3
안전코딩
Secure Coding

1. 사례 : 온도변환

이번에는 온도변환 프로그램 사례를 가지고 좀 더 복잡한 입력확인 방법을 경험해보자.

온도변환 문제 (요구사항)

- 섭씨 온도는 화씨로 화씨 온도는 섭씨로 변환하는 프로그램을 작성한다.
- 입출력은 모두 실행창에서 이루어져야 한다.
- 입력
  - 화씨를 섭씨로 변환할지 (문자열 C 또는 c) 섭씨를 화씨로 변환할지 선택 (문자열 F 또는 f)
  - 온도 (정수)
- 출력
  - 변환한 온도 (소수점 이하는 반올림하여 정수로 표현)
  - 섭씨 또는 화씨 여부

온도변환 알고리즘

1. 서비스의 이름을 다음과 같이 출력한다.
   ```
   섭씨-화씨 온도변환기
   ```
2. 다음과 같은 메시지를 보여주면서 섭씨를 화씨로 바꿀지 화씨를 섭씨로 바꿀지 사용자가 C 또는 F를 선택하게 한다. 소문자도 허용하도록 한다.
   ```
   섭씨 온도를 알고 있으면 C를
   화씨 온도를 알고 있으면 F를 입력해주세요.
   C/F : _
   ```
3. 사용자 입력이 C이면, 다음과 같이 보여주면서 온도 값을 정수로 입력받는다.
   ```
   섭씨를 화씨로 변환해드립니다.
   섭씨 온도를 입력해주세요 : _
   ```
4. 사용자 입력이 F이면, 다음과 같이 보여주면서 온도 값을 정수로 입력받는다.
   ```
   화씨를 섭씨로 변환해드립니다.
   화씨 온도를 입력해주세요 : _
   ```
5. 사용자의 온도 입력을 정수로 바꾼 뒤, 해당 온도 변환 공식을 써서 온도 변환 계산을 수행한다.
6. 사용자가 선택한 메뉴가 C이고 사용자가 온도로 0을 입력했다면, 다음과 같이 프린트하고 프로그램을 종료한다.
   ```
   섭씨 0 도(C)는 화씨 32 도(F) 입니다.
   감사합니다. 또 찾아주세요.
   ```

온도변환 프로그램 (기능우선코딩)

```
# 입력
print('섭씨-화씨 온도변환기')
print('섭씨 온도를 알고 있으면 C를')
print('화씨 온도를 알고 있으면 F를 입력해주세요.')
base = input('C/F : ')
if base == 'C' or base == 'c':
    print('섭씨를 화씨로 변환해드립니다.')
    tempin = input('섭씨 온도를 입력해주세요 : ')
elif base == 'F' or base == 'f':
    print('화씨를 섭씨로 변환해드립니다.')
    tempin = input('화씨 온도를 입력해주세요 : ')
print()
# 계산 및 출력
tempin = int(tempin)
if base == 'C' or base == 'c':
    tempout = 9.0 / 5.0 * tempin + 32
    print('섭씨', tempin, '도(C)는 화씨', round(tempout), '도(F) 입니다.')
elif base == 'F' or base == 'f':
    tempout = (5 * tempin - 160) / 9.0
    print('화씨', tempin, '도(F)는 섭씨', round(tempout), '도(C) 입니다.')
print('감사합니다. 또 찾아주세요.')
```

이 프로그램을 이해한 다음 실행해보고, 요구사항 대로 구현 되었는지 다양한 입력 사례를 가지고 시험해보자. 시험해보면 알겠지만 이 프로그램은 요구사항에 제시한 우량 입력에 대해서 잘 작동하며 정확하게 변환 값을 계산하여 출력 해준다. 따라서 기능상으로 완벽한 프로그램이다. 그런데 입력 요구사항을 무시한 불량 입력에 대해서는 제대로 작동하지 않고 오류가 발생하며 프로그램이 비정상적으로 멈춘다.

그러면 입력 요구사항을 무시한 불량입력에 대해서 프로그램이 견딜 수 있도록 입력확인 코드를 채워넣어보자.

