파이썬 문법 기초
강의환경소개

시스템경영공학부
이지환 교수

컴퓨터

- 계산기와 컴퓨터의 차이점

- 계산기는 숫자의 계산만을 수행할 수 있음
- 컴퓨터는 계산 외에 {다양한 프로그램}을 실행할 수 있는 범용적 도구임

프로그램

- 프로그램(Program)
  - 컴퓨터에서 실행할 수 있는 명령어의 집합
- 소프트웨어 (Software)
  - 프로그램이 실행되는데 필요한 프로그램 뿐만 아니라 각종 데이터, 문서, 매뉴얼 등이 포함되어 있는 개념
  - 워드, 엑셀, 크롬, 윈도우, 게임, …
- 프로그래밍(Programming)
  - 프로그램을 작성하는 작업을 의미
  - 목적에 맞는 알고리즘을 설계하고 그에 따라 코드를 작성하는 과정
  - **프로그램은 사용자가 직접 만들어낼 수 있다!**

프로그래밍 언어

- 사용자가 컴퓨터에게 명령을 내리기 위해 사용하는 언어
- 프로그래밍 언어에는 다양한 종류가 있으며, 각각의 언어마다 특화된 분야가 존재

  - C, C++ (빠름, 어려움) : 시스템 프로그래밍
  - Java, C# (중간, 중간): 웹 (백엔드), 엔터프라이즈 솔루션, DB
  - Javascript (느림, 쉬움): 웹 (프론트)
  - Python (느림, 쉬움): 데이터 분석, 업무자동화, AI/머신러닝
- 하지만, 중요한 개념은 모두 동일

코드(code)

- 프로그램의 명령어들을 프로그래밍 언어을 통해 구현한 것.
- 소스코드(Source Code)로 불리운다.

Python 언어의 중요 활용분야

- 데이터 분석: 데이터를 수집, 가공, 분석 시각화
- 인공지능: 머신러닝, 딥러닝 등의 분야에서 파이썬의 라이브러리를 활용하여 모델 구축 및 학습
- 자동화: 일상적인 작업의 자동화 수행
- 웹프로그래밍 : 프레임워크를 사용한 웹페이지 개발

향후 4주간 수업의 목표

- 거시적 목표
  - 파이썬을 이용해 업무 프로세스를 개선하고 생산성 향상 도모
  - 파이썬을 업무시 발생하는 데이터의 분석과 과학적 관리 도모
  - 하반기 교육(머신러닝/AI)의 원활한 실습을 위한 사전준비

- 주요 내용 (할애 시간은 유동적으로 조절)
  - 파이썬의 기초적인 문법 습득
    - 값, 변수, 연산자, 흐름의 제어(조건, 반복), 모듈화 (함수), 클래스와 객체개념
  - 파이썬을 이용한 데이터 처리 방법 습득
    - 정형데이터(엑셀 형태로 표현되는)의 처리
    - 다양한 그래프 그리기
  - 파이썬을 이용한 업무자동화 사례 소개 및 습득
    - 파이썬을 활용한 엑셀,워드,pdf,파일작업등의 자동화
    - 파이썬을 활용한 데이터 수집(웹 스크래핑)

Google Colaboratory 소개

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

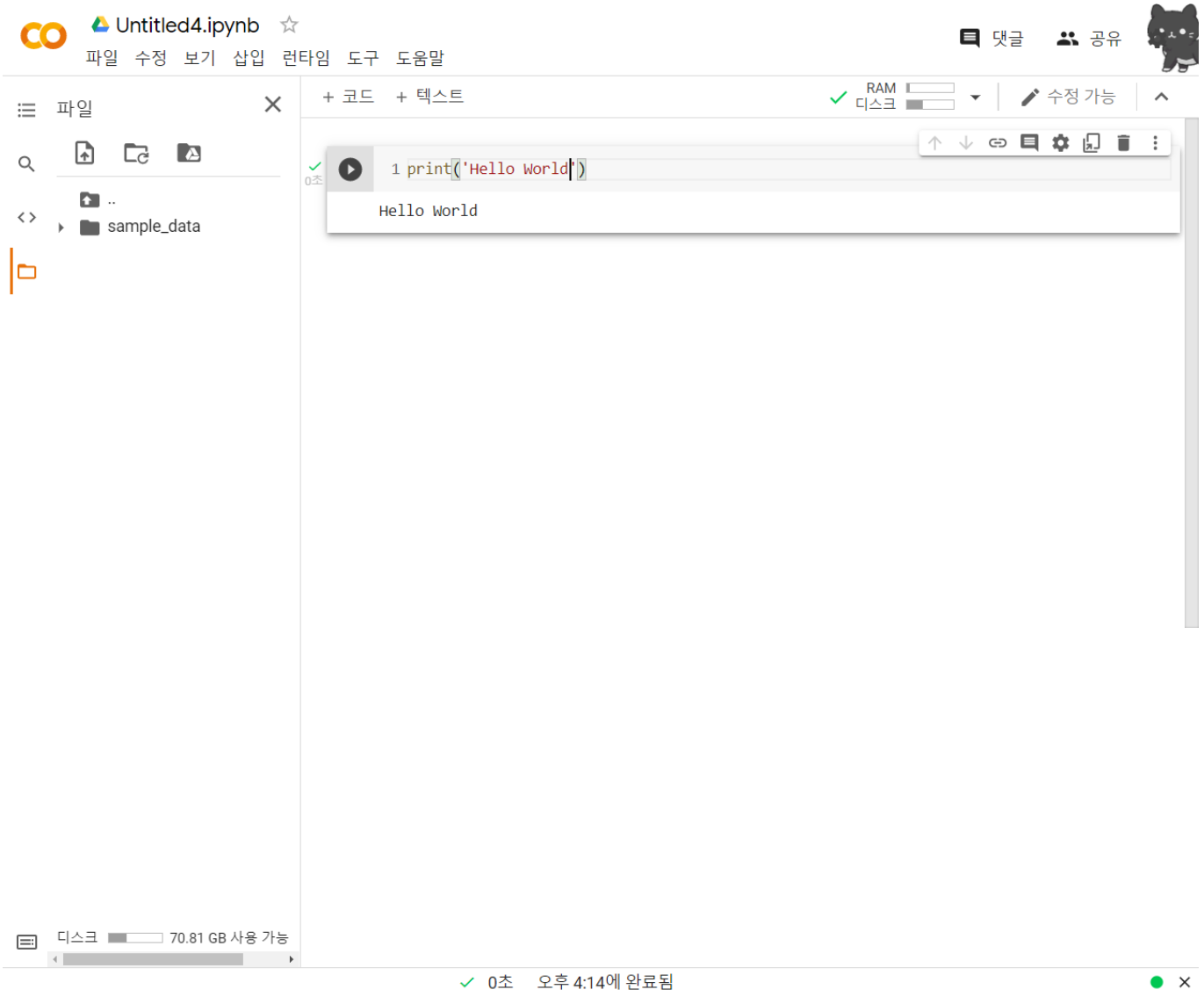
Colab의 기본 사용법 익히기

- 노트북
  - 코드와 문서를 함께 작성하고 실행할 수 있는 하나의 문서 단위
  - Colab에서는 작성된 노트북이 구글 드라이브에 자동 저장된다

- 셀
  - 노트북은 셀이라는 단위로 구성됨
  - 코드나 텍스트를 작성할 수 있는 작은 칸

- 런타임
  - 코드를 실행하는 환경을 의미

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

코드 셀

- 파이썬 코드를 작성하고 실행할 수 있는 셀
- 프로그램의 일부(혹은 전부)를 셀에서 구현
- 실행(play) 버튼을 통해 셀 안의 코드를 실행하고 그 결과를 확인할 수 있음
  - 정삭작동 혹은 오류
- 실행 순서기록
  - 실행되지 않은 셀은 []과 같은 표시가 나타남
  - 실행된 셀에는 번호가 부여됨 (숫자가 작을 수록 먼저 실행된 셀)
- 실행 순서가 중요
  - 코드를 실행할 때는, 변수나 함수 등의 정의가 이전에 실행된 셀에 있어야 정상적으로 실행됨

프로그램의 실행순서

- 한줄의 코드
  - 프로그램에서 하나의 명령문
- 코드 셀에는 여러 줄로 구성된 코드가 있을 수 있다.
  - 여러 개의 명령문으로 구성된 프로그램임
- 프로그램은 기본적으로 위에서 아래방향으로 순차적으로 명령문을 실행한다!
- 하지만 곧 실행의 순서를 제어하는 방법을 배울 것이다!

