 else의 사용

- 둘 중 하나의 조건을 선택할 때 사용
  - If 이거나,
  - Else 이거나,

[8]:
```
x = 30
if x>20:
    print('x is larger than 20.')
else:
    print('x is less than 20.')
```

```
x is larger than 20.
```

[9]:
```
x = 10
if x>20:
    print('x is larger than 20.')
else:
    print('x is less than 20.')
```

```
x is less than 20.
```

4-4 조건이 두 개 이상일 때

- 계층적인 조건문 사용으로 3개 이상의 조건 표현 가능

[11]:
```
x = 30
if x>20:
    print('x is larger than 20.')
else:
    if x> 10:
        print('x is less than 20.')
    else:
        print('x is less than 10.')
```

```
x is larger than 20.
```

[12]:
```
x = 15
if x>20:
    print('x is larger than 20.')
else:
    if x> 10:
        print('x is less than 20.')
    else:
        print('x is less than 10.')
```

```
x is less than 20.
```

[13]:
```
x = 5
if x>20:
    print('x is larger than 20.')
else:
    if x> 10:
        print('x is less than 20.')
    else:
        print('x is less than 10.')
```

```
x is less than 10.
```

4-5 코드블럭

- 코드블럭: 명령문이 한번에 실행되는 단위
  - 탭으로 분리

[14]:
```
x = 3
if x<4:
    print('Hello World')
    print('x is less than four.')
```

```
Hello World
x is less than four.
```

[15]:
```
x = 5
if x<4:
    print('Hello World')
    print('x is less than four.')
else:
    print('Hello World')
    print('x is larger than four.')
```

```
Hello World
x is larger than four.
```

조건문 연습문제

연습 문제 2.5.4

어떤 농장에서는 수박이 10kg이 넘으면 1등급, 그렇지 않고 7kg이 넘으면 2등급,, 그렇지 않고 4kg이 넘으면 3등급, 나머지는 4등급을 준다고 한다. 이 수박의 등급을 정하는 파이썬 코드를 작성한다.

연습 문제 2.5.5

죄수의 딜레마는 게임 이론의 유명한 사례이다

- https://ko.wikipedia.org/wiki/죄수의_딜레마

두 명의 범죄자 A, B가 체포되어 서로 다른 취조실에서 격리되어 심문을 받고 있다. 이들에게 자백 여부에 따라 다음의 선택이 가능하다.

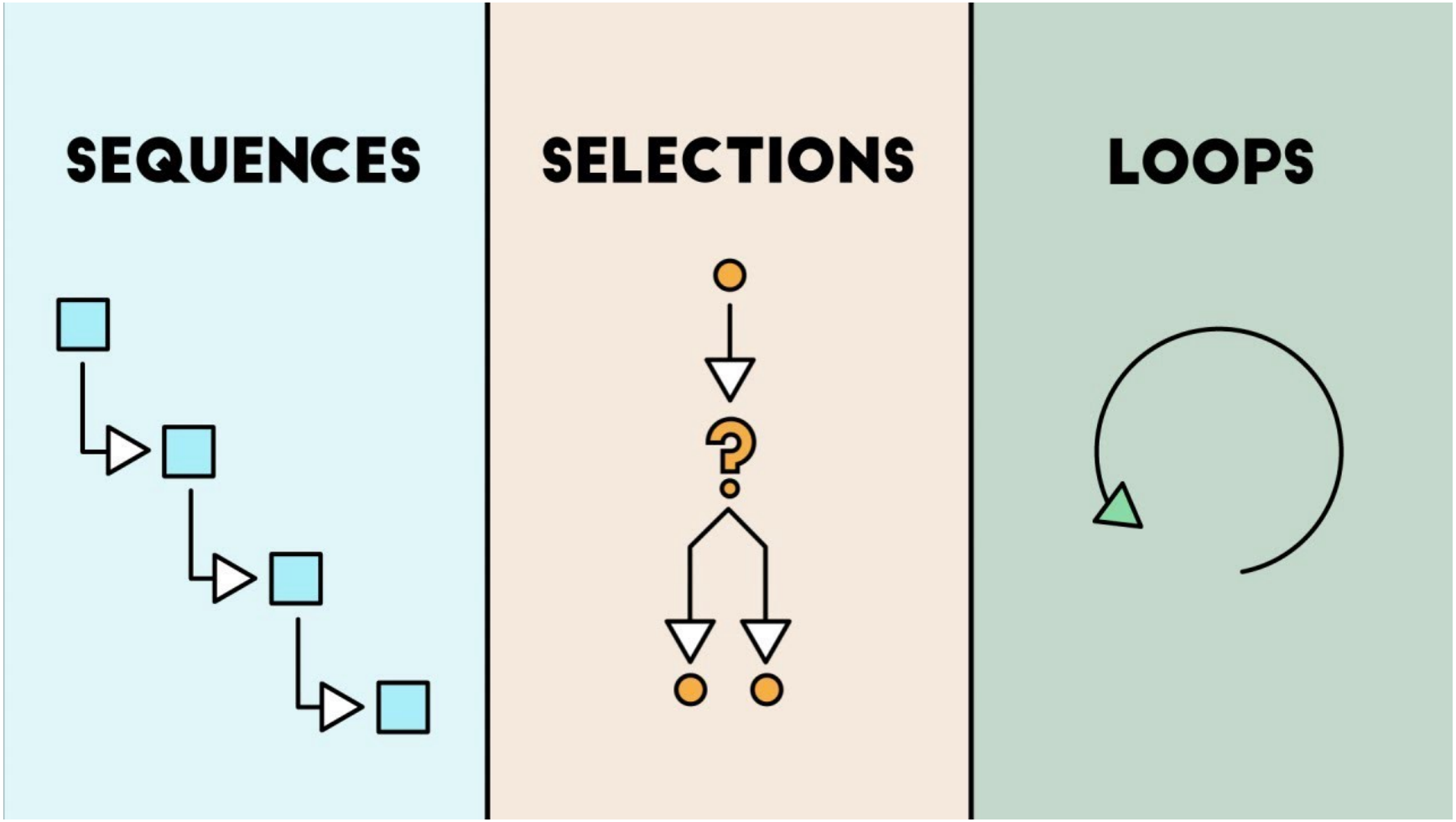
- 둘 중 하나가 배신하여 죄를 자백하면 자백한 사람은 즉시 풀어주고 나머지 한 명이 10년을 복역해야 한다.
- 둘 모두 서로를 배신하여 죄를 자백하면 둘 다 5년을 복역한다.
- 둘 모두 죄를 자백하지 않으면 둘 다 1년을 복역한다.

죄수를 나타내는 변수 `XA`, `XB`는 자백하면 `True`, 그렇지 않으면 `False`를 가지는 값이다. 이때 변수 `XA`, `XB` 값에 따라 각각이 복역해야 하는 연 수를 변수 `YA`, `YB`라고 할 때 `YA`, `YB`를 계산하는 프로그램을 만든다.

파이썬 문법 기초 (반복문)