온도변환 프로그램에 입력확인 추가 (안전코딩)

첫 입력이 C/c/F/f가 아닌 경우 tempin 변수가 지정되지 않아 오류가 발생한다. 따라서 입력이 C/c/F/f가 아닌 경우 재입력을 받도록 입력확인을 하여 tempin 변수에 항상 C 또는 F의 값만 지정되도록 해보자. while 문을 써서 다음과 같이 간단히 입력확인 할 수 있다.

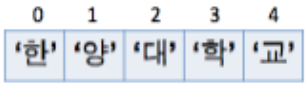
```
base = input('C/F : ')
while not (base == 'C' or base == 'c' or base == 'F' or base == 'f'):
    base = input('C/F : ')
```

다음 입력은 온도인데 정수 문자열만 허용해야 한다. 정수 문자열은 어떤 모양일까? 숫자로만 구성되어 있으면 정수이다. 그런데 앞에 음수임을 나타내는 - 기호가 붙을 수도 있다. 경우에 따라서 양의 정수임을 강조해주는 + 기호가 붙을 수도 있다. 숫자로만 구성되어 있는지 확인하는 메소드는 이미 공부한

대로 `isdigit()` 이다. 앞에 음양을 표시하는 기호가 붙어도 정수이므로 입력확인에서 통과시켜야 한다. 문자열의 구조와 문자열 조작방법을 먼저 공부한 다음, 정수 문자열 입력확인 방법을 알아보자.

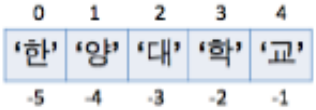
문자열의 위치번호

문자열(string)은 문자를 일렬로 나열한 일차원 벡터로 볼 수 있으며, 각 문자는 차례로 **위치번호(index)**가 매겨져 있으며, 위치번호는 항상 0부터 시작한다. 즉, 문자열 '한양대학교'는 다음과 같이 문자별로 위치번호가 매겨진다.



위치번호는 맨 왼쪽 문자부터 `0`으로 시작하여 `1`씩 증가하도록 매긴다.

음수로 위치번호를 지정할 수도 있다. 음수 위치번호는 맨 오른쪽 문자부터 `-1`로 시작하여 `1`씩 감소하도록 매긴다.



문자열의 길이가 `n`이면, 가용 위치번호 범위는 `0 ~ n-1` 과 `-n ~ -1` 이다. 범위에 벗어나는 위치번호를 사용하면 `IndexError`로 오류 처리된다.

문자열 조각 복제

위치번호를 주고 해당 위치의 문자를 복제하여 만들 수 있다. 문법과 의미는 다음과 같다.

문법	〈문자열〉 [〈정수식〉]
의미	〈문자열〉을 계산한 결과 문자열을 `s`라고 하고, 〈정수식〉을 계산한 결과 값을 `i`라고 하면, `s`의 `i`번째 위치번호에 저장된 문자로 구성된 문자열

다음을 실행창에서 한줄 씩 실행하여 결과를 확인해보자.

```
'SMaSH'[0]
name = 'SMaSH'
name[4]
name[5]
name[-1]
name[-4]
name[-6]
name[1.0]
name[1+1]
```

위치번호의 범위를 주고 그 범위에 해당하는 문자열의 일부를 복제할 수도 있다. 문법과 의미는 다음과 같다.

문법	〈문자열〉 [〈정수식〉$_1$: 〈정수식〉$_2$]

의미	〈문자열〉을 계산한 결과 문자열을 s라고 하고, 〈정수식〉$_1$을 계산한 결과 값을 begin 이라고 하고, 〈정수식〉$_2$을 계산한 결과 값을 end 라고 하면, 문자열 s에서 begin 위치에서 시작하여 end-1 위치까지만 남기고 나머지는 잘라낸 문자열

〈정수식〉$_1$ 을 비워두면 0으로 자동설정되고, 〈정수식〉$_2$을 비워두면 문자열 s에의 길이인 len(s)로 자동설정된다. 따라서 s[:]와 같이 양쪽다 비워두면 문자열 전체를 나타내는 셈이 되어 s와 같다.

다음을 실행창에서 한줄 씩 실행하여 결과를 확인해보자.

```
name = 'SMaSH'
name[3:5]
name[3:]
name[:2]
name[:]
name[:2] + 'A' + name[3:]
name[:-3] + 'A' + name[-2:]
```

문자열은 일단 만들면 **수정불가**(immutable)하다. 즉, 위치번호 또는 위치번호 범위를 주고 문자열 조각을 복제하면 기존의 문자열은 그대로 있고, 지정한 조각 만큼의 문자열이 새로 생긴다.