텍스트 셀

- 코드 셀과 달리 실행할 필요 없음
- 다만, HTML이나 Markdown과 같은 표기문자의 문법을 사용할 수 있음
  - # (글씨 매우 크게)
  - ## (글씨 적당히 크게)
- 코드의 구역을 나누거나 설명을 달때 유용

파이썬 문법 기초 (값)

시스템경영공학부
이지환 교수

1. 값 (Value)

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

print()

- 셀 실행시 값을 모니터로 출력해주는 함수
- 괄호안에 출력하는 값을 넣어주면, 입력된 값을 화면에 출력함
- ,를 이용해 여러 개의 값을 동시에 출력할 수 있음

연습문제

- 다양한 종류의 값을 테스팅 해보시오

파이썬 문법 기초 (변수)

시스템경영공학부
이지환 교수

변수(Variable)

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

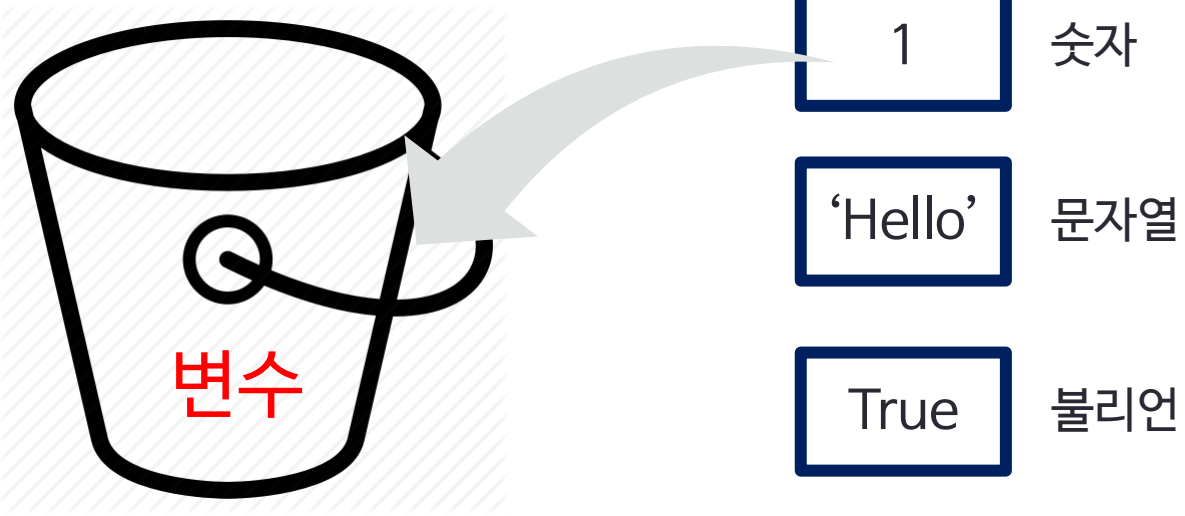
- 변수명 = 값

- =
  - 변수의 대입에 사용하는 연산자를 대입연산자라고 한다.

- 변수명: 저장하고싶은 값의 별명(내가 마음대로 붙인다)

- 프로그램 내부에서 변수명을 통해 값에 접근할 수 있다.

변수(Variable)



변수(Variable)

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

변수명 규칙

- (규칙 1) 알파벳과 숫자 그리고 특수문자 _만 허용한다
- (규칙 2) 숫자로 시작할 수 없다
- (규칙 3) 숫자로만 구성할 수 없다

- 허용
  - x_1233
  - asdf123
  - AAA

- 불가
  - 12X
  - !X
  - 23

연습문제

- 변수명 규칙을 따르는 5개의 변수를 만들어 보고, 각각에 다양한 종류의 값을 대입해 보시오.

파이썬 문법 기초 (연산자)

시스템경영공학부
이지환 교수

연산자의 종류

- 두 개의 값을 결합하여 새로운 값을 만들어내는 기호

- 숫자사이의 연산자
  +, -, *, /, %, //, **

- 문자열 사이의 연산자
  (문자)+(문자)
  (문자)*(숫자)

- 불리언값 사이의 연산자
  and, not, or

- 비교 연산자
  - >, <, >=, <=, ==, !=

연산자의 우선순위

- 연산자가 한 statement에서 여러 개가 연달아 나올경우 우선순위를 따른다. 다음은 숫자사이에서 사용되는 연산자의 우선순위이다.

- 1순위: 괄호
- 2순위: **
- 3순위: *, /, %, //
- 4순위: +,-,

- 가장 좋은 습관은 괄호를 붙여 우선순위를 명확하게 하는 것

문자연산

3.2 문자연산

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

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

```
[31]: 'Hello '*5
```

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

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
```

비교연산자

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

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

- 다음의 참 거짓여부를 확인해보시오
  - 25의 세제곱은 10000보다 크다
  - 46의 네제곱은 10000에서 100000사이에 있다.

파이썬 문법 기초 (예약명령어)

시스템경영공학부
이지환 교수

예약 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

예약명령어 예시

- print
  - 변수나 값을 출력
  - 콤마로 여러 개의 값을 구분하여 한번에 출력 가능

- input
  - 변수 = input(“안내값”)
  - 사용자의 입력을 기다렸다가 입력된 값을 변수에 저장
  - 입력값은 모두 문자열값으로 저장됨

- int(문자열값)
  - 괄호 안의 값이 문자열이지만 숫자인 경우 문자열 값을 숫자로 변환
  - float, double

- str(숫자)
  - 괄호 안의 값이 숫자값이라면 이를 문자열 값으로 변환

출력 명령어

2-2. 출력 명령어: print

[3]:
```
'hello'
```

[3]:
```
'hello'
```

[4]:
```
print("hello")
```

```
hello
```

[5]:
```
x = 'hello'
print(x)
```

```
hello
```

[6]:
```
1
```

[6]:
```
1
```

[2]:
```
print(1)
```

```
1
```

[7]:
```
x=1
print(1)
```

```
1
```

입력 명령어

2-3. 입력 명령어: input

프로그램 실행시 사용자의 입력을 기다림, 사용자가 특정 값을 입력하고 enter를 누르면, 결과 변수에 입력값이 저장됨 (주의사항: 모든 입력값은 문자열값으로 간주됨)

[9]:
```
x = input("값을 입력하세요")
x
```

```
값을 입력하세요 1
```

[9]:
```
'1'
```

[14]:
```
y = input("값을 입력하세요???")
print(y)
```

```
값을 입력하세요??? asd
asd
```

문자열을 숫자로 변환

2-4. 값의 변환(문자열->숫자)

[18]:
```
x = int('123') # 문자열이지만 숫자라면 int 명령어를 통해 정수형 숫자로 바꿀 수 있음
x
```

[18]:
```
123
```

[19]:
```
x+30
```

[19]:
```
153
```

숫자를 문자열로 변환

2-4. 값의 변환(숫자->문자열)