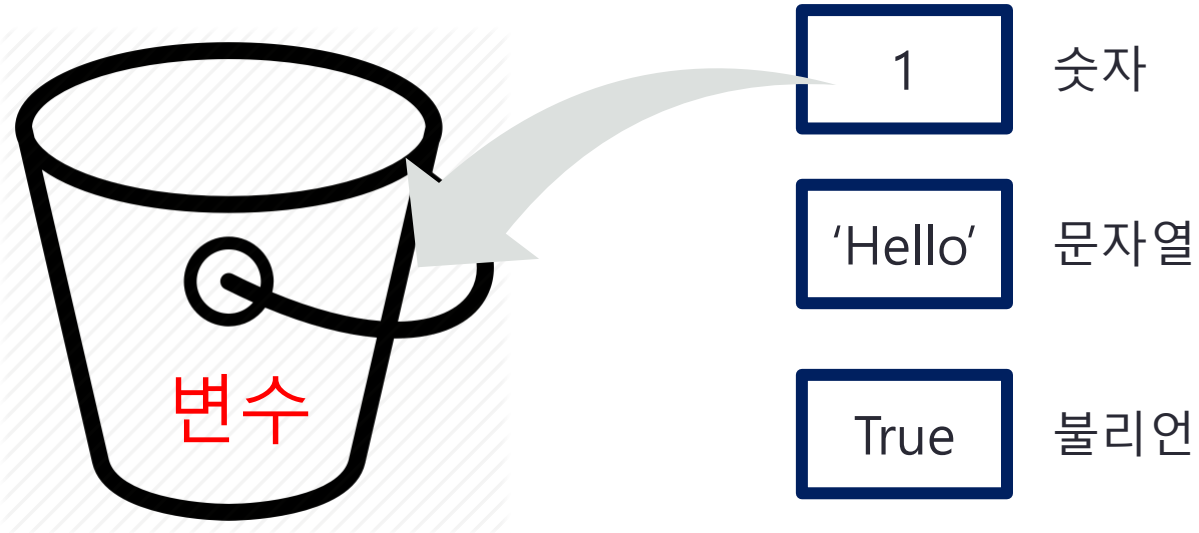
시스템경영공학부
이지환 교수

5-1. 리스트



5-1. 리스트

- **지금까지: 하나의 변수에 한 개의 값만 저장**

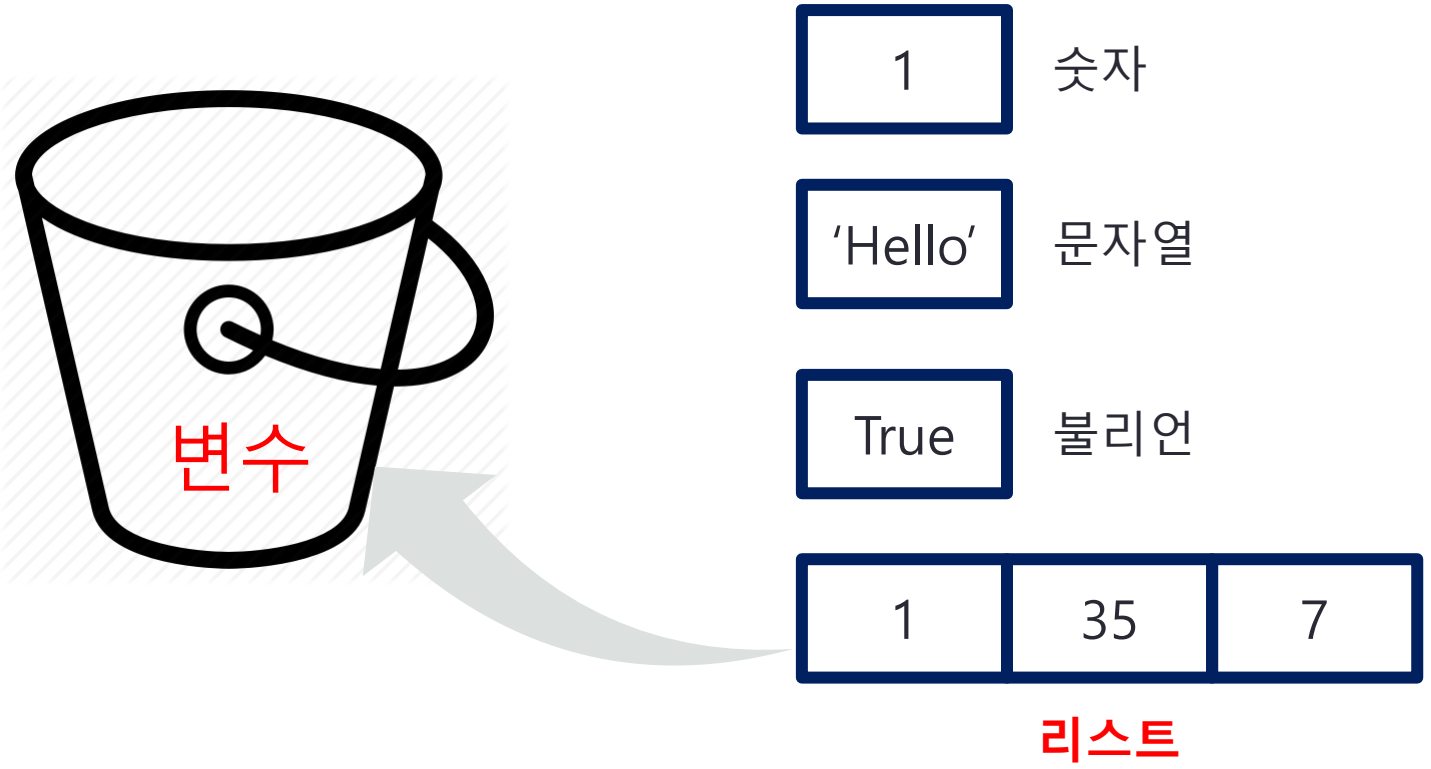


5-1. 리스트

- 파이썬에서 다룰 수 있는 값의 종류 중 하나
- 여러 개의 값을 순서에 따라 담아 놓은 '자료구조'

5-1. 리스트

- 리스트도 또다른 형태의 값이다 → 따라서 변수에 저장할 수 있다.



5-1. 리스트

[1]:
```
# 리스트 생성문법
[1,35,7]
```

[1]:
```
[1, 35, 7]
```

[2]:
```
x = [1,35, 7] #리스트 변수에 저장
print(x) #리스트 값 확인
```

```
[1, 35, 7]
```

5-1. 리스트

- 리스트의 길이, 담을수 있는 값의 종류 모두 자유롭다.

1	35	7	
1	35	7	20
‘a’	35	‘Hello’	20
True	False	‘Hello’	1

5-1. 리스트

- 변수들에 리스트를 저장하고, print를 이용해 값을 확인해보시오

[3]:
```
x = [1, 35, 7, 20]
y = ['a', 35, 'hello', 20] # 다양한 종류의 값을 담을 수 있다
z = [True, False, 1, 3] # Boolean값도 담을 수 있다
m = [] #아무값도 없어도 리스트다
print(x)
print(y)
print(z)
print(m)
```

```
[1, 35, 7, 20]
['a', 35, 'hello', 20]
[True, False, 1, 3]
[]
```

[5]:
```
mixed = ['hello',[1,2],[]] #리스트 안에 리스트도 담을 수 있다
print(mixed)
```

```
['hello', [1, 2], []]
```

5-2. for를 이용한 반복

- 코드가 일정한 횟수 혹은 조건에 따라 반복적으로 실행되는 구조

- for를 이용한 반복
- while을 이용한 반복

5-2. for를 이용한 반복

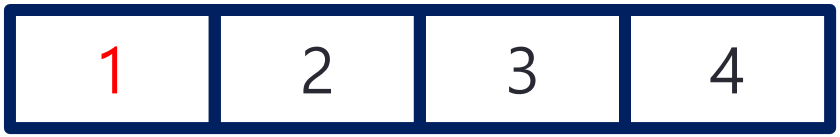
```
for 카운터변수 in 리스트:
    반복할 구문
    반복할 구문
```

[6]:
```
numbers = [1,2,3,4,5]
for i in numbers:
    print(i)
```

```
1
2
3
4
5
```

5-2. for를 이용한 반복

```
numbers = [1,2,3,4,5]
for i in numbers:
   print(i)
```

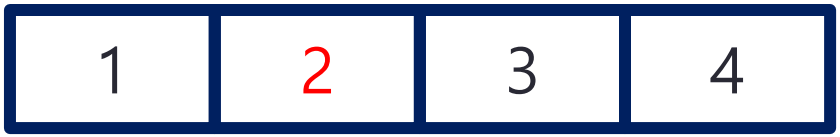


리스트 안의 값을 모두 순회할때까지 반복

첫번째 반복 → i = 1

5-2. for를 이용한 반복

```
numbers = [1,2,3,4,5]
for i in numbers:
   print(i)
```

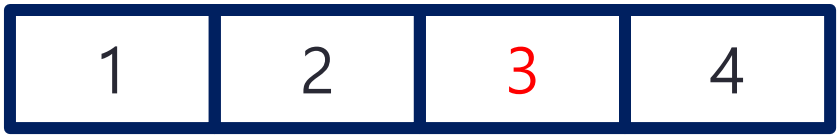


리스트 안의 값을 모두 순회할때까지 반복

첫번째 반복 → i = 1
두번째 반복 → i = 2

5-2. for를 이용한 반복

```
numbers = [1,2,3,4,5]
for i in numbers:
   print(i)
```

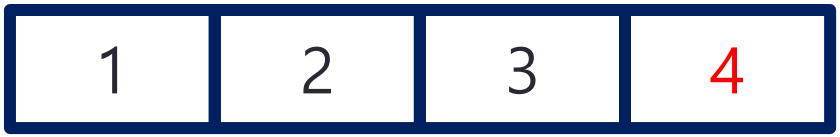


리스트 안의 값을 모두 순회할때까지 반복

첫번째 반복 → i = 1
두번째 반복 → i = 2
세번째 반복 → i = 3

5-2. for를 이용한 반복

```
numbers = [1,2,3,4,5]
for i in numbers:
   print(i)
```



리스트 안의 값을 모두 순회할때까지 반복

첫번째 반복 → i = 1
두번째 반복 → i = 2
세번째 반복 → i = 3
네번째 반복 → i = 4

5-2. for를 이용한 반복

[7]:
```
numbers = [1,2,3,4,5]
for i in numbers:
    print('hello')
```

```
hello
hello
hello
hello
hello
```

