

CES1017 Programming Fundamentals

# Week 3: Function Abstraction & Application

Instructor: Eunil Park (pa1324@hanyang.ac.kr)



**HANYANG UNIVERSITY**



## Last Class

1. Expression !?
2. 입력 함수
3. 출력 함수
4. 이항연산자
5. Do it for learning

## Today's Schedule

0. Check-up & Overview
1. 문자열과 리스트
2. 시간 단위 변화
3. 조건문
4. Secure Coding
5. While 반복문
6. 함수
7. 사례 학습

## Check-up

- 처음 실습을 한 소감은?
  - 실제로 “코딩을 해 봐야” 완전히 이해가 됨
  - 강의시간에 집중!
- Testing, debugging의 중요성!
  - Unit test 필요!
  - 예:  $a = (r + 3 * (2 - b)) / (4 * 2 - 1)$ 
    - r은 정수? 실수? 문자? 불확실할 땐 type() 활용
    - $(2 - b)$ ,  $(r + 3 * (2 - b))$ ,  $(4 * 2 - 1)$ 가각각 의도대로 계산되는지 체크해야 함
    - 괄호가 많을 때는 괄호가 맞게 사용되었는지 잘 확인해야 함

# 00. Check-up & Overview



## Check-up

- 결과 예측?

```
a = input("a:")  
b = input("b:")  
print(a+b)
```

buur(a+b)

vs.

```
a = int(input("a:"))  
b = int(input("b:"))  
print(a+b)
```

buur(a+b)

a:3

b:4

>>>

vs.

a:3

b:4

>>>

## Today's Goal

- 시간단위를 변환하는 간단한 프로그램 작성
- 안전 코딩 (Secure Coding)의 개념,  
조건문, 반복문, 함수 정의/호출의 기본 개념 학습

# 01. 문자열과 리스트

## 문자열

- 문자, 단어 등으로 구성된 문자들의 집합
- 예:

"Life is too short, You need Python"

"a"

"123"

## 리스트

- 숫자와 문자열만으로 프로그래밍을 하기엔 부족
- 예: 1~10까지 숫자 중 홀수 모음인 1, 3, 5, 7, 9라는 집합이 있을 때 이를 순차적으로 저장하기 위해 사용

### > 리스트(List)

# 01. 문자열과 리스트



HANYANG  
UNIVERSITY

## 리스트 예

- 1, 3, 5, 7, 9라는 숫자 모음

```
>> odd = [1, 3, 5, 7, 9]
```

- 리스트를 만들 때: 대괄호([ ])로 사용
- 각 요소 값: 쉼표(,)로 구분

```
>>> a = []
```

```
>>> b = [1, 2, 3]
```

```
>>> c = ['Life', 'is', 'too', 'short']
```

```
>>> d = [1, 2, 'Life', 'is']
```

```
>>> e = [1, 2, ['Life', 'is']]
```

- 비어있는 리스트 생성: a = list()

# 01. 문자열과 리스트

## 리스트와 문자열 인덱싱

- 리스트/문자열은 인덱싱 적용가능
- 문자열
- 리스트

```
>>> a = "David Beckham"  
>>> |
```

```
>>> a[2]
```

```
>>> a[4]
```

```
>>> a[-1]
```

```
>>> a[-4]
```

```
>>> a = [1, 2, 3, 4]  
>>> a  
[1, 2, 3, 4]
```

```
>>> a[0]
```

```
>>> a[0] + a[2]
```

```
>>> a[-1]
```

```
>>> b = [1, 2, 3, ['a', 'b', 'c']]  
>>> b[-1]
```

```
>>> b[3][0]
```

- 인덱싱(Indexing): 무엇인가를 "가리킨다"는 의미

# 01. 문자열과 리스트

## 리스트/문자열 슬라이싱

- 리스트/문자열은 슬라이싱 적용가능
- 문자열

```
>>> a = "David Beckham"  
>>> |
```

```
>>> b = a[1] + a[8] + a[7]  
>>> b
```

```
>>> a[6:13]
```

```
>>> a[6:]
```

```
>>> a[:5]
```

```
>>> a[:5] + ' Robert Joseph ' + a[6:]
```