```
[21]: x =123
      x
```

```
[21]: 123
```

```
[22]: y=str(x)
      y
```

```
[22]: '123'
```

연습문제

- 프로그램을 실행하면 “값을 입력하세요” 라는 안내문이 출력되고 사용자가 숫자를 입력하면 입력한 숫자에 50을 더하여 출력하는 프로그램을 만들어 보시오

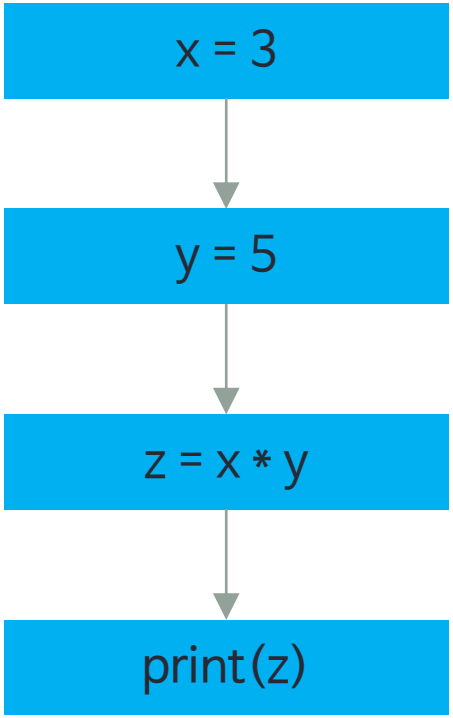
- 프로그램을 실행하면 “값을 입력하세요” 라는 안내문이 출력되고 사용자가 숫자를 입력하면 입력한 숫자에 50을 더하여 출력하는 프로그램을 만들어 보시오

파이썬 문법 기초 (조건문)

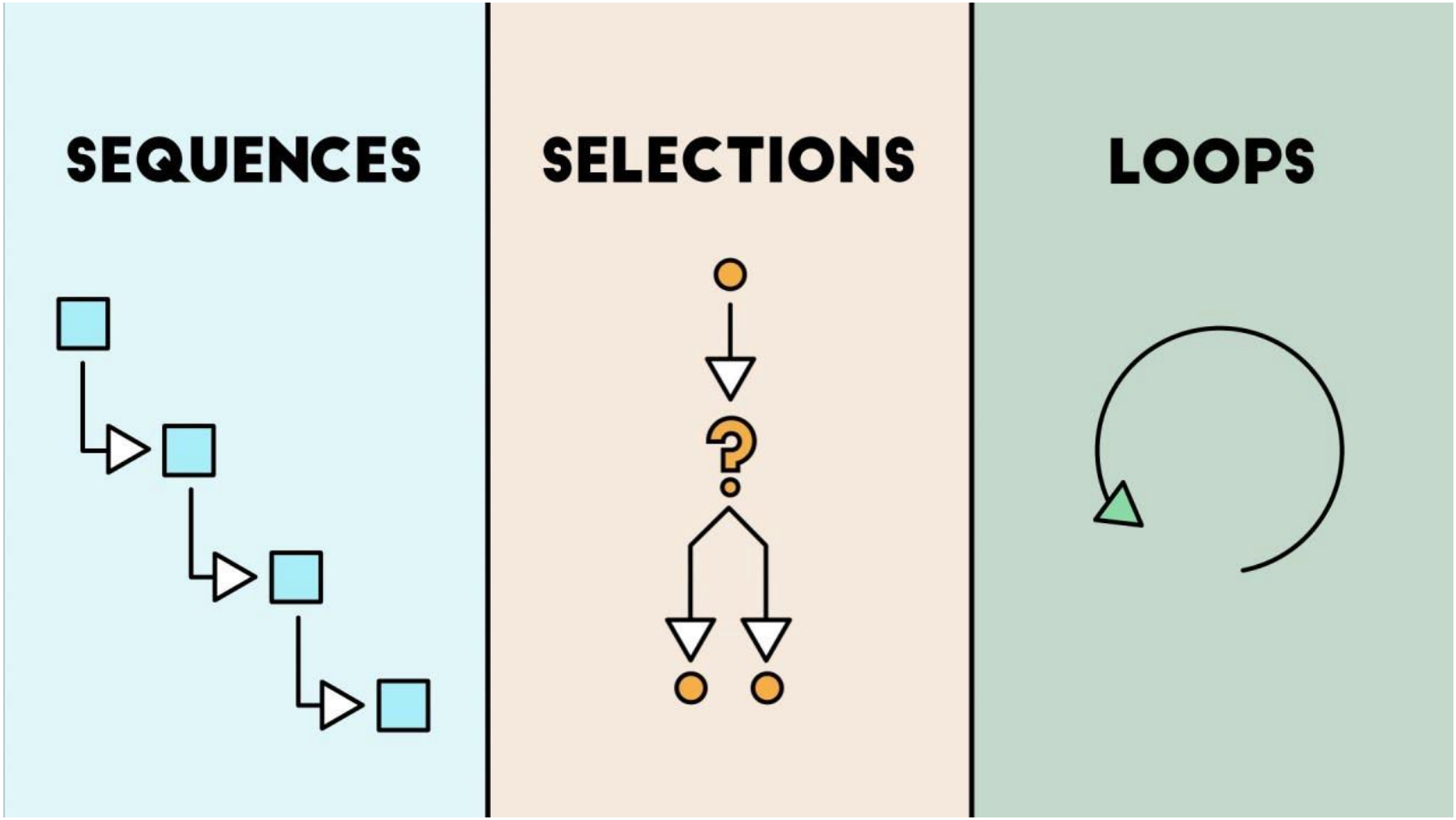
시스템경영공학부
이지환 교수

지금까지 프로그램의 흐름 (sequence)

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

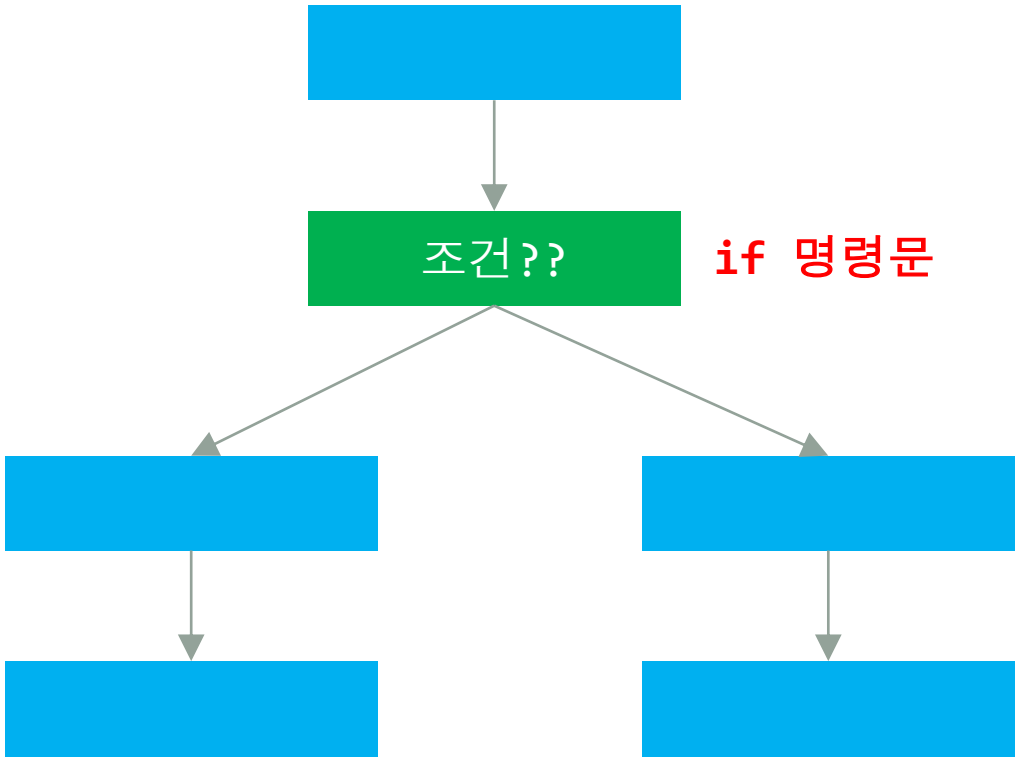


프로그램 코드의 진행



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 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

- 문제1: 프로그램 실행시 사용자에게 입력값을 받아 name이라는 변수에 저장한다. 만약 입력값이 “Bond”라면 “welcome 007”이라는 메시지가, 다른 값이라면 “Good Morning”이라는 메시지가 나오도록 프로그램을 작성해보시오.