[8]:
```
for i in numbers:
    print(i+3)
```

```
4
5
6
7
8
```

5-2. for를 이용한 반복

[10]:
```
temp = 'any'
print('Hi' + temp + 'Man!')
```

```
HianyMan!
```

[9]:
```
x = ['kim','na','park','lee']
for i in x:
    print('Hi', i, 'Nice to meet you!')
```

```
Hi kim Nice to meet you!
Hi na Nice to meet you!
Hi park Nice to meet you!
Hi lee Nice to meet you!
```

5-3. range 명령어

```
range(n)
```

- 입력값: n, 숫자
- 출력값: [0, 1, ⋯, n-1], 리스트
- n개의 값을 가진 리스트를 간편하게 생성하고 싶을때 활용

[12]:
```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

5-3. range 명령어

[13]:
```
for i in range(5):
    print('hello world')
```

```
hello world
hello world
hello world
hello world
hello world
```

[14]:
```
for i in range(5):
    print(i**2)
```

```
0
1
4
9
16
```

연습문제

문제1.¶

다음처럼 1월부터 12월까지 바꾸어가며 영어로 출력하는 반복문을 설계해보시오

One of the months of the year is	January
One of the months of the year is	February
One of the months of the year is	March
...	
One of the months of the year is	December

문제 2.¶

주어진 8개의 섭씨온도 [15, 30, 25, 30, 75, 60, 95, 100]에 대하여 각각을 화씨온도로 변환하여 출력하는 프로그램을 설계하시오

문제 3.¶

다음과 같은 숫자의 리스트 [11, 10, 31, 3, 66, 17, 42, 99, 20] 중에서 짝수를 골라서 순서대로 출력하는 프로그램을 설계해보시오.

문제 4.¶

다음과 같이 *을 계단형태로 출력하고자 한다. 반복문을 사용하여 30층 짜리 계단을 출력하는 프로그램을 설계해보시오

```
*
**
***
****
*****
...
```

5-4. 변수 업데이트

- = 의 오른쪽 부분의 연산이 모두 이루어져 값으로 변환된 후 변수에 저장된다.

```
a = 3
n = a+3
```

3

5-4. 변수 업데이트

- 변수에 값을 할당하기 전 =의 오른쪽 부분의 연산이 먼저 이루어진다.

```
a = 3
n = a+3
```

6

5-4. 변수 업데이트

- 다음과 같이 변수 자신을 업데이트할 수 있다.

[15]:
```
n = 3
n = n+3
print(n)
```

```
6
```

[16]:
```
n = 3
n = n+3
n = n+3
n = n+3
print(n)
```

```
12
```

5-4. 변수 업데이트

[17]:
```
x = 0
for i in range(5):
    x = x+1
print(x)
```

```
5
```

[18]:
```
x=0
for i in range(5):
    x = x+i
print(x)
```

```
10
```

[19]:
```
x=0
for i in [1,3,5,7,9]:
    x = x+i
print(x)
```

```
25
```

연습문제(ex04.py)

문제 1.¶

1부터 50까지의 연속된 값의 곱(50팩토리얼)을 하나의 변수로 업데이트하며 계산해보시오.
for를 사용해야 한다.

문제 2.¶

1부터 100까지의 연속된 값 중 홀수만 골라서 합하는 프로그램을 작성하시오.
for와 range를 사용할 것

문제 3.¶

1부터 100까지 연속된 값 중 3의 배수 또는 7의 배수인 숫자의 갯수를 구하려고 한다.
프로그램을 설계해보시오.

5-5. 중첩반복문

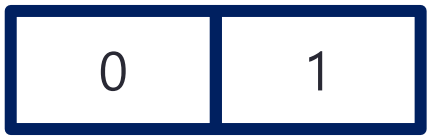
[20]:
```
for i in range(2):
    print('hi')
    for j in range(3):
        print('there')
```

```
hi
there
there
there
hi
there
there
there
```

5-5. 중첩반복문

```
for i in range(2):
    print('hi')
    for j in range(3):
        print('there')
```

```
hi
there
there
there
hi
there
there
there
```



hi

5-5. 중첩반복문

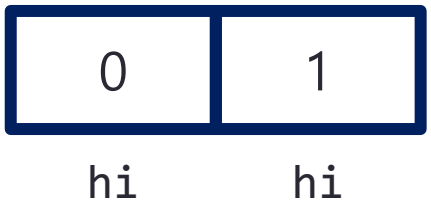
```
for i in range(2):
    print('hi')
    for j in range(3):
        print('there')
```


```
hi
there
there
there
hi
there
there
there
```

5-5. 중첩반복문

```
for i in range(2):
    print('hi')
    for j in range(3):
        print('there')
```

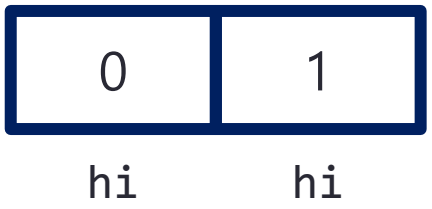
```
hi
there
there
there
hi
there
there
there
```

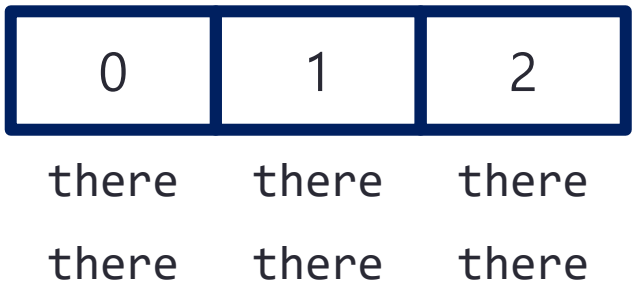


5-5. 중첩반복문

```
for i in range(2):
    print('hi')
    for j in range(3):
        print('there')
```

```
hi
there
there
there
hi
there
there
there
```





5-5. 중첩반복문

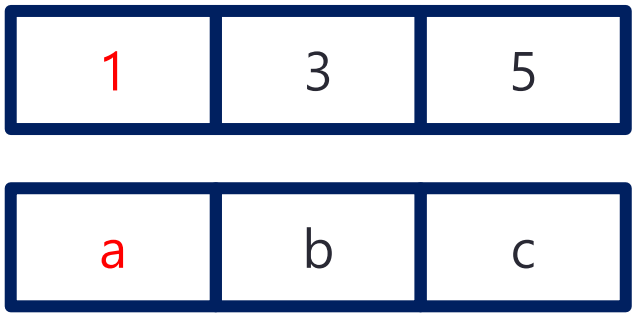
[21]:
```
for i in [1,3,5]:
    for j in ['a','b','c']:
        print(i,j)
```

```
1 a
1 b
1 c
3 a
3 b
3 c
5 a
5 b
5 c
```

5-5. 중첩반복문

```
for i in [1,3,5]:
    for j in ['a','b','c']:
        print(i,j)
```

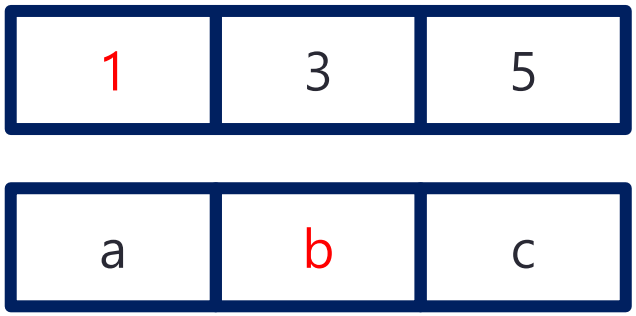
```
1 a
1 b
1 c
3 a
3 b
3 c
5 a
5 b
5 c
```



5-5. 중첩반복문

```
for i in [1,3,5]:
    for j in ['a','b','c']:
        print(i,j)
```

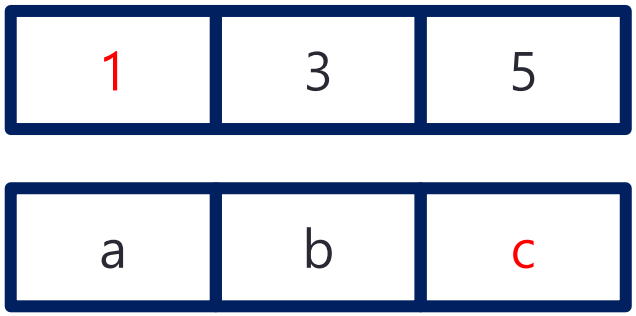
```
1 a
1 b
1 c
3 a
3 b
3 c
5 a
5 b
5 c
```



5-5. 중첩반복문

```
for i in [1,3,5]:
    for j in ['a','b','c']:
        print(i,j)
```

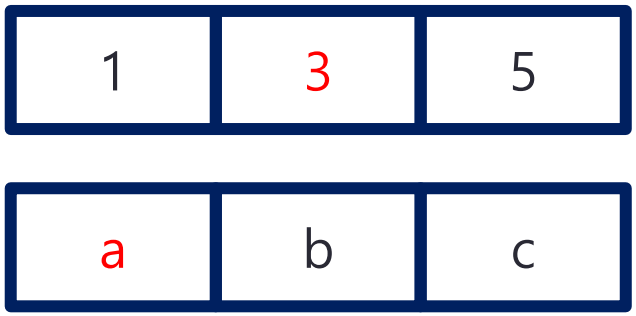
```
1 a
1 b
1 c
3 a
3 b
3 c
5 a
5 b
5 c
```



5-5. 중첩반복문

```
for i in [1,3,5]:
    for j in ['a','b','c']:
        print(i,j)
```

```
1 a
1 b
1 c
3 a
3 b
3 c
5 a
5 b
5 c
```



연습문제

연습문제 1.¶

다음과 같이 구구단을 출력하는 프로그램을 작성해보시오.