온도변환 프로그램에 입력확인 추가 (안전코딩) - 계속

이제 정수 입력확인을 하는 코드를 작성할 준비가 되었다. 다음 조건 중에서 하나 만이라도 만족하면 정수이므로 입력을 다시 받을 필요가 없다.

- 숫자로만 구성되어 있다.
- 앞에 + 기호가 나온 다음, 나머지는 숫자로만 구성되어 있다.
- 앞에 - 기호가 나온 다음, 나머지는 숫자로만 구성되어 있다.

이 조건을 각각 다음과 같이 논리식으로 작성할 수 있다. 입력 문자열은 tempin에 저장되어 있다고 하자

- `tempin.isdigit()`
- `tempin[0] == '+' and tempin[1:].isdigit()`
- `tempin[0] == '-' and tempin[1:].isdigit()`

셋 중에 하나만 만족하면 되므로 or로 연결하면 다음과 같이 통과조건이 완성된다.

```
tempin.isdigit() or tempin[0] == '+' and tempin[1:].isdigit() or tempin[0] == '-' and
tempin[1:].isdigit()
```

이 조건식은 길어서 줄이 넘어가서 읽기 힘들다. 하나의 긴 표현식을 여러 줄에 걸쳐서 보기좋게 표현하려면, 다음과 같이 역슬래시를 써서 다음 줄에 식이 계속된다는 표시를 해야 한다.

```
tempin.isdigit() or \
tempin[0] == '+' and tempin[1:].isdigit() or \
tempin[0] == '-' and tempin[1:].isdigit()
```

이제 통과조건이 완성되었으므로 while 문을 써서 이 통과조건을 만족하지 않으면 재입력받도록 하면 된다. 입력확인을 완성한 프로그램은 다음과 같다. 입력확인하는 부분은 진한색으로 표시되어 있으니 확인해보자.

```
# 입력
print('섭씨-화씨 온도변환기')
print('섭씨 온도를 알고 있으면 C를')
print('화씨 온도를 알고 있으면 F를 입력해주세요.')
base = input('C/F : ')
while not (base == 'C' or base == 'c' or base == 'F' or base == 'f'):
    base = input('C/F : ')
if base == 'C' or base == 'c':
    print('섭씨를 화씨로 변환해드립니다.')
    tempin = input('섭씨 온도를 입력해주세요 : ')
    while not (tempin.isdigit() or \
               tempin[0] == '+' and tempin[1:].isdigit() or \
               tempin[0] == '-' and tempin[1:].isdigit()):
        tempin = input('섭씨 온도를 입력해주세요 : ')
elif base == 'F' or base == 'f':
    print('화씨를 섭씨로 변환해드립니다.')
    tempin = input('화씨 온도를 입력해주세요 : ')
    while not (tempin.isdigit() or \
               tempin[0] == '+' and tempin[1:].isdigit() or \
               tempin[0] == '-' and tempin[1:].isdigit()):
        tempin = input('화씨 온도를 입력해주세요 : ')
print()
# 계산 및 출력
tempin = int(tempin)
if base == 'C' or base == 'c':
    tempout = 9.0 / 5.0 * tempin + 32
    print('섭씨', tempin, '도(C)는 화씨', round(tempout), '도(F) 입니다.')
elif base == 'F' or base == 'f':
    tempout = (5 * tempin - 160) / 9.0
    print('화씨', tempin, '도(F)는 섭씨', round(tempout), '도(C) 입니다.')
print('감사합니다. 또 찾아주세요.')
```

이제 프로그램이 꽤 길어졌다. 다음에는 함수를 만들어 프로그램을 더 읽기 쉽고 관리하기 쉽게 만드는 과정을 공부해보자.