- 문제2: 이 프로그램은 사용자에게 숫자값을 입력받는다. 입력받는 값이 짝수라면 “짝수” 라는 메시지가 출력되도록, 홀수라면 “홀수”라는 메시지가 출력되도록 프로그램을 작성해 보시오.
  - hint: 사용자가 입력하는 값은 모두 문자열로 작성된다.

조건문 연습문제

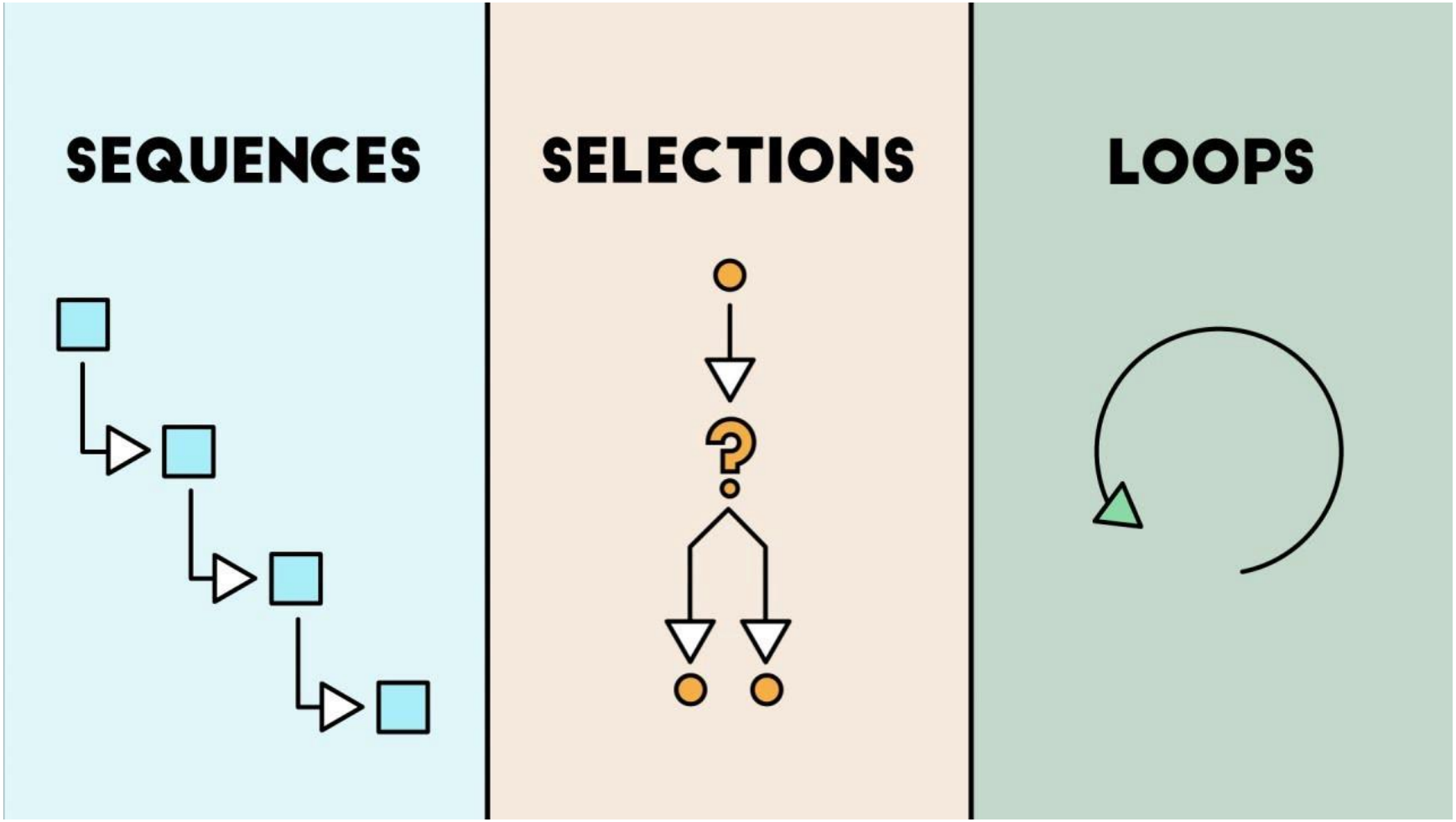
연습 문제 2.5.4

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.

파이썬 문법 기초 (반복문)

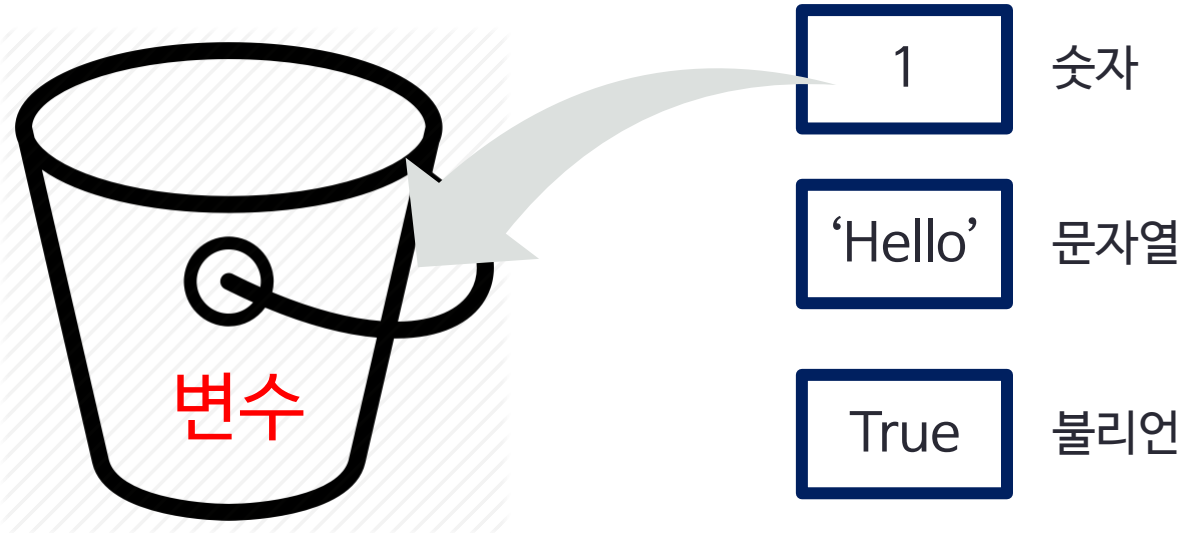
시스템경영공학부
이지환 교수

5-1. 리스트



5-1. 리스트

- 지금까지: 하나의 변수에 한 개의 값만 저장

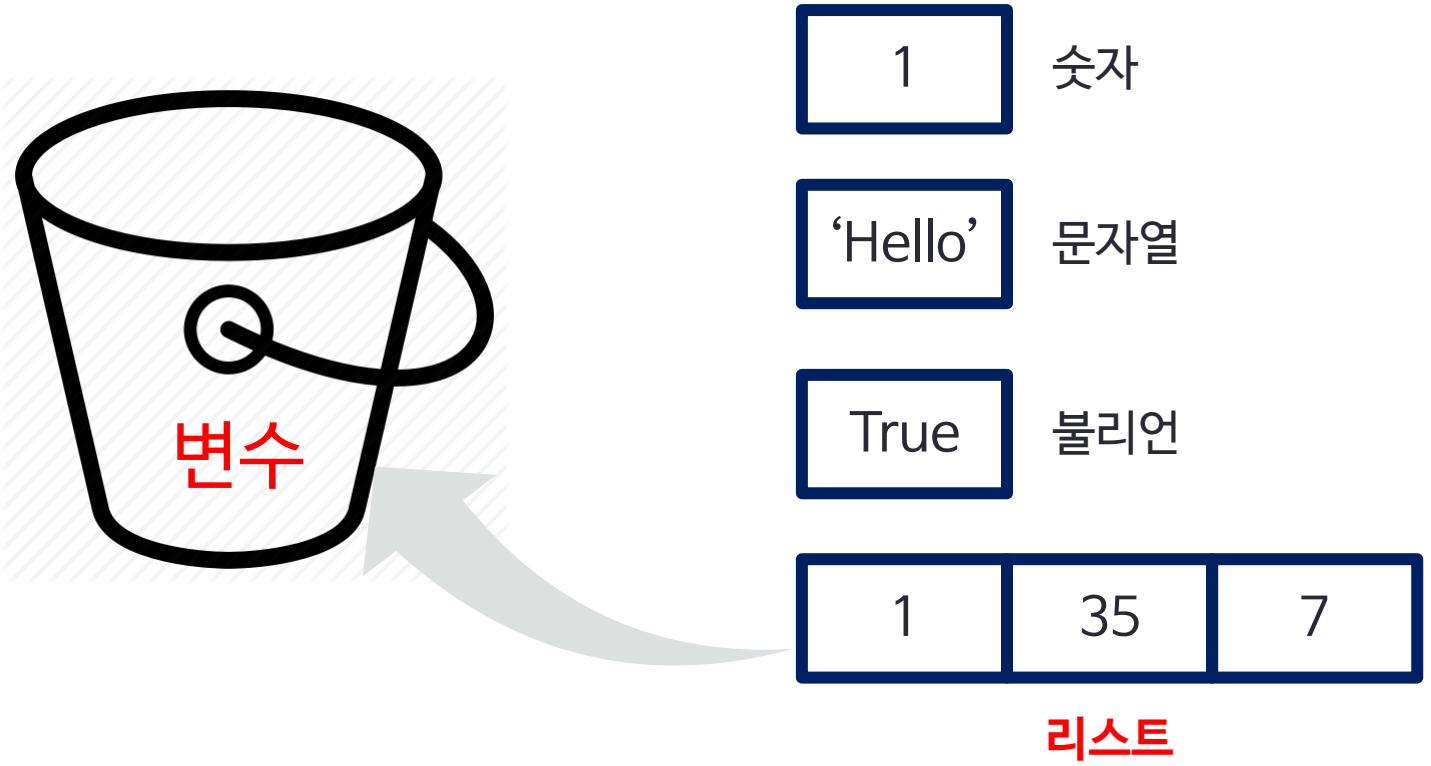


5-1. 리스트

- 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

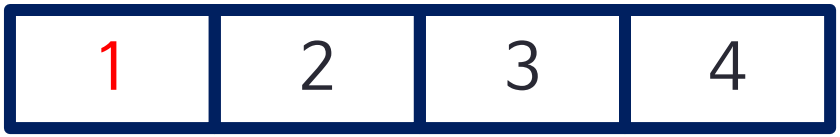
```
for 카운터변수 in 리스트:
    반복할 구문
    반복할 구문
```

[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)
```