- 리스트

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

```
>>> a[:3]
```

```
>>> a[-2:]
```

```
>>> a[2:4]
```

```
>>>
```

```
>>>
```

[3' ↓]



## 02. 시간 단위 변환

### 기본 프로그램: 날짜-분 환산

- Day > Minute 변환
  - 하루? `print(1 * 24 * 60)`
  - 닷새? `print(5 * 24 * 60)`
- 사용자가 원하는 날 수를 입력하는 프로그램

```
days = input('Enter a number : ')
days = int(days)      < Why?
print(days * 24 * 60)
```

buut(qsyls... 5t... eo)

## 02. 시간 단위 변환

### 프로그램의 요구사항

#### 날짜-분 환산 프로그램 : 요구사항 Requirements

날짜day를 분minute으로 환산한다.

입출력은 실행창에서 한다.

|    |                                     |                        |
|----|-------------------------------------|------------------------|
| 입력 | 정수                                  | < 프로그램의 기능             |
| 출력 | 입력을 분으로 환산한 정수<br>단, 입력이 음수인 경우에는 0 | < 입력 조건<br><br>< 출력 조건 |

### 요구사항을 반영한 프로그램

```
days = input('Enter a number : ')
days = int(days)
if days >= 0 :
    print(days * 24 * 60)
else :
    print(0)
```

< if? 무슨 의미일까

< else? 무슨 의미일까

# 03. 조건문

## 문법 및 의미

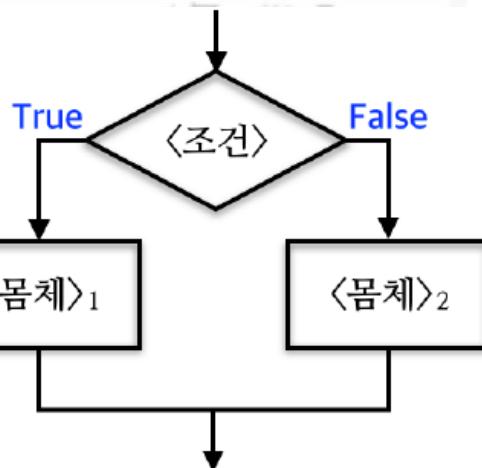
`if <조건> :`

`4칸 <몸체>1`

`else :`

`<몸체>2`

- 문법 (syntax/grammar)
  - <조건>: 논리식
  - <몸체>: 한 줄 이상의 명령문
  - <몸체>는 반드시 일정하게 같은 간격으로 들여쓰기(indentation)을 해야 함. Python은 보통 4칸.



- 의미 (semantics/meaning)
  - <조건>의 논리식 결과가 참(True)이면 <몸체1> 실행, 거짓(False)이면 <몸체2> 실행