온도변환 프로그램 - 함수만들기

이제 **강의비디오** 3-2를 시청하면서 함수를 만드는 과정을 구경하자. 다음은 최종 완성된 프로그램이다. 이 프로그램을 다시 한번 숙독하고 완전히 이해했는지 확인하자.

```
def input_base():
    print()
    print('섭씨 온도를 알고 있으면 C를')
    print('화씨 온도를 알고 있으면 F를 입력해주세요.')
    base = input('C/F : ')
    while not (base == 'C' or base == 'c' or base == 'F' or base == 'f'):
        base = input('C/F : ')
    return base

def isdigit_signed(s):
    return s.isdigit() or \
           s[0] == '+' and s[1:].isdigit() or \
           s[0] == '-' and s[1:].isdigit()

def input_temparature(fro,to):
    print(fro,'를',to,'로 변환해드립니다.')
    tempin = input(fro + '온도를 입력해주세요 : ')
    while not (isdigit_signed(tempin)):
        tempin = input(fro + '온도를 입력해주세요 : ')
    return tempin

def c2f(c): return 9.0 / 5.0 * c + 32

def f2c(f): return (5 * f - 160) / 9.0

def stop():
    cont = input('계속하시겠습니까? (y/n) ')
    while not (cont == 'y' or cont == 'n'):
        cont = input('계속하시겠습니까? (y/n) ')
    return cont == 'n'

```

```
32 def tempconv():
33     print('섭씨-화씨 온도변환기')
34     while True:
35         base = input_base()
36         if base == 'C' or base == 'c':
37             tempin = input_temparature('섭씨','화씨')
38         elif base == 'F' or base == 'f':
39             tempin = input_temparature('화씨','섭씨')
40         tempin = int(tempin)
41         if base == 'C' or base == 'c':
42             print('섭씨', tempin, '도(C)는 화씨',
43                   round(c2f(tempin)), '도(F) 입니다.')
44         elif base == 'F' or base == 'f':
45             print('화씨', tempin, '도(F)는 섭씨',
46                   round(f2c(tempin)), '도(C) 입니다.')
47         if stop():
48             break
49     print()
50     print('감사합니다. 또 찾아주세요.')
51
52 tempconv()
```

위 프로그램의 마지막 줄 52 에서 주 함수인 `tempconv()` 를 호출하면 이 프로그램은 작동을 시작한다.

실습 문제에 쓸 지식

문자열 메소드 : partition(〈분리문자열〉)

partition 메소드는 〈분리문자열〉을 중심으로 문자열을 세 조각으로 나눈다. 예를 들어, 대상 문자열이 "3.14159"이고, 〈분리문자열〉이 "."인 경우, "3.14159".partition(".")를 실행하면 ("3", ".", "14159")와 같이 〈분리문자열〉 를 중심으로 3조각으로 나눈 뭉치를 내준다. 이 조각 뭉치를 **튜플(tuple)**이라고 하는데, 튜플도 문자열과 같은 요령으로 위치번호(index) 0, 1, 2를 사용하여 조각으로 분리할 수 있다. 다음을 실행하여 확인해보자.

```
pi = "3.14159".partition(".")
print(pi)
print(pi[0])
print(pi[1])
print(pi[2])
```

만약 분리문자열이 대상 문자열에 없으면, 첫째 조각에 대상 문자열 전체, 둘째와 셋째 조각에는 빈문자열을 들어있는 세조각 뭉치 튜플을 내준다. 예를 들어 분리문자열이 "."이고, 대상 문자열이 "365"이면, ("365", "", "")와 같이 3조각으로 나눈다. 다음도 실행하여 확인해보자.

```
year = "365".partition(".")
print(year)
print(year[0])
print(year[1])
print(year[2])
```

실습 문제 1. 정수 입력확인

다음 get_int 함수는 정수 입력을 받아서 확인 후 정수로 변환하여 내주는 함수이다.

```
def get_int(message) :
    s = input(message)
    while not s.isdigit() :
        s = input(message)
    return int(s)
```

이 함수를 다양한 입력을 가지고 호출하여 써보면서 이해하자.

그리고 나서, + 또는 - 부호를 앞에 붙인 정수도 허용하도록 개선한 get_int_signed 함수를 다음과 같이 작성하였다.

```
def get_int_signed(message) :
    s = input(message)
    while not (remove_sign(s).isdigit()):
        s = input(message)
    return int(s)
```

앞에서 공부한 방법과는 다르게 입력 문자열 맨 앞에 있는 부호를 떼어버린 다음, 남은 문자열이 숫자로만 구성되어 있는지 확인하는 방식으로 입력확인을 하도록 remove_sign 함수를 만들어보자.