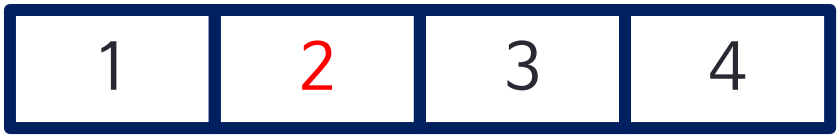


리스트 안의 값을 모두 순회할때까지 반복

첫번째 반복 → i = 1

5-2. for를 이용한 반복

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

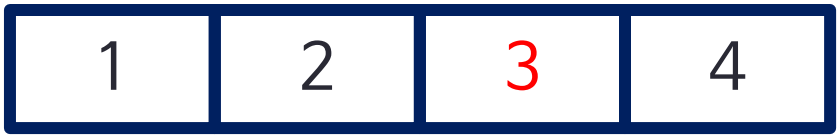


리스트 안의 값을 모두 순회할때까지 반복

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)
```

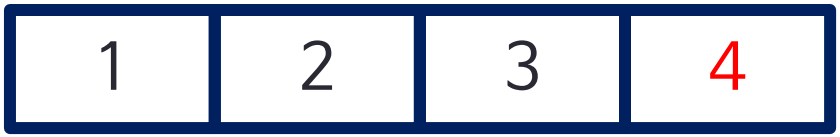


리스트 안의 값을 모두 순회할때까지 반복

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

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. while 명령어

- while 다음 조건을 만족하는 한 코드블럭을 계속 실행

```
while 조건:
    반복할 구문
    반복할 구문
```

5-5. while 명령어

[2]:
```
while True:
    print("hey")
```

...

[3]:
```
while False:
    print("hey")
```

5-5. while 명령어

[4]:
```
i=0
while i<5:
    i=i+1
    print(i)
```

```
1
2
3
4
5
```

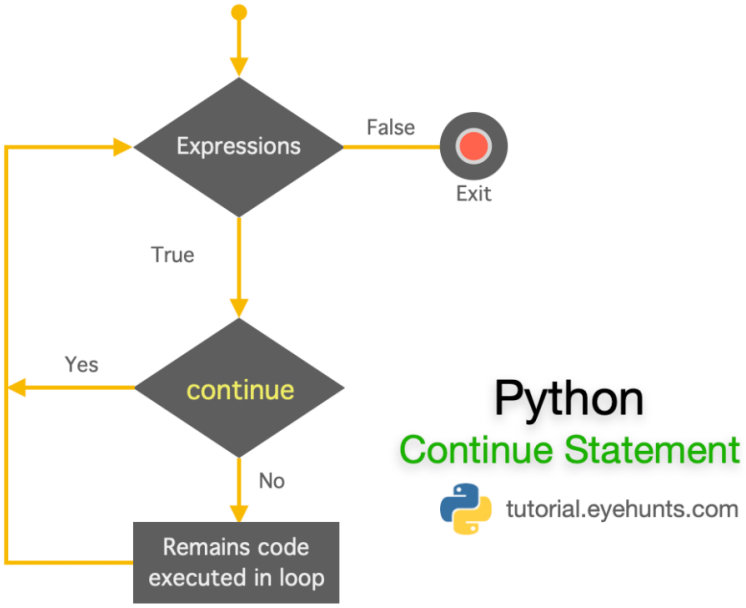
while 연습문제

- 문제1) while을 이용하여 “Hello”를 10번 반복하여 출력하는 프로그램을 만들어 보시오

- 문제2) while을 이용하여 1부터 100까지 숫자의 합을 더하여 그 값을 result에 저장해 보시오

5-5. continue 명령어

- for, while 모두 적용
  - 반복문 코드블럭 내에서 continue를 만나면, 다음부분을 무시하고 다음 반복으로 바로 건너뛰게 됨



5-5. continue 명령어

- for, while 모두 적용
  - 반복문 코드블럭 내에서 continue를 만나면, 다음부분을 무시하고 다음 반복으로 바로 건너뛰게 됨

[17]:
```
for i in range(10):
    if i%2 == 0:
        continue
    print(i)
```

```
1
3
5
7
9
```

[9]:
```
var = 5
while var > 0:
    var = var - 1
    if var == 2:
        continue
    print('Current variable value :', var)
print("Good bye!")
```

```
Current variable value : 4
Current variable value : 3
Current variable value : 1
Current variable value : 0
Good bye!
```

5-6 break 명령어

- for, while 모두 적용
  - 반복문 코드블럭 내에서 break를 만나면, 그 순간 반복루프를 종료함

[10]:
```
for i in range(10):    # 0부터 9999까지 반복
    print(i)
    if i == 3:    # i가 100일 때
        break       # 반복문을 끝냄. for의 제어흐름을 벗어남
```

```
0
1
2
3
```

[12]:
```
i = 0
while True:    # 무한 루프
    print(i)
    i += 1          # i를 1씩 증가시킴
    if i == 12:    # i가 100일 때
        break       # 반복문을 끝냄. while의 제어흐름을 벗어남
```

```
0
1
2
3
4
5
6
7
8
9
10
11
```

연습문제

- while문을 이용하여 다음 프로그램을 만들어보시오
  - 이 프로그램은 0~9사이의 숫자를 맞추는 프로그램이다. 정답 숫자를 answer 변수에 저장되어 있다. 사용자가 입력한 값이 정답이 아니면 “틀렸습니다”가 출력되며, 옳은 정답을 입력하면 “맞았습니다”가 출력되고 프로그램이 끝나도록 설계되어있다. 이 프로그램을 실행해 보시오.

```
입력해보세요: 1
틀렸습니다.
입력해보세요: 2
틀렸습니다.
입력해보세요: 5
틀렸습니다.
입력해보세요: 9
틀렸습니다.
입력해보세요: 3
맞았습니다.
```

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
```

hi

there there there

5-5. 중첩반복문

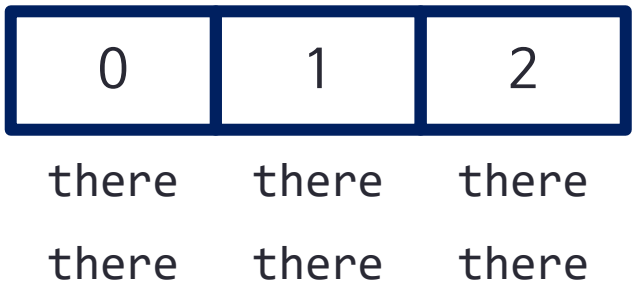
```
for i in range(2):
    print('hi')
    for j in range(3):
        print('there')
```

```
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

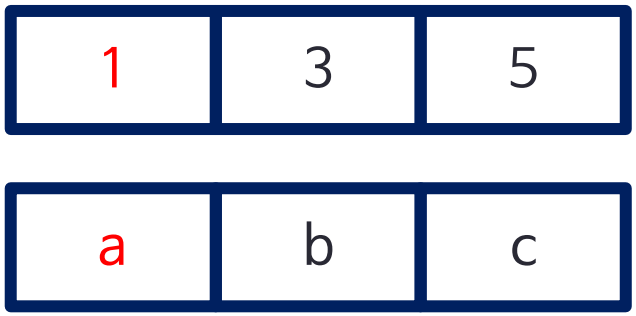
[21]:
```
for i in [1,3,5]:
    for j in ['a','b','c']:
        print(i,j)
```