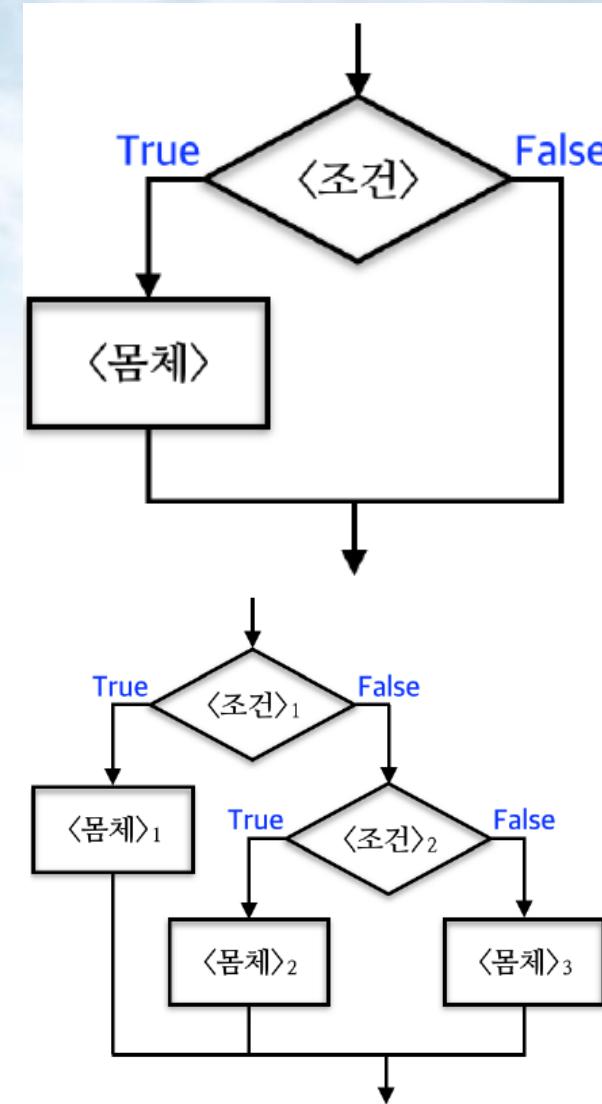
# 03. 조건문



## 문법 및 의미

```
if <조건> :  
    <몸체>
```

```
if <조건식>1 :  
    <몸체>1  
    elif <조건식>2 :  
        <몸체>2  
    else :  
        <몸체>3
```





# 03. 조건문

## 조건문 프로그램 예

```
score = input('Enter your score : ')
score = int(score)
if 90 <= score <= 100 :
    print('A')
elif 80 <= score <= 89 :
    print('B')
elif 70 <= score <= 79 :
    print('C')
elif 60 <= score <= 69 :
    print('D')
elif 0 <= score <= 50:
    print('F')
else:
    print('Invalid input!')
```

입력 스코어

95

69

3

-4

106

입력 스코어  
95? input 69? input 3? input -4? input 106?  
else:  
input(E)

# 04. Secure Coding

## 입력 확인

```
days = input('Enter a number : ')
days = int(days)
if days >= 0 :
    print(days * 24 * 60)
else :
    print(0)
```

input abc?

> 사용자 입력이 프로그램에서 기대하고 있는 형태인가?

## Secure Coding

- 시큐어 코딩(혹은 안전 코딩; Secure Coding)
  - 어떤 기형 불량입력에 대해서도 프로그램이 비정상 종료하지 않고 버틸 수 있도록 코딩하는 것
- 시큐어 코딩의 시작: 상세한 “요구사항” 작성



# 04. Secure Coding

## 상세한 요구사항

### 날짜-분 환산 프로그램 : 요구사항 Requirements

기능      날짜day를 분minute으로 환산한다.

입출력은 실행창에서 한다.

입력      정수

출력      입력을 분으로 환산한 정수  
단, 입력이 음수인 경우에는 0

보안      0 이상의 정수가 아닌 입력은 재입력을 받도록 조치한다.

# 04. Secure Coding

## 수정된 요구사항 및 코드

| 날짜-분 환산 프로그램 : 요구사항 Requirements |  |                                     |
|----------------------------------|--|-------------------------------------|
| 기능                               | 날짜day를 분minute으로 환산한다.<br>입출력은 실행창에서 한다. |                                     |
|                                  | 입력                                       | 정수                                  |
|                                  | 출력                                       | 입력을 분으로 환산한 정수<br>단, 입력이 음수인 경우에는 0 |
| 보안                               | 0 이상의 정수가 아닌 입력은 재입력을 받도록 조치한다.          |                                     |

```
days = input('Enter a number : ')
```

```
while not days.isdigit():
```

 < 0~9 숫자가 아닌 입력이 들어오면 무시하고

```
days = input('Enter a number : ')
```

사용자에게 다시 입력 요청

```
days = int(days)
```

```
if days >= 0 :
```

```
    print(days * 24 * 60)
```

```
else :
```

```
    print(0)
```