실습 문제 2. 정수+실수 입력확인

다음 isfloat 함수는 문자열 인수 s가 소수점이 있는 실수 형태인지 확인하는 함수이다.

```
def isfloat(s) :
    (m,_,n) = s.partition(".")
    return (m.isdigit() and (n.isdigit() or n == "")) or m == "" and n.isdigit()
```

이 함수를 다양한 입력을 가지고 호출하여 써보면서 어떤 형태의 실수를 확인하는지 이해하자.

다음 get_float 함수는 정수 또는 소숫점이 있는 실수 형태로 입력을 받아서 확인 후 부동소수점수(float)로 변환하여 내주는 함수이다.

```
def get_float(message) :
    s = input(message)
    while not (s.isdigit() or isfloat(s)) :
        s = input(message)
    return float(s)
```

"+" 또는 "-" 부호를 앞에 붙인 정수 또는 고정소수점수도 허용하도록 get_fixed 함수를 개선한 get_fixed_signed 함수를 빈칸을 채워 완성하라.

```
def get_fixed_signed(message) :
    s = input(message)

    while
        s = input(message)

    return float(s)
```

- 귀띔 : 실습문제 1에서 만든 함수도 사용해도 좋다.

실습 문제 3. 안전한 제곱근 함수

표준입력창에서 정수 또는 실수를 입력받아 제곱근을 구하여 표준출력창에 프린트하는 프로시저 safe_sqrt() 를 작성하자. safe_sqrt() 프로시저를 호출하면 표준입출력창을 통하여 사용자와 대화식으로 프로그램이 아래와 같이 작동한다. 보통 폰트로 된 부분은 컴퓨터가 사용자에게 보여주는 부분이고, 파란색으로 볼드체로 진하게 보이는 부분이 사용자가 키보드로 입력한 부분이다.

```
제곱근을 구해드립니다.
0이상의 수를 입력하세요.
수를 입력하세요.
4
4 의 제곱근은 2.0 입니다.
계속하시겠습니까? (y/n) Y
계속하시겠습니까? (y/n) y
수를 입력하세요.
4.8
4.8 의 제곱근은 2.1909 입니다.
계속하시겠습니까? (y/n) 예
계속하시겠습니까? (y/n) y
수를 입력하세요.
-45
수를 입력하세요.
+38
수를 입력하세요.
twenty seven
수를 입력하세요.
0
0 의 제곱근은 0.0 입니다.
계속하시겠습니까? (y/n) n
안녕히 가세요.
```

사용자의 키보드 입력은 확인하여 0이상의 정수와 실수만 통과시키고 나머지는 재입력하도록 한다. 위의 사례를 보면 4, 4.8, 0은 제곱근 계산이 가능한 정수와 실수이므로 입력확인을 통과하여 제곱근 계산을 하여 결과를 보여준다. 그러나 -45, +38, twenty seven은 허용하는 입력이 아니므로 재입력을 요청한다. 한번 계산이 끝나면 계속할지를 물어본다. y를 입력하면 계속하고, n를 입력하면 프로그램을 종료한다.

코딩 가이드

- 제곱근 계산은 표준 라이브러리 math 모듈의 sqrt 함수를 사용한다.
- 계산한 제곱근은 실행사례에서 보여주는 대로 소수점 4째자리 미만은 반올림하여 표준출력창에 프린트한다.
- 강의 시간 및 실습 시간에 완성한 isfloat() 함수와 stop() 함수를 사용하여야 한다.

기 제작 함수

```
def isfloat(s) :
    (m,_,n) = s.partition(".")
    return m.isdigit() and (n.isdigit() or n == "") or \
           m == "" and n.isdigit()

def stop():
    cont = input('계속하시겠습니까? (y/n) ')
    while not (cont == 'y' or cont == 'n'):
        cont = input('계속하시겠습니까? (y/n) ')
    return cont == 'n'
```

완성해야 할 함수

```
def safe_sqrt():
    import math
    print("제곱근을 구해드립니다.")
    print("0이상의 수를 입력하세요.")
    while True:









    print("안녕히 가세요.")
```