```
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)
```

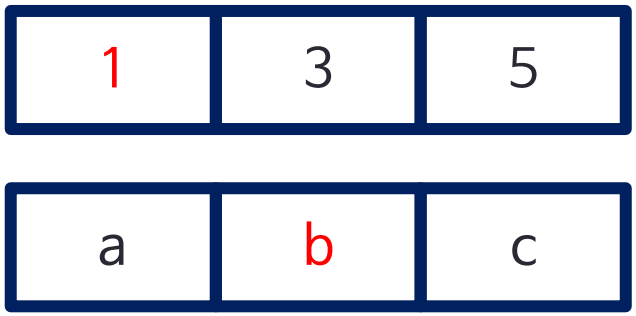
```
1 a
1 b
1 c
3 a
3 b
3 c
5 a
5 b
5 c
```



5-5. 중첩반복문

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

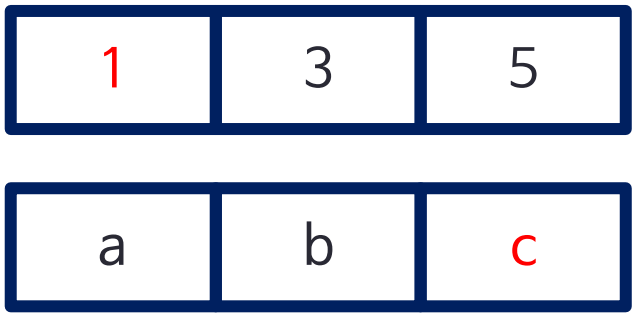
```
1 a
1 b
1 c
3 a
3 b
3 c
5 a
5 b
5 c
```



5-5. 중첩반복문

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

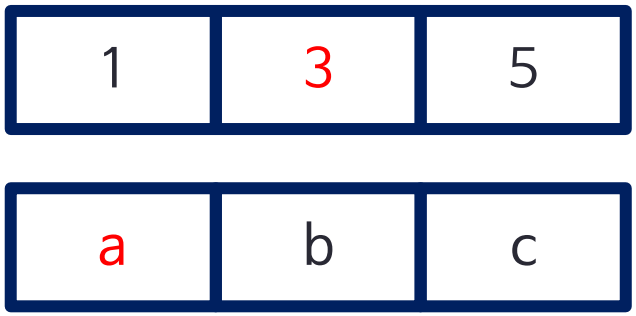
```
1 a
1 b
1 c
3 a
3 b
3 c
5 a
5 b
5 c
```



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

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.

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

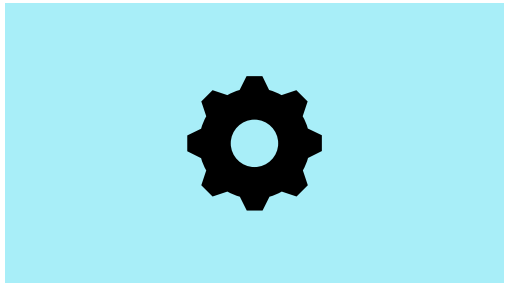
파이썬 문법 기초 (함수)

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.

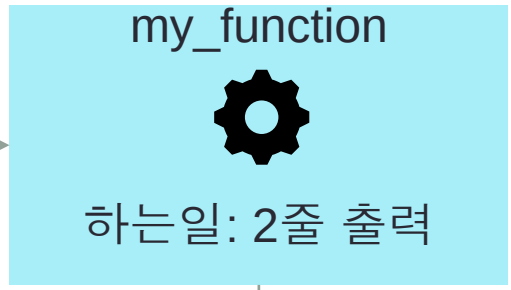
[14]:
```
def my_function():
    print('hello')
    print('world')

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

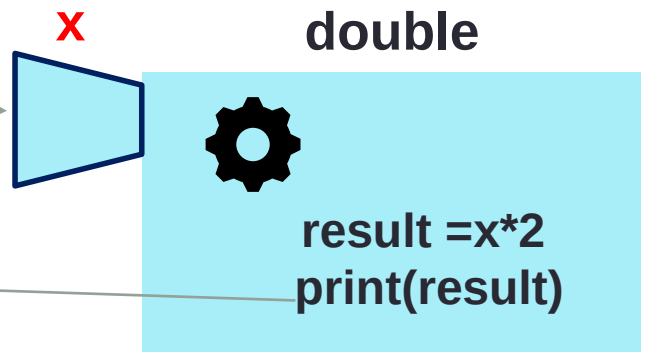
```
[3]: def double(x):
         result = 2*x
         print(result)
```

```
[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

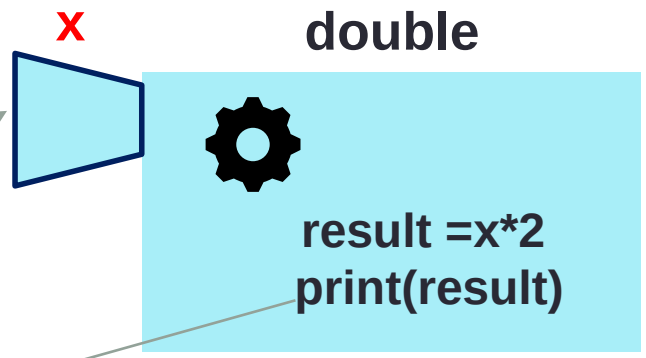
```
[3]: def double(x):
         result = 2*x
         print(result)
```

```
[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

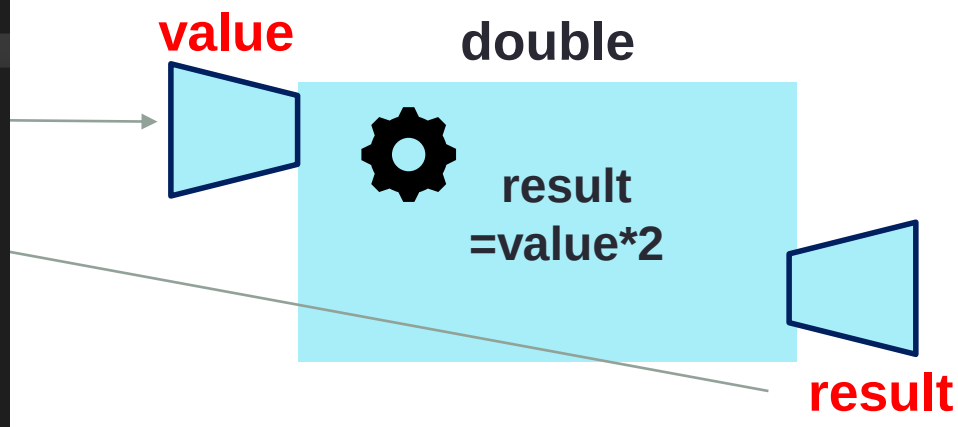
```
def 함수이름(입력변수):
    코드
    코드
    …
    return 출력변수
```

6-3. 출력변수 실습

```
[8]   1 def double(x):
      2     result = 2*x
      3     return result
```

```
1 y = double(2)
2 print(y)
3 y = double(10)
4 print(y)
5 z = double(30)
6 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

6-4. 출력값이 여러 개인 함수

- 출력값을 여러 개 갖도록 할 수 있다.

[1]:
```
def power(a,b):
    return a,a**b

a,b = power(2,3)
```

[2]:
```
print(a,b)
```

```
2 8
```

연습문제

- 입력값 a,b를 순서대로 받아 a를 b로 나눈 몫과 나머지를 각각 반환하는 함수를 만들어보시오.

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.¶

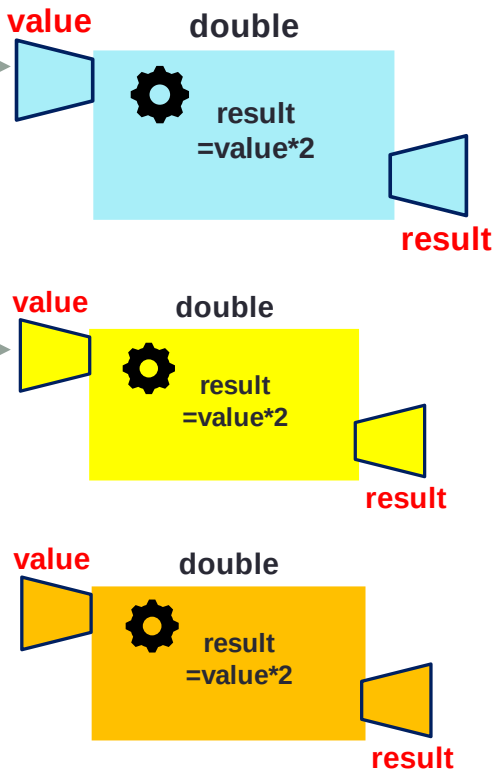
어떤수 n을 입력하여 짝수면 True, 홀수면 False를 입력하는 함수를 만들어 보시오.

In [1]:

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

    ##
    return result
```