```
1*1=1
1*2=2
..
9*8=72
9*9=81
```

연습문제

연습문제 2.¶

프랑스의 수학자 페르마(Ferma)의 마지막 정리는 다음과 같다.

n 이 2보다 큰 자연수인 경우에, $a^n+b^n=c^n$ 이 되는 자연수 a,b,c 는 존재하지 않는다.

n이 3이고 a,b,c 가 1부터 10까지의 자연수일 때 페르마의 마지막 정리가 사실임을 중첩 반복문을 써서 보인다. (힌트: 1부터 10까지 반복되는 for 반복문이 3개 중첩되어야 한다.)

$$a^n + b^n = c^n$$

n=3에 대해서 위 공식을 만족하는
1에서 10까지의 정수 a,b,c가 존재하지
않음을 보여야 한다.

연습문제

연습 문제 2.7.1

for 반복문과 문자열 연산을 사용하여 다음과 같이 출력한다.

```
*
**
***
****
*****
******
*******
********
*********
**********
```

연습 문제 2.7.2

for 반복문과 문자열 연산을 사용하여 다음과 같이 출력한다.

```
**********
*********
********
*******
******
*****
****
***
**
*
```

연습 문제 2.7.3

for 반복문과 문자열 연산, if 조건문을 사용하여 다음과 같이 출력한다.

```
*
**
***
****
*****
******
*******
********
*********
**********
*********
********
*******
******
*****
****
***
**
*
```

연습문제

연습 문제 2.7.5

어떤 주식의 가격은 매일 한 번 동전을 던져서 앞면이 나오면 전날 가격의 2배가 되고, 뒷면이 나오면 전날 가격의 절반이 된다. 1일에 주식의 가격이 1,024원이었을 때, 4일 주식의 가격이 나올 수 있는 경우를 모두 구한다. (힌트: for 반복문이 3개 중첩되어야 한다)

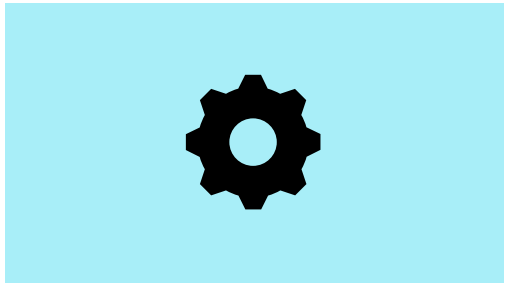
파이썬 문법 기초 (함수)

시스템경영공학부
이지환 교수

6. 함수(Function)

- 어떤 코드를 실행하기로 컴퓨터와 맺은 약속
- 약속에 이름을 붙임
- 함수 이름의 호출만으로 약속에 따라 프로그램 실행

함수



6-1. 함수의 정의

- **프로그램 덩어리에 이름을 붙이고 컴퓨터와 약속**

```
def 함수이름():
    코드
    코드
    …
```

6-1. 함수의 정의

- 다음 두줄을 실행하는 코드를 함수로 정의해보자. 함수의 이름은 my_function이다.

```
[1]: print('hello')
     print('world')

hello
world
```

```
[14]: def my_function():
          print('hello')
          print('world')

      my_function()

hello
world
```

함수이름:
my_function

하는일: 2줄 출력

6-1. 함수의 호출

- 함수를 정의하였다면 함수의 이름을 호출하는 것만으로, 함수에 정의되어있는 코드를 실행할 수 있다.

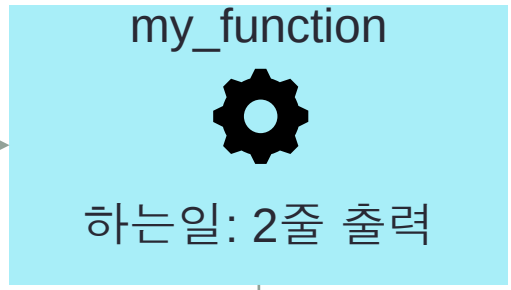
[14]:
```
def my_function():
    print('hello')
    print('world')

my_function()
```

```
hello
world
```

6-1. 함수의 호출

- 함수를 정의하였다면 함수의 이름을 호출하는 것만으로, 함수에 정의되어있는 코드를 실행할 수 있다.

```
def my_function():
    print('hello')
    print('world')
```




```
print('test')
my_function()
print('test')
```

```
test
hello
world
test
```

6-2. 함수의 정의 (입력변수)

- 프로그램 덩어리에 이름을 붙이고 컴퓨터와 약속
- 입력변수를 받아 처리

```
def 함수이름(입력변수):
    코드
    코드
    …
```

6-2. 입력변수 실습

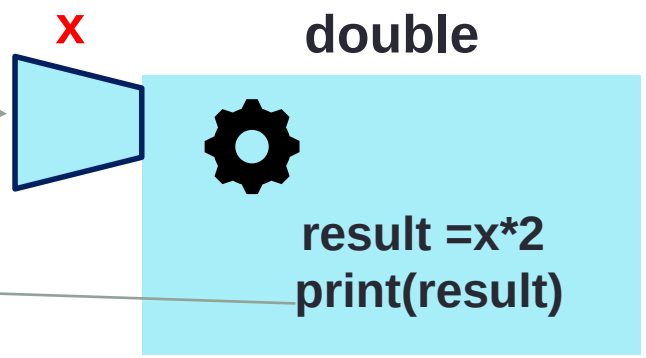
```
[3]: def double(x):
         result = 2*x
         print(result)
```

```
[5]: double(3)
     double(4)
     double(5)
```

```
6
8
10
```

```
[10]: ## 왜 이런 결과가 나왔을까?
      my = double(3)
      print(my)
```

```
6
None
```



6-3. 입력변수 실습

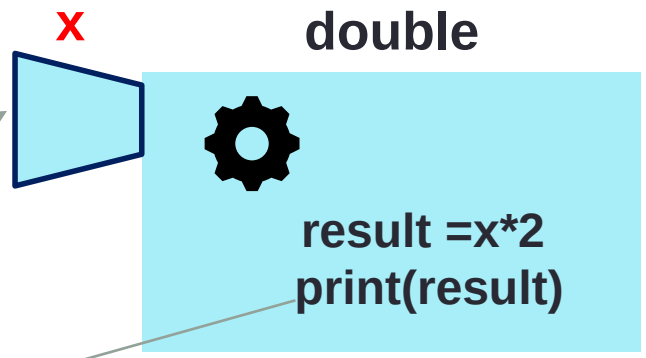
```
[3]: def double(x):
         result = 2*x
         print(result)
```

```
[5]: double(3)
     double(4)
     double(5)
```

```
6
8
10
```

```
[10]: ## 왜 이런 결과가 나왔을까?
      my = double(3)
      print(my)
```

```
6
None
```



6-3. 함수의 정의 (입력변수 + 출력변수)

- 함수에서 처리된 결과값을 프로그램의 다른 부분에서 활용하고 싶다면 출력변수를 정의해야 함
- 처리된 결과를 return 명령어를 이용하여 처리

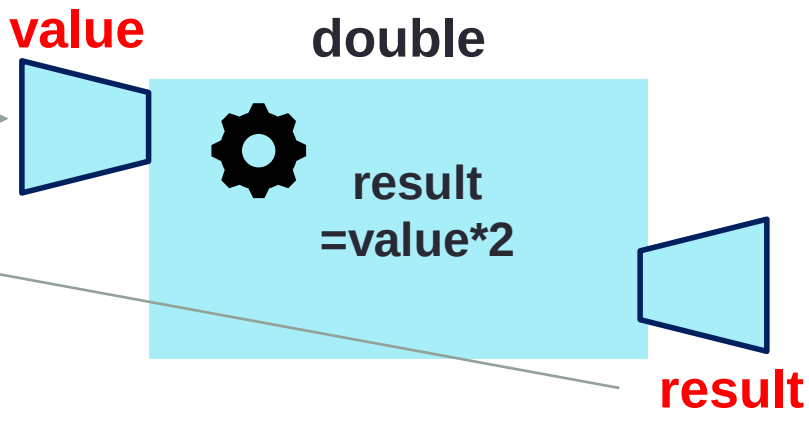
```
def 함수이름(입력변수):
    코드
    코드
    …
    return 출력변수
```

6-3. 출력변수 실습

[12]:
```
def double(x):
    return x*2
```

[13]:
```
y=double(2)
print(y)
y=double(10)
print(y)
z=double(30)
print(z)
```