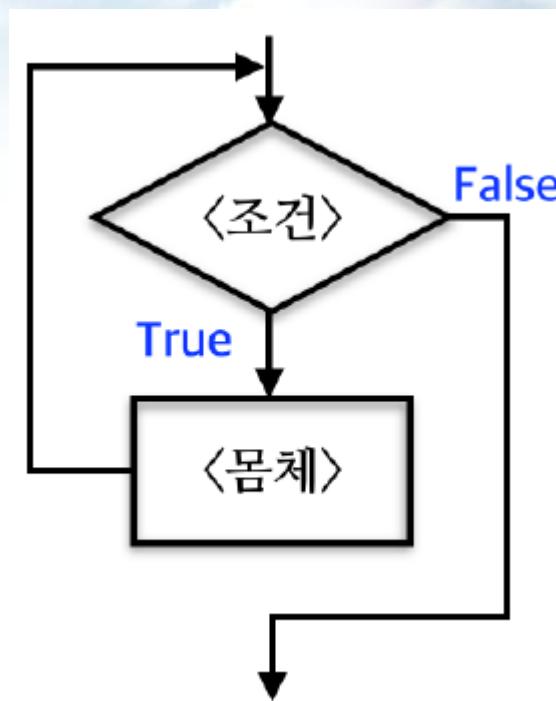
# 05. While 반복문

## 문법 및 의미

- 지정한 조건이 만족하는 동안 동일 블록을 반복(iteration) 실행

```
while <조건> :  
    4칸 <몸체>
```

[syntax]



[semantics]

# 05. While 반복문

## <문자열>.isdigit()

- The Python Standard Library  
(<https://docs.python.org/2/library/index.html>)
  - > 4.7.1 String Methods
  - > isdigit()
- <문자열>.isdigit()  
모두 숫자면 True, 그렇지 않으면 False return

# 05. While 반복문

## 수정된 코드의 해석

```
1 days = input('Enter a number : ')
2 while not days.isdigit():
3     days = input('Enter a number : ')
4 days = int(days)
5 if days >= 0 :
6     print(days * 24 * 60)
7 else :
8     print(0)
```

- 줄2: 줄1에서 입력한 문자열(days)이  
숫자가 아니면(not; isdigit() 결과) → 줄3<몸체> 실행  
숫자가 맞으면(true; isdigit() 결과) → 줄4로 넘어감
- 줄3: days를 입력 받음.  
새로 입력 받은 문자열(days)를 바탕으로 다시 줄2로 돌아감

# 05. While 반복문

## 코드 향상

```
1 days = input('Enter a number : ')
2 while not days.isdigit():
3     days = input('Enter a number : ')
4 days = int(days)
5 print(days * 24 * 60) < 왜 올바르게 동작할까?
```

```
1 print("I change 'days' to 'minutes'.")
2 days = input('Enter the number of days : ')
3 while not days.isdigit():
4     print("It is not 'digit'! Please enter 'digit'!")
5     days = input('Enter the number of days : ')
6 days = int(days)
7 print(days, " days are", days * 24 * 60, " minutes.")
8 print("Thank you for using me!") < 사용자의 편의성 향상
```

# 05. While 반복문

## 입력확인 패턴

```
s = input()  
while not <통과 조건>:  
    s = input()
```

- 통과시킬 우량 입력이 될 조건을 입력문자열 s를 이용하여 논리식을 만들어 <통과조건> 위치에 넣음 (예: s.isdigit())

# 05. While 반복문

서비스의 무한반복

while True:  
<몸체>

- <몸체>를 무한정 반복 실행
- <몸체>를 빠져나가고 싶으면 break 명령 사용

# 05. While 반복문

## 서비스의 무한반복 예

```
1 print("I change 'days' to 'minutes'.")  
2 while True:  
3     days = input('Enter the number of days : ')  
4     while not days.isdigit():  
5         print("It is not 'digit'! Please enter 'digit'!")  
6         days = input('Enter the number of days : ')  
7     days = int(days)  
8     print(days, " days are", days * 24 * 60, " minutes.")  
9     cont = input("Could you want to continue?(Yes/No)")  
10    while not (cont == 'Yes' or cont == 'No'):  
11        cont = input("Could you want to continue?(Yes/No)")  
12    if ( ):  
13        break  
14 print("Thank you for using me!")
```