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))
```

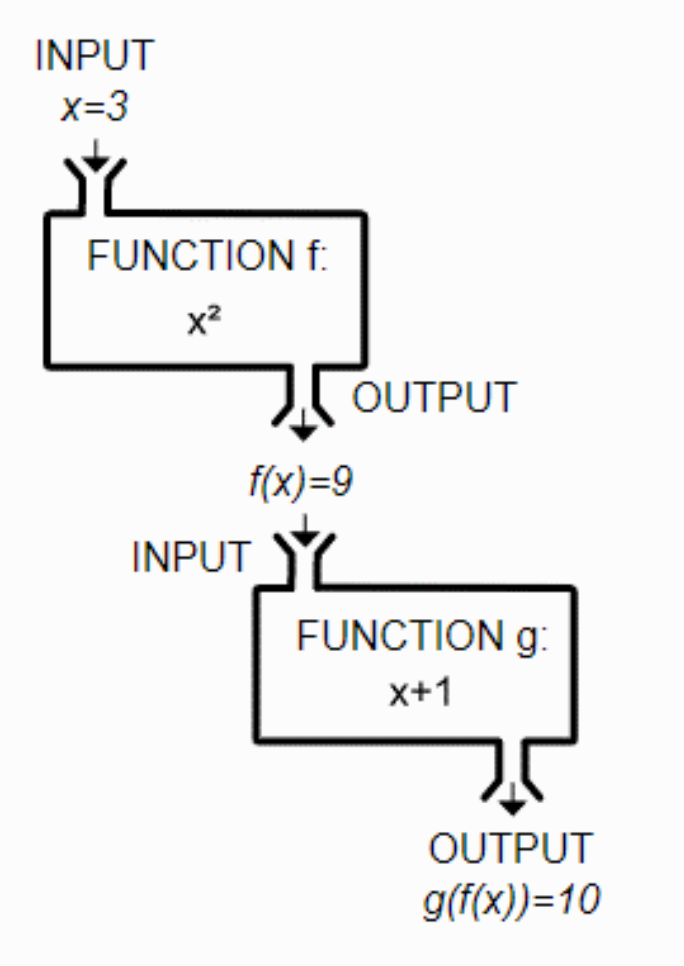
```
14
55
```

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
```

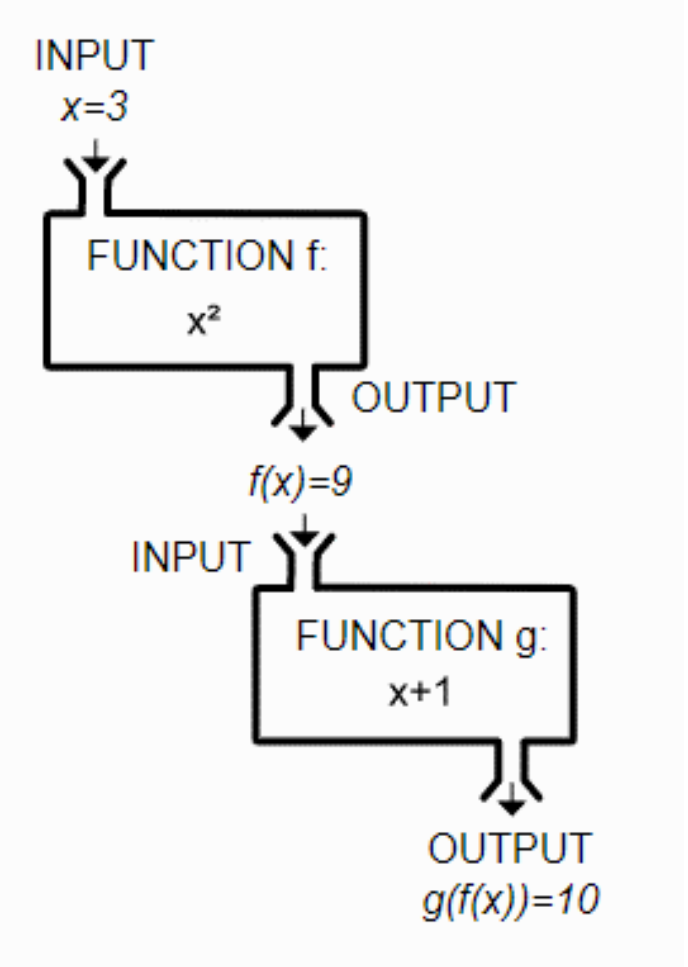
6-9. 함수의 결과값을 입력값으로 사용



함수의 결과값을 입력값으로 사용

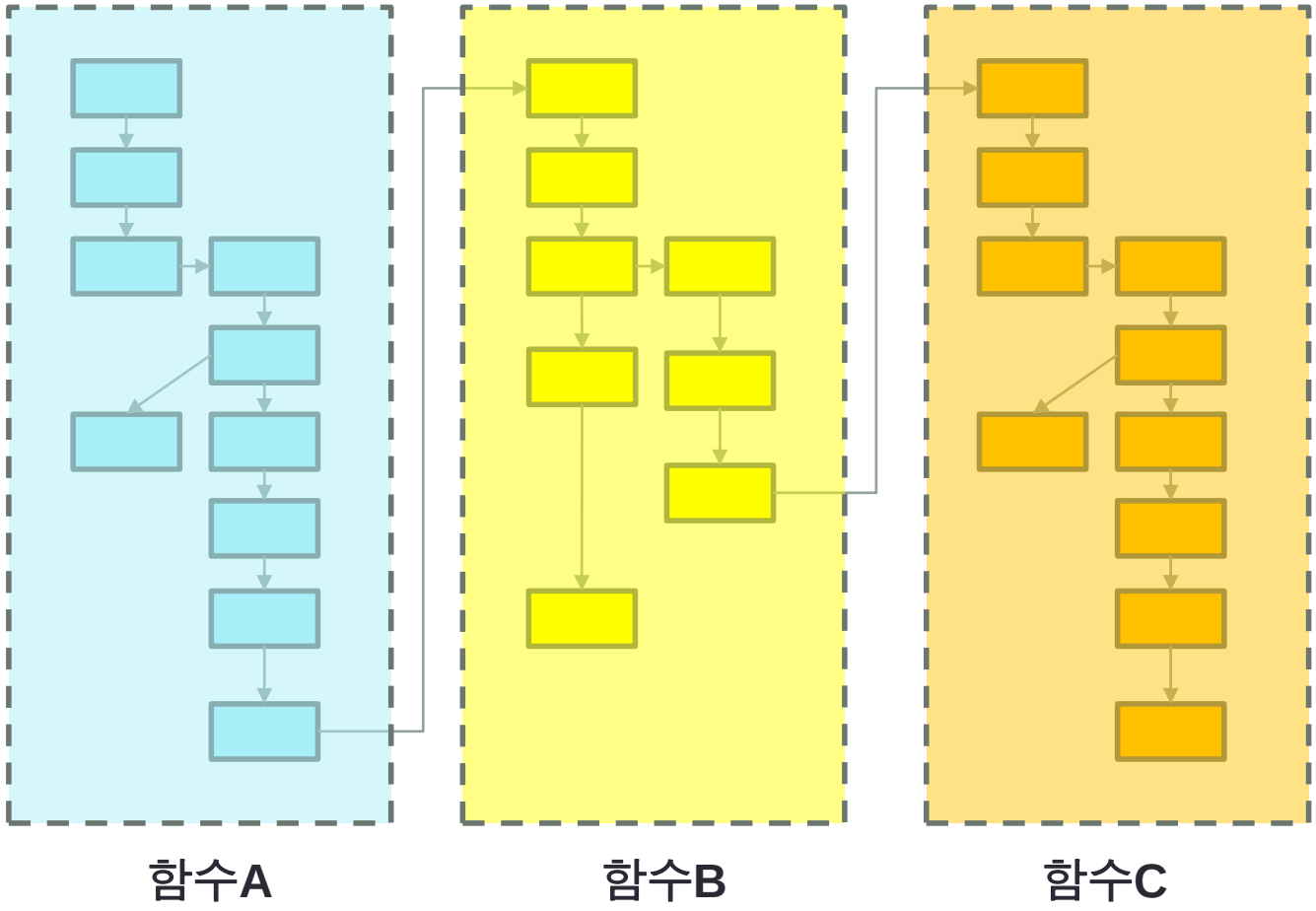
[13]:
```
def f(x):
    return x**2

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]
```

연습문제

- numbers라는 변수에 [1,2,3,4,5,6,7] 이라는 리스트가 저장되어 있다. numbers 내의 모든 값의 제곱값을 갖는 새로운 리스트를 만들어보시오.