```
4
20
60
```



6-4. 입력값이 여러 개인 함수

- **입력변수를 여러 개 받을 수 있다.**
  - jegop(2,3) --> 8
  - jegop(3,4) --> 81

[11]:
```
def jegop(x,y):
    return x**y

jegop(2,3)
```

[11]: 8

[12]:
```
jegop(3,4)
```

[12]: 81

6-5. 지역변수

- 함수의 입출력의 사용된 변수들은 함수 내에서만 유용하다.
- 이러한 변수들을 지역변수(local variable)라고 정의한다.

  - 함수에 사용되는 변수들은 함수 안에서만 의미있다.
  - 즉, 함수 밖에서 사용할 수 없다.

[23]:
```
def tenTimes(x):
    result = x*10
    return result
```

[24]:
```
result
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-24-0ac921c19f1a> in <module>
----> 1 result

NameError: name 'result' is not defined
```

[25]:
```
result=10
tenTimes(2)
```

[25]:
```
20
```

[28]:
```
result
```

[28]:
```
10
```

연습문제

문제1.¶

어떤수 n을 입력하면 1부터 n까지의 합을 계산하여 출력하는 함수를 만들어 보시오.
함수의 이름은 my_sum이고, 입력 변수는 number, 출력변수는 result이다.

In [2]:

```
def my_sum(number):
    ## 이 부분을 작성해보시오.

    ##
    return result
```

문제2.¶

어떤수 n을 입력하여 짝수면 True, 홀수면 False를 입력하는 함수를 만들어 보시오.

In [1]:

```
def check_even(number):
    ## 이 부분을 작성해보시오.

    ##
    return result
```

연습문제

연습 문제 2.6.1

1. 짝수가 입력되면 짝수라는 문자열을, 홀수가 입력되면 홀수라는 문자열을 반환하는 함수를 만든다.
2. 윤년을 나타나는 수가 입력되면 윤년이라는 문자열을, 그렇지 않은 수가 입력되면 평년이라는 문자열을 반환하는 함수를 만든다.

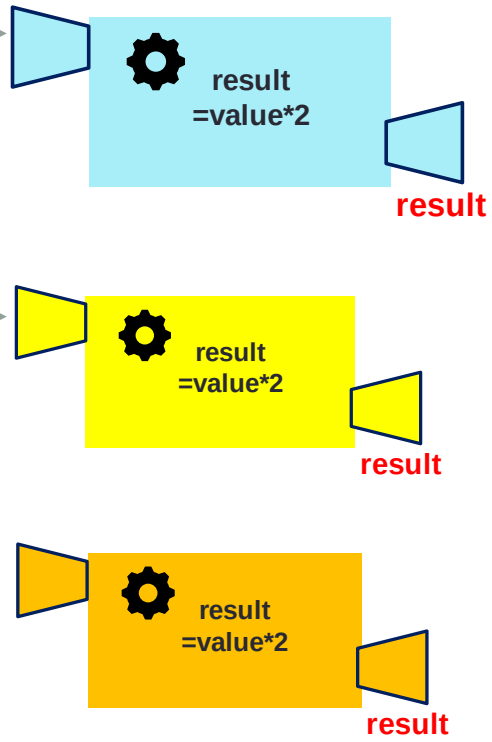
연습 문제 2.6.2

평년일 때 1, 3, 5, 7, 8, 10, 12 월은 31일, 4, 6, 9, 11월은 30일, 2월은 28일이다. 월을 나타내는 숫자를 입력하면 그 달의 날짜 수를 반환하는 함수 `days1`를 만든다. 사용 예는 다음과 같다.

```
>>> days1(11) # 11월의 날짜 수
30
```

6-6. 함수를 사용한 프로그램 구조화

```
def function_1(value):
    code..
    code..
    return x

def function_2(value):
    code..
    code..
    return x

def function_3(value):
    code..
    code..
    return x

function_1()
function_2()
function_3()
```



6-6. 함수를 사용한 프로그램 구조화

```
def print_line():
    print('='*50)

def print_five():
    for i in range(5):
        print('hello')

def print_star():
    print('*'*50)

print_line()
print_star()
print_five()
print_star()
print_line()
```

```
==================================================
**************************************************
hello
hello
hello
hello
hello
**************************************************
==================================================
```

6-7. 함수의 입출력값에는 제한이 없다.

[7]:
```
def sum_list(a):
    result=0
    for i in a:
        result = result + i
    return result

inputList = [1,3,5,7,9]
output = sum_list(inputList)
print(output)
```

```
25
```

[8]:
```
sum_list(3)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-114a0811582d> in <module>
----> 1 sum_list(3)

<ipython-input-7-a634a7d8a67e> in sum_list(a)
      1 def sum_list(a):
```

6-8. 함수안의 함수

[11]:
```
def print_line():
    print('='*50)

def print_five():
    for i in range(5):
        print('hello')

def print_star():
    print('*'*50)

def main():
    print_line()
    print_star()
    print_five()
    print_star()
    print_line()

main()
```

```
==================================================
**************************************************
hello
hello
hello
hello
hello
**************************************************
==================================================
```

6-8. 함수안의 함수

[12]:
```
def square(x):
    return x**2

def my_sum(x):
    result = 0
    for i in range(x+1):
        result = result + square(i)
    return result

print(my_sum(3))
print(my_sum(5))
```

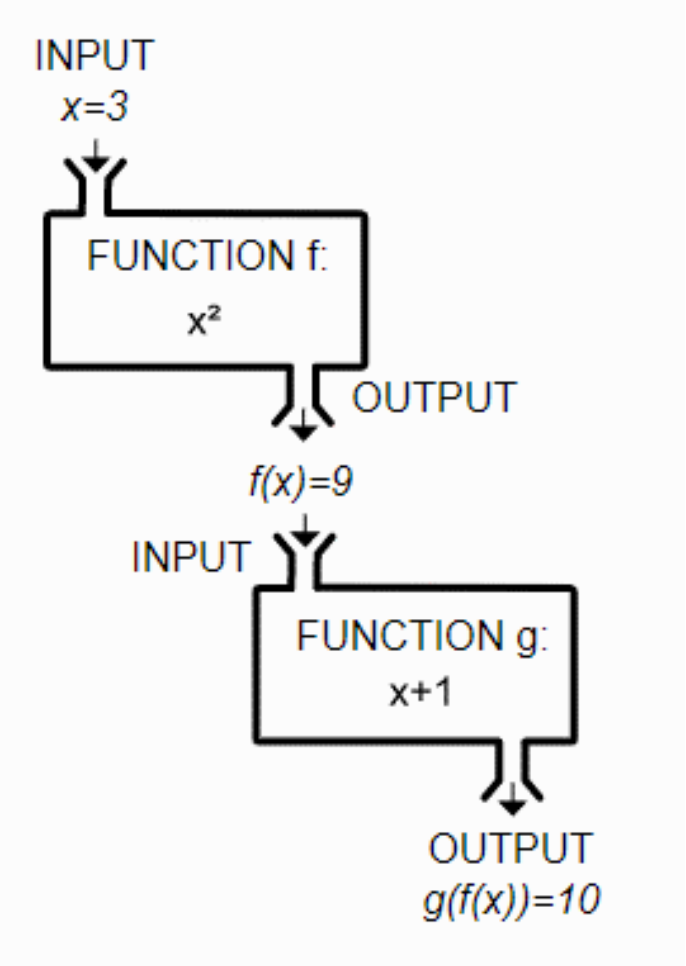
```
14
55
```

연습문제

In [11]:

```
# 연습문제 check_even은 주어진 수가 짝수일때 True 아닐때 False를 반환하는 함수이다.
# my_even_sum은 1부터 주어진 number까지 짝수만 더하여 출력하는 함수이다.
# check_even을 my_even_sum함수 안에서 사용하여 짝수만 더하여 출력하는 함수를 계산해 보시오.