# 06. 함수

## 함수

- 특정 일을 하는 “부품” 프로그램

## 라이브러리 불박이 함수

- 우리가 언제나 불러 쓸 수 있는 기본 부품
- 표준라이브러리(Python Standard Library)의
  - 2. Built-in Functions에 기술되어 있음
- 예
  - `print`: 내주는 값이 없는 함수(procedure)
  - `input`
  - `int, float, str`

# 06. 함수

## 자작제작함수

- 사용자 정의 함수(user-defined function)
  - 프로그램 안에 같은 코드가 여러 번 중복되면 프로그램을 이해하거나 유지관리 어려워짐
  - 코드의 일부분을 수정해야 하는 경우 중복코드를 모두 일관성 있게 수정하는 것도 쉽지 않음
  - 따라서 중복되는 부분을 “함수”로 만들고 필요할 때마다 불러 쓰면 프로그램의 이해가 쉽고 수정/유지관리도 쉬워짐

# 06. 함수

## 함수의 정의

- 함수 정의(function definition)
  - <함수이름>은 변수 이름 만드는 규칙과 동일
  - 괄호 안 <변수>는 <몸체>에서 사용할 변수이름을 정하는 것으로 **형식 파라미터 (formal parameter)**라고 함. 0개 이상 사용 가능.
  - 함수를 정의할 때 몸체가 실행되지는 않음

```
def <함수이름> ( <변수>1, <변수>2, ⋯, <변수>n ) :  
    <몸체>
```

- 함수에서 만들어낸 정보를 결과로 내주는 것: `return <표현식>`
  - 몸체 끝에 사용
  - <표현식> 결과가 함수 호출의 결과가 됨
  - `return`이 없으면 아무런 값도 내주지 않음. `return`이 없는 함수를 `procedure`라고 함

# 06. 함수

## 함수의 호출

- 함수 호출(function call)
  - 이미 정의 되어있는 함수만 호출 가능
  - 괄호 안에 들어가는 표현식은  
실제 파라미터(actual parameter) 혹은 인수(argument)라고 함
  - 호출을 하면 인수를 계산한 결과 값을 각각 짹을 맞추어  
형식 파라미터 변수에 지정하고 함수의 <몸체>를 실행
  - 인수(실제파라미터) 개수 = 형식파라미터의 개수

〈함수이름〉 ( 〈표현식〉<sub>1</sub>, 〈표현식〉<sub>2</sub>, ⋯ , 〈표현식〉<sub>n</sub> )

# 06. 함수

## 함수 작성 예

- 반지름 r을 입력 받아 원의 면적을 내주는 함수(area\_circle) 정의
  - 계산결과에서 소수점 k번째 자리 아래를 반올림 할 것
    - Round(m, n)함수를 사용하면 소수점 자리 이하 자리 수(n)를 지정하여 m을 반올림할 수 있음

```
def area_circle(r, k):
    import math
    return round(math.pi * r**2, k)
```

# 07. 사례 학습

## 시간단위 변환 프로그램

```
1 print("I change 'days' to 'minutes'.")  
2 while True:  
3     days = input('Enter the number of days : ')  
4     while not days.isdigit():  
5         print("It is not 'digit'! Please enter 'digit'!")  
6         days = input('Enter the number of days : ')  
7     days = int(days)  
8     print(days, " days are", days * 24 * 60, " minutes.")  
9     cont = input("Could you want to continue?(Yes/No)")  
10    while not (cont == 'Yes' or cont == 'No'):  
11        cont = input("Could you want to continue?(Yes/No)")  
12    if (cont == 'No'):  
13        break  
14 print("Thank you for using me!")
```

< 함수로  
만들 부분

print("I change 'days' to 'minutes'.")  
break

if (cont == "No"):