  [1,4,9,16,25,36,49]

- 다음과 같은 리스트가 주어져 있을때 7000을 6000 다음에 넣어 다음과 같이 나올수 있도록 해보시오

```
list1 = [10, 20, [300, 400, [5000, 6000], 500], 30, 40]
```

```
[10, 20, [300, 400, [5000, 6000, 7000], 500], 30, 40]
```

연습문제

- 다음과 같은 두개의 리스트가 있다. 두 리스트를 동시에 순회하여 다음과 같이 출력해 보시오

```
list1 = [10, 20, 30, 40]
list2 = [100, 200, 300, 400]
```

```
10 400
20 300
30 200
40 100
```

7-6. 리스트에 값 있는지 확인

```
[18]: 11 in x
```

```
[18]: True
```

```
[19]: 10 in x
```

```
[19]: False
```

```
[20]: if 11 in x:
          print("11 is in x")
```

```
11 is in x
```

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

- 다음 프로그램을 만들어보시오
  - 처음 몇 개의 데이터를 입력받을지 결정
  - 그 후, 순서대로 숫자를 입력받음
  - 입력된 값은 result 리스트에 저장됨
  - 입력완료 후 입력된 리스트를 출력

Hint

Input: 2
Enter element at index 1: 2
Enter element at index 2: 4

Expected output

Result: ['2', '4']

연습문제

- 주어진 리스트에서 중복되는 숫자는 제거하는 프로그램을 만들어보시오
  - hint: in을 이용한다
  - hint: 새로운 리스트를 만든다

Hint

Given num = [2,3,4,5,2,6,3,2]

Expected output

Result: [2, 3, 4, 5, 6]

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

- 두 개의 딕셔너리를 합쳐서 하나의 큰 딕셔너리를 만드는 함수를 작성해 보시오

Hint

Given

dict1 = {'key 1': 2, 'key 2': 3}

dict2 = {'key 3': 4, 'key 4': 5}

Expected output

{'key 1': 2, 'key 2': 3, 'key 3': 4, 'key 4': 5}

연습문제

- **딕셔너리에 각 키에 해당하는 값의 평균값을 계산하는 함수를 만들어보시오**

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.

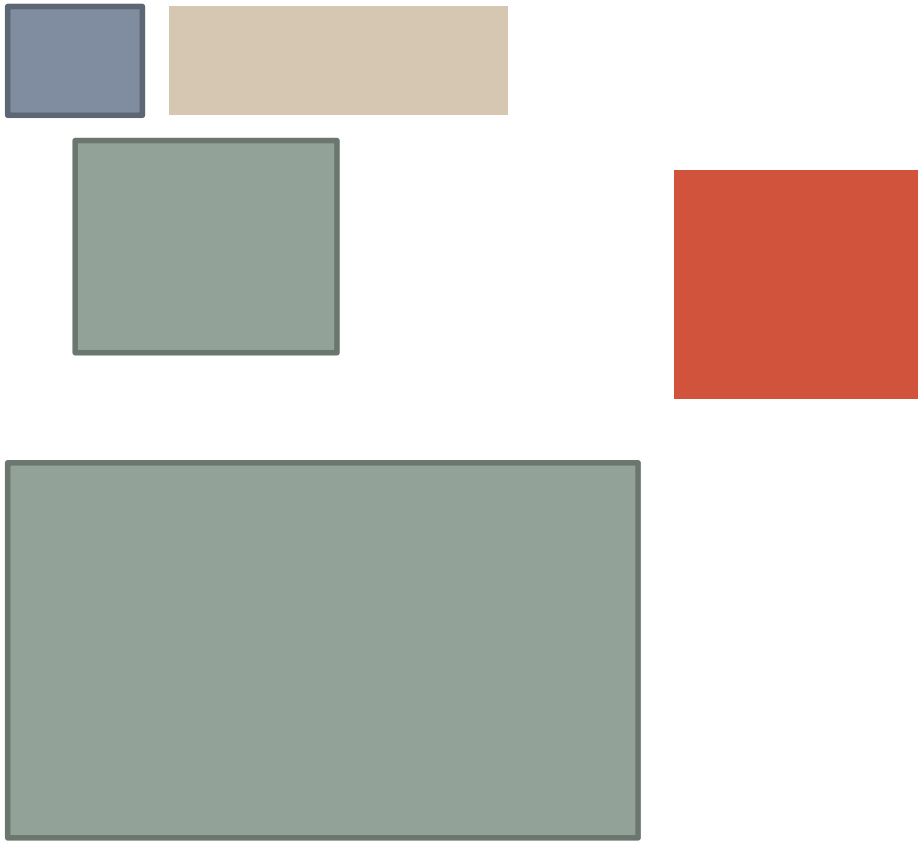
```
철수        98
영희        80
순이       100
돌이        70
================
평균        87
```

파이썬 문법 기초
객체지향 프로그래밍

시스템경영공학부
이지환 교수

9-1. 클래스의 정의

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



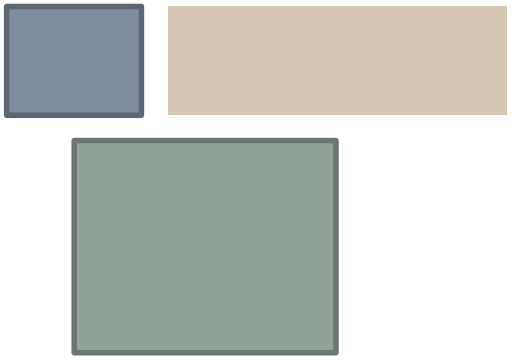
9-1. 클래스의 정의

- 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

Turtle API 살펴보기

- https://docs.python.org/ko/3/library/turtle.html

객체지향 프로그래밍 예시

```
[55]: from ColabTurtlePlus.Turtle import *
      clearscreen()
      setup(500,300)
      t = Turtle() # 새로운 객체 만들기
```

객체지향 프로그래밍 예시

[58]:
```
from ColabTurtlePlus.Turtle import *
clearscreen()
setup(500,300)
t = Turtle() # 새로운 객체 만들기
print(t.position()) #객체의 상태
print(t.heading()) #객체의 상태
print(t.color()) #객체의 상태
```

객체지향 프로그래밍 예시

•[62]:
```
from ColabTurtlePlus.Turtle import *
clearscreen()
setup(500,300)
t = Turtle()
print(t.position()) #객체의 상태
print(t.heading()) #객체의 상태
print(t.color()) #객체의 상태
t.forward(45) # 객체의 행동
print(t.position()) #객체의 상태
print(t.heading()) #객체의 상태
print(t.color()) #객체의 상태
t.left(45) # 객체의 행동
t.forward(45) #객체의 행동
print(t.position()) #객체의 상태
print(t.heading()) #객체의 상태
print(t.color()) #객체의 상태
t.color("red","red") #객체의 행동
t.forward(50) #객체의 행동
print(t.position()) #객체의 상태
print(t.heading()) #객체의 상태
print(t.color()) #객체의 상태
```

```
(0.0, 0.0)
0
('black', 'black')
(45.0, 0.0)
0
('black', 'black')
(76.82, 31.819999999999993)
45
('black', 'black')
(112.17500000000001, 67.175)
45
('red', 'red')
```

객체지향 프로그래밍 예시

```
[63]: def printStatus(turtle):
          print(turtle.position())
          print(turtle.heading())
          print(turtle.color())

      clearscreen()
      setup(500,300)
      t = Turtle()
      printStatus(t)
      t.forward(45)
      printStatus(t)
      t.left(45)
      t.forward(45)
      printStatus(t)
      t.color("red","red")
      t.forward(50)
      printStatus(t)
```

연습문제

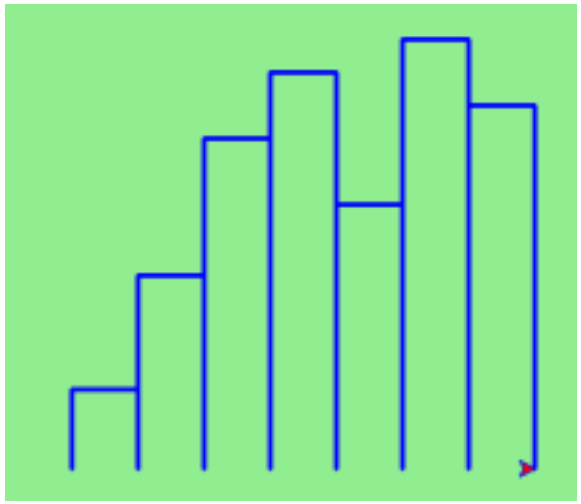
- 높이를 입력하면 높이만큼 다음 그림이 만들어지는 함수를 설계해 보시오
- 너비는 10으로 통일
- 입력값(객체, 높이)

- 예시(turtle, 45)

연습문제

- 다음과 같이 리스트 숫자 값을 입력하면 히스토그램이 그려지는 함수를 설계해보시오.

xs = [48, 117, 200, 240, 160, 260, 220]



파이썬 문법 기초
클래스 만들기

시스템경영공학부
이지환 교수

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.