def check_even(number)
    ## 이 부분을 작성하시오

    ##
    return result

def my_even_sum(number):
    ## 이 부분을 작성하시오

    ##
    return result
```

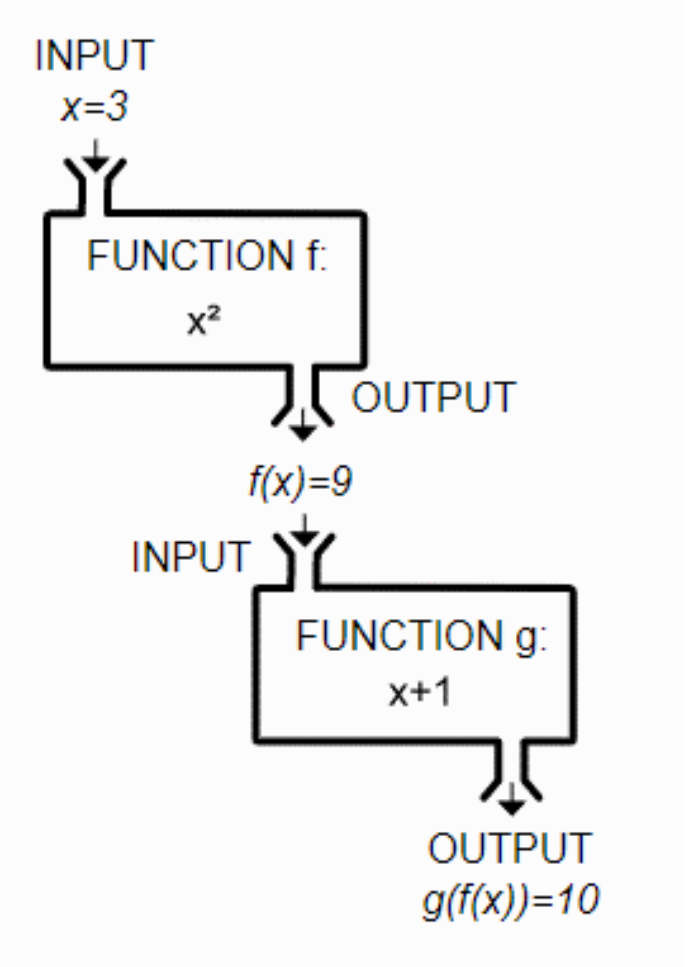
6-9. 함수의 결과값을 입력값으로 사용



함수의 결과값을 입력값으로 사용

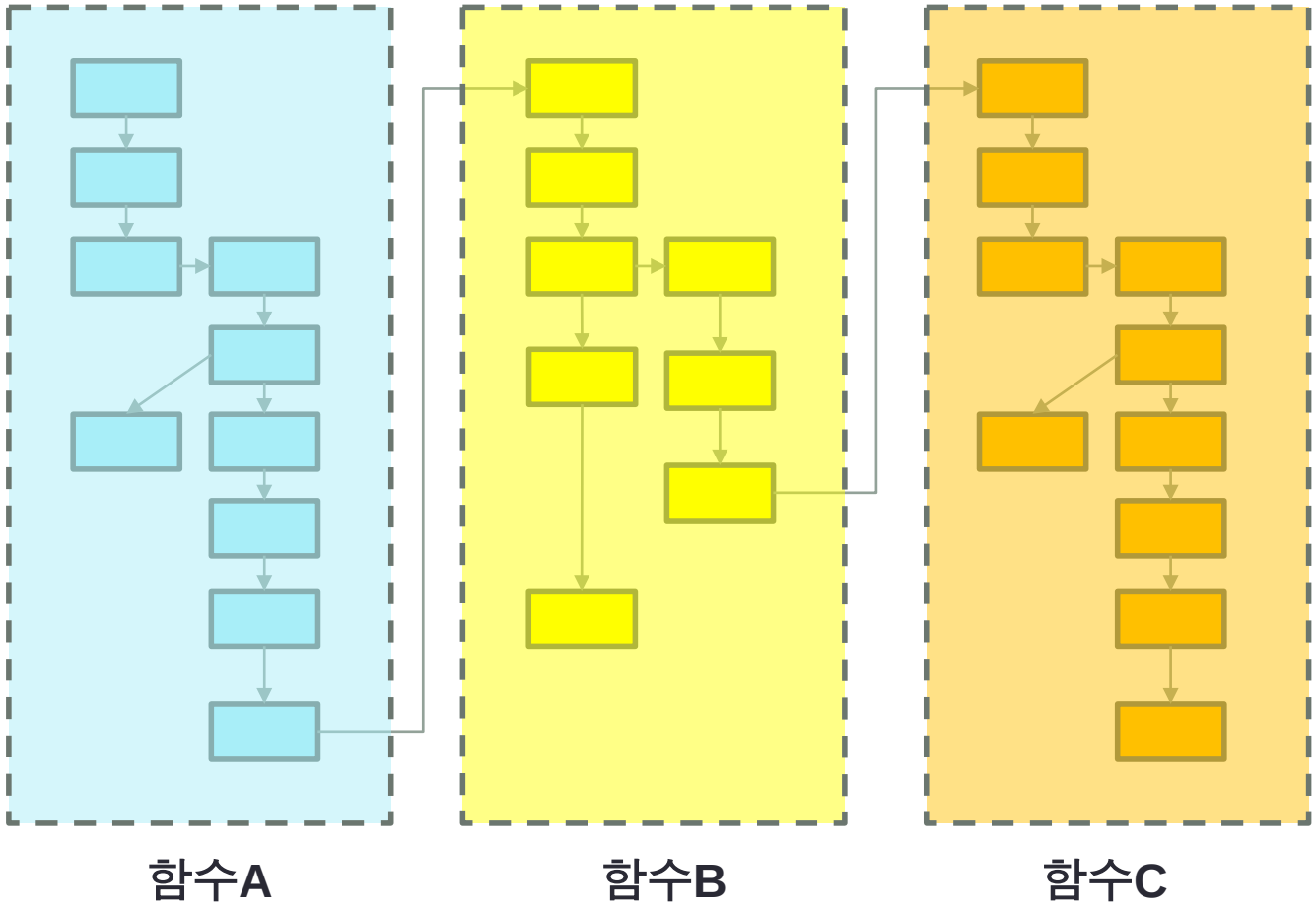
[13]:
```
def f(x):
    return x**2

def g(y):
    return y+1

print(g(f(3)))
```

```
10
```



함수가 필요한 이유

- 프로그램 덩어리를 함수로 나누어 관리



파이썬 문법 기초 (자료구조: 리스트)

시스템경영공학부
이지환 교수

7-1. 리스트의 성질

```
[5]: [88, 90, 100]
```

```
[5]: [88, 90, 100]
```

```
[6]: x=[1,3,5,7,9]
     print(x)
```

```
[1, 3, 5, 7, 9]
```

```
[9]: albist =  ["hello", 2.0, 5, [10, 20]]
     albist
```

```
[9]: ['hello', 2.0, 5, [10, 20]]
```

7-2. 리스트 인덱싱

- 인덱스: 리스트에 저장된 개별 값의 주소
- 리스트 n번째 값 = 리스트[n-1]

```
[11]: a=[1,3,5,7,9]
      a[0]

[11]: 1

[12]: a[1]

[12]: 3

[13]: a[2]

[13]: 5
```

7-3. 리스트 슬라이싱

[16]:
```
a=[1,3,5,7,9]
print(a[0:3])
```

```
[1, 3, 5]
```

[17]:
```
print(a[1:3])
```

```
[3, 5]
```

[18]:
```
print(a[:2])
print(a[2:])
```

```
[1, 3]
[5, 7, 9]
```

7-4. 리스트 값 변경

[24]:
```
x = list(range(1,10,2))
x
```

[24]:
```
[1, 3, 5, 7, 9]
```

[25]:
```
x[0] = 11
x
```

[25]:
```
[11, 3, 5, 7, 9]
```

[26]:
```
x[1] = 12
x
```

[26]:
```
[11, 12, 5, 7, 9]
```

7-5. 리스트에 값 추가하기

```
[27]: x.append(1)
      x

[27]: [11, 12, 5, 7, 9, 1]

[28]: x.append(3)
      x

[28]: [11, 12, 5, 7, 9, 1, 3]
```

7-6. 리스트에서 값 지우기

```
[29]: del x[0]
      x

[29]: [12, 5, 7, 9, 1, 3]

[30]: del x[0]
      x

[30]: [5, 7, 9, 1, 3]

[31]: del x[3]
      x

[31]: [5, 7, 9, 3]
```

7-7. 복수할당

[32]:
```
x = [1,2,3]
a,b,c = x
```

[33]:
```
print(a,b,c)
```