# 07. 사례 학습

## 시간단위 변환 프로그램

```
1 def get_number():
2     days = input('Enter the number of days : ')
3     while not days.isdigit():
4         print("It is not 'digit'! Please enter 'digit'!")
5         days = input('Enter the number of days : ')
6     return int(days)
7
8 print("I change 'days' to 'minutes'.")
9 while True:
10    days = get_number() →
11    print(days, " days are", days * 24 * 60, " minutes.")
12    cont = input("Could you want to continue?(Yes/No)")
13    while not (cont == 'Yes' or cont == 'No'):
14        cont = input("Could you want to continue?(Yes/No)")
15    if (cont == 'No'):
16        break
17 print("Thank you for using me!")
```

사용자로부터

정수 문자열을

입력 받아 정수로

변환하는 함수

실행시 간단하게

get\_number() 호출

호출 결과는

days 변수에 지정됨

# 07. 사례 학습

## 시간단위 변환 프로그램

```
1 def get_number( ):  
2     days = input(message1)  
3     while not days.isdigit():  
4         print(message2)  
5         days = input(message1)  
6     return int(days)  
7  
8 print("I change 'days' to 'minutes'.")  
9 while True:  
10    days = get_number('Enter the number of days : ','It is not 'digit'! Please enter 'digit'!')  
11    print(days, " days are", days * 24 * 60, " minutes.")  
12    cont = input("Could you want to continue?(Yes/No)")  
13    while not (cont == 'Yes' or cont == 'No'):  
14        cont = input("Could you want to continue?(Yes/No)")  
15    if (cont == 'No'):  
16        break  
17    print("Thank you for using me!")
```

함수를 좀더 “일반적”인 목적으로 수정  
입력 시 message1 출력  
숫자 아니면 message 2 출력

# 07. 사례 학습

## 시간단위 변환 프로그램

```
1 def get_number(message1, message2):
2     days = input(message1)
3     while not days.isdigit():
4         print(message2)
5         days = input(message1)
6     return int(days)
7
8 def stop():
9     cont = input("Could you want to continue?(Yes/No)")
10    while not (cont == 'Yes' or cont == 'No'):
11        cont = input("Could you want to continue?(Yes/No)")
12    return (cont == 'No')
13
14 print("I change 'days' to 'minutes'.")
15 while True:
16     days = get_number('Enter the number of days : ','It is not 'digit'! Please enter 'digit'!')
17     print(days, " days are", days * 24 * 60, " minutes.")
18     if stop():
19         break
20 print("Thank you for using me!")
```

사용자로부터 계속할지 말지를  
묻는 함수를 정의

stop()이 True를 return하면, break로 프로그램 종료

# 07. 사례 학습

## 시간단위 변환 프로그램

```
1 def get_number(message1, message2):  
2     days = input(message1)  
3     while not days.isdigit():  
4         print(message2)  
5     days = input(message1)  
6     return int(days)  
7  
8 def stop():  
9     cont = input("Could you want to continue?(Yes/No)")  
10    while not (cont == 'Yes' or cont == 'No'):  
11        cont = input("Could you want to continue?(Yes/No)")  
12    return (cont == 'No')  
13  
14 def day2minute():  
15     print("I change 'days' to 'minutes'.")  
16     while True:  
17         days = get_number('Enter the number of days : ','It is not 'digit'! Please enter 'digit'!')  
18         print(days, " days are", days * 24 * 60, " minutes.")  
19         if stop():  
20             break  
21     print("Thank you for using me!")  
22  
23 day2minute()을 메인 프로시저인 호출해야 프로그램이 실행됨
```

마지막으로

day->minute

변환하는 함수정의

## Summary

1. Check-up & Overview
2. 시간 단위 변화
3. 조건문
4. Secure Coding
5. While 반복문
6. 함수
7. 사례 학습

# Thanks

Week 3: Function Abstraction & Application

Instructor: Eunil Park ([pa1324@hanyang.ac.kr](mailto:pa1324@hanyang.ac.kr))