```
1 2 3
```

7-8. 리스트 메소드

[6]:
```
#insert
mylist = [5,27,3,12]
mylist.insert(1,12)
print(mylist)
mylist.insert(3,50)
print(mylist)

#count
print(mylist.count(12))

#sort
mylist.sort()
print(mylist)
```

```
[5, 12, 27, 3, 12]
[5, 12, 27, 50, 3, 12]
2
[3, 5, 12, 12, 27, 50]
```

- .insert(n,x) : n번째 인덱스 부분에 x를 추가
- .count(x) : 리스트 안에 x의 갯수
- .sort() : 리스트 안에 들어있는 숫자 정렬

연습문제

연습 문제 2.10.1

어떤 학생이 5개의 과목을 수강하여 다음과 같은 성적(grade)을 받았다. (4점이 만점)

$$X = 4, 3, 2, 3, 4$$

이 5개 과목의 이수 학점(credit hours)은 각각 다음과 같다.

$$W = 3, 3, 1, 2, 2$$

이 때 평균 평점(GPA)은 성적의 단순 평균이 아니라 이수 학점을 가중치(weight)로 써서 다음과 같이 가중 평균을 구해야 한다.

$$\frac{\text{성적과 이수 학점을 곱한 값의 총합 즉, 가중합}}{\text{이수 학점의 총합}} = \frac{3 \times 4 + 3 \times 3 + 1 \times 2 + }{3 + 3 + 1 + 2}$$

i번째 과목의 성적을 X_i라고 하고 i번째 과목의 이수 학점을 W_i라고 하면 가중 평균은 다음과 같은 수식으로 나타낼 수도 있다.

$$\frac{W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5}{W_1 + W_2 + W_3 + W_4 + W_5}$$

이 학생의 평균 평점을 구하는 코드를 작성한다.

연습문제

연습 문제 2.10.2

자료의 분산(Variance)는 각 자료 값에서 자료의 평균값을 뺀 나머지를 제곱한 값의 평균을 말한다. 예를 들어 자료가 다음과 같다고 하자

$$X = 6, 5, 4, 7, 3, 5$$

이 자료의 평균은 다음과 같다.

$$\frac{(6+5+4+7+3+5)}{6} = 5$$

각 자료 값에서 자료의 평균값을 뺀 나머지를 제곱한 값을 모두 더한 값의 평균은 다음과 같이 구한다.

$$\frac{(6-5)^2 + (5-5)^2 + (4-5)^2 + (7-5)^2 + (3-5)^2 + (5-5)^2}{6}$$

이 자료의 분산을 구하는 코드를 작성한다.

파이썬 문법 기초 (자료구조: 딕셔너리)

시스템경영공학부
이지환 교수

8-1. 딕셔너리 자료형

- 딕셔너리: key-value 의 쌍으로 구성
- key: 유일한 값 (숫자, 문자, 어떤 값이든 가능)
- value: 키에 대응하는 값

```
[2]: {"a":10, "b":20}

[2]: {'a': 10, 'b': 20}
```

```
[3]: x={"a":10, "b":20}
     print(x)

     {'a': 10, 'b': 20}
```

8-2. key를 이용해 value 호출

```
[4]: print(x['a'])

     10

[6]: print(x['b'])

     20
```

8-3. 딕셔너리 갱신, 추가, 삭제

[7]:
```
# 추가
x['c']=30
print(x)
```

```
{'a': 10, 'b': 20, 'c': 30}
```

[8]:
```
# 수정
x['a']=15
print(x)
```

```
{'a': 15, 'b': 20, 'c': 30}
```

[9]:
```
# 삭제
del x['c']
print(x)
```

```
{'a': 15, 'b': 20}
```

8-4. 키가 있는지 확인

```
[9]: print(x)
     {'a': 15, 'b': 20}

[10]: 'a' in x
[10]: True

[11]: 'c' in x
[11]: False
```

8-5. 딕셔너리와 반복

[12]:
```
for i in x:
    print(i)
```

```
a
b
```

[13]:
```
x.keys()
```

[13]:
```
dict_keys(['a', 'b'])
```

[14]:
```
for i in x.keys():
    print(i)
```

```
a
b
```

[16]:
```
for i in x:
    print(x[i])
```

```
15
20
```

연습문제

연습 문제 2.11.1

딕셔너리에 저장된 자료가 다음과 같다.

```
data = {
    "철수": 98,
    "영희": 80,
    "순이": 100,
    "돌이": 70,
}
```

for문을 사용하여 다음과 같이 출력하는 코드를 만들어라.

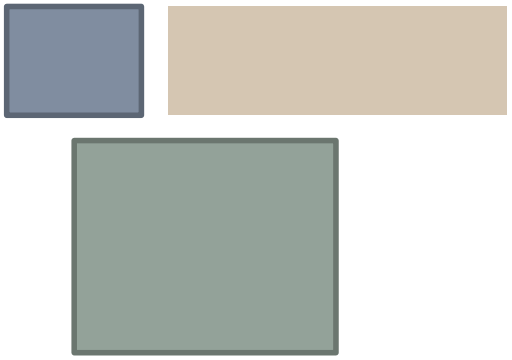
```
철수        98
영희        80
순이       100
돌이        70
================
평균        87
```

파이썬 문법 기초
클래스와 객체지향 프로그래밍

시스템경영공학부
이지환 교수

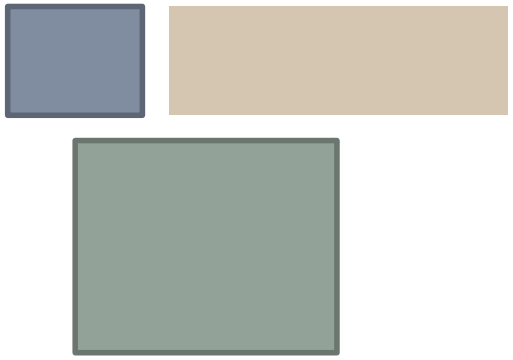
9-1. 클래스의 정의

- 파워포인트 사각형을 표현하기 위해 어떤 값들이 필요할까?
  - 높이
  - 너비
  - 선굵기
  - 선색깔
  - 면색깔
  - …
  - 위치



9-1. 클래스의 정의

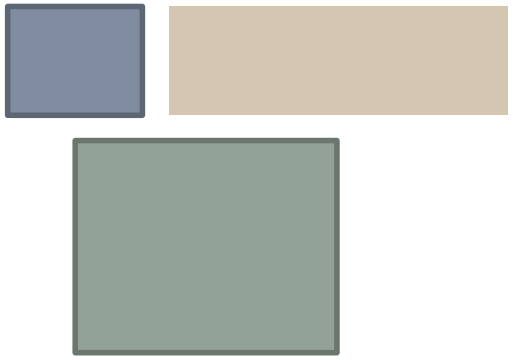
- 파워포인트 사각형을 표현하기 위해 어떤 값들이 필요할까?
  - 높이
  - 너비
  - 선굵기
  - 선색깔
  - 면색깔
  - …
  - 위치



- 우리는 이 값들을 사용하여 다양한 작업을 수행한다.
  - 위치변경
  - 선 바꾸기
  - 면 바꾸기
  - 복사
  - …

9-1. 클래스의 정의

- 파워포인트 사각형을 표현하기 위해 어떤 값들이 필요할까?
  - 높이
  - 너비
  - 선굵기
  - 선색깔
  - 면색깔
  - …
  - 위치



- 우리는 이 값들을 사용하여 다양한 작업을 수행한다.
  - 위치변경
  - 선 바꾸기
  - 면 바꾸기
  - 복사
  - …

- 또한 한 프로그램에서 여러 개의 사각형을 동시에 관리해야 한다.

9-1. 클래스의 정의

- 객체의 구성요소
  - 값
  - 행동 (메소드)

- 객체지향 프로그래밍
  - 객체의 정의를 위주로 프로그래밍을 진행

- 용어 구분
  - 클래스 : 객체의 설계도
  - 객체 : 설계도에 따라 생성되어 프로그램에서 실제로 다루는 대상

9-1. 클래스의 정의

```
# 데이터
h = 10
v = 20

# 데이터를 조작 --> 면적이라는 새로운 데이터 생성
def area(x,y):
    return x*y

# 결과 확인
area(h,v)
```

9-1. 클래스의 정의

- self: 클래스 자신을 의미

[2]:
```
# 클래스의 정의
class Rectangle():

    # 생성자
    # 데이터 정의
    def __init__(self, x,y):
        self.h = x #클래스 변수(값)
        self.v = y #클래스 변수

    # 데이터 조작(행동)의 정의
    def area(self):
        return self.h * self.v
```

객체가 생성될 때, x,y
두개의 변수를 받아
내부변수 h,v에 저장

이 사각형 객체의
너비를 계산하여 반환

9-1. 클래스의 정의

[2]:
```
# 클래스의 정의
class Rectangle():

    # 생성자
    # 데이터 정의
    def __init__(self, x,y):
        self.h = x #클래스 변수(값)
        self.v = y #클래스 변수

    # 데이터 조작(행동)의 정의
    def area(self):
        return self.h * self.v
```

클래스로부터 객체만들기

[3]:
```
r = Rectangle(10,20) #데이터 받기
a = r.area() #받은 데이터를 이용해 새로운 값 만들기
print(a)
```

```
200
```

[4]:
```
a = Rectangle(1, 1)   # 가로 1, 세로 1인 사각형
b = Rectangle(2, 1)   # 가로 2, 세로 1인 사각형
c = Rectangle(4, 2)   # 가로 4, 세로 2인 사각형
d = Rectangle(6, 3)   # 가로 6, 세로 3인 사각형
e = Rectangle(8, 5)   # 가로 8, 세로 5인 사각형
```

[6]:
```
print(a.h, a.v)
print(b.h, b.v)
print(c.h, c.v)
print(d.h, d.v)
print(e.h, e.v)
```

```
1 1
2 1
4 2
6 3
8 5
```

[5]:
```
print(a.area())
print(b.area())
print(c.area())
print(d.area())
print(e.area())
```

```
1
2
8
18
40
```

연습문제

연습 문제 2.12.1

삼각형의 넓이를 계산하기 위한 클래스를 만든다. 이 클래스는 다음과 같은 속성을 가진다.

- 밑변의 길이 `b` 와 높이 `h`
- 삼각형의 넓이를 계산하는 메서드 `area`

연습 문제 2.12.2

사각 기둥의 부피를 계산하기 위한 클래스를 만든다. 이 클래스는 다음과 같은 속성을 가진다.

- 밑면의 가로 길이 `a`, 밑면의 세로 길이 `b`, 높이 `h`
- 부피를 계산하는 메서드 `volume`
- 겉넓이를 계산하는 메서드 `surface`

9-2. 캐릭터 클래스(예시)

- 시나리오
  - 게임 케릭터를 객체로 표현
  - 데이터: 생명력
  - 행동: 공격받음 (현재 생명력에서 10씩 감소)

[6]:
```
class Character():
    def __init__(self):
        self.life = 1000

    def attacked(self):
        self.life = self.life - 10
        print("공격받음! 생명력 = ", self.life)
```

[7]:
```
a = Character()
b = Character()
c = Character()
```

[8]:
```
a.attacked()
```
```
공격받음! 생명력 =  990
```

[9]:
```
for i in range(3):
    b.attacked()
```
```
공격받음! 생명력 =  990
공격받음! 생명력 =  980
공격받음! 생명력 =  970
```

[10]:
```
for i in range(10):
    a.attacked()
```
```
공격받음! 생명력 =  980
공격받음! 생명력 =  970
공격받음! 생명력 =  960
공격받음! 생명력 =  950
공격받음! 생명력 =  940
공격받음! 생명력 =  930
공격받음! 생명력 =  920
공격받음! 생명력 =  910
공격받음! 생명력 =  900
공격받음! 생명력 =  890
```

9-3. 클래스의 상속

- 시나리오
  - 기본 캐릭터: 생명력을 가짐
  - 기본캐릭터의 속성을 이어받되, [전사, 마법사]로 캐릭터를 특화시키고자 함
  - [전사, 마법사]는 [힘,지능]을 추가적 속성으로 가지며, 직업에 따라 그 값이 다름

- 상속
  - 기본캐릭터의 속성을 이어받되, [전사, 마법사]로 캐릭터를 특화시키고자 함
  - 이 부분을 가능하게 해줌

[11]:
```
class Warrior(Character):
    def __init__(self):
        super(Warrior, self).__init__()
        self.strength = 15
        self.intelligence = 5
```

[12]:
```
class Wizard(Character):
    def __init__(self):
        super(Wizard, self).__init__()
        self.strength = 5
        self.intelligence = 15
```

[13]:
```
a = Warrior()
b = Wizard()
```

[14]:
```
a.life, b.life
```

[14]:
```
(1000, 1000)
```

[15]:
```
a.strength, b.strength
```

[15]:
```
(15, 5)
```

[16]:
```
a.intelligence, b.intelligence
```

[16]:
```
(5, 15)
```

[17]:
```
a.attacked()
```

```
공격받음! 생명력 =  990
```

[18]:
```
b.attacked()
```

```
공격받음! 생명력 =  990
```

9-3. 클래스의 상속

- 시나리오
  - 기본 캐릭터: 생명력을 가짐
  - 기본캐릭터의 속성을 이어받되, [전사, 마법사]로 캐릭터를 특화시키고자 함
  - [전사, 마법사]는 [힘,지능]을 추가적 속성으로 가지며, 직업에 따라 그 값이 다름

- 상속
  - 기본캐릭터의 속성을 이어받되, [전사, 마법사]로 캐릭터를 특화시키고자 함
  - 이 부분을 가능하게 해줌

```
[11]: class Warrior(Character):
          def __init__(self):
              super(Warrior, self).__init__()
              self.strength = 15
              self.intelligence = 5

[12]: class Wizard(Character):
          def __init__(self):
              super(Wizard, self).__init__()
              self.strength = 5
              self.intelligence = 15

[13]: a = Warrior()
      b = Wizard()

[14]: a.life, b.life

[14]: (1000, 1000)

[15]: a.strength, b.strength

[15]: (15, 5)

[16]: a.intelligence, b.intelligence

[16]: (5, 15)

[17]: a.attacked()
      공격받음! 생명력 =  990

[18]: b.attacked()
      공격받음! 생명력 =  990
```

9-4. 메소드 오버라이딩

- 상속받은 객체에 추가적인 행동을 부여할 수 있다.
- 현재
  - attacked (공격받음) 만 존재
- 시나리오
  - 전사, 마법사에 attack이라는 행동을 만들고, 직업에 따라 서로 다른 행동을 하도록 정의

[19]:
```
class Warrior(Character):
    def __init__(self):
        super(Warrior, self).__init__()
        self.strength = 15
        self.intelligence = 5

    def attack(self):
        print("육탄공격")
```

[20]:
```
class Wizard(Character):
    def __init__(self):
        super(Wizard, self).__init__()
        self.strength = 5
        self.intelligence = 15

    def attack(self):
        print('원거리 공격')
```

[21]:
```
a = Warrior()
b = Wizard()
a.attack()
b.attack()
```

```
육탄공격
원거리 공격
```

연습문제

연습 문제 2.12.3

게임 캐릭터 코드에서 `attacked` 메서드도 오버라이딩을 하여 전사와 마법사가 공격을 받을 때 life 속성값이 다르게 감소하도록 한다.

연습 문제 2.12.4

다음과 같이 자동차를 나타내는 `Car` 클래스를 구현한다.

- 이 클래스는 최고 속도를 의미하는 `max_speed`라는 속성과 현재 속도를 나타내는 `speed`라는 속성을 가진다.
- 객체 생성시 `max_speed` 속성은 160이 되고 `speed` 속성은 0이 된다.
- `speed_up`, `speed_down`이라는 메서드를 가진다. `speed_up`을 호출하면 `speed` 속성이 20씩 증가하고 `speed_down`을 호출하면 `speed` 속성이 20씩 감소한다.
- 스피드 속성 `speed`의 값은 `max_speed` 속성 값, 즉 160을 넘을 수 없다. 또 0 미만으로 감소할 수도 없다.
- 메서드는 호출시 속도 정보를 출력하고 명시적인 반환값을 가지지 않는다.
- 위 기능이 모두 정상적으로 구현되었음을 보이는 코드를 추가한다.

연습 문제 2.12.5

`Car` 클래스를 기반으로 `SportCar`와 `Truck`이라는 두 개의 자식 클래스를 구현한다.

- `SportCar` 클래스는 `max_speed` 속성이 200 이고 `speed_up`, `speed_down` 호출시 속도가 45씩 증가 혹은 감소한다.
- `Truck` 클래스는 `max_speed` 속성이 100 이고 `speed_up`, `speed_down` 호출시 속도가 15씩 증가 혹은 감소한다.
- 스피드 속성 `speed`의 값은 `max_speed` 속성 값을 넘을 수 없다. 또 0 미만으로 감소할 수도 없다.
- 메서드는 호출시 속도 정보를 출력하고 명시적인 반환값을 가지지 않는다.
- 위 기능이 모두 정상적으로 구현되었음을 보이는 코드를 추가한다.