테크 지식 & 파이썬 프로그래밍 기초

4주차
조건문과 반복문

20 . .

지은이: 이은재

우리는 컴퓨터 프로그램이 처리할 수 있는 작업의 세 가지 주요한 특징이 순차, 선택, 반복이라는 것을 이미 체험해보았습니다. 지금까지는 가장 기본이 되는 순차 프로그램을 작성하기 위한 문법을 공부한 셈입니다. 처음 프로그래밍을 배운 사람이라면 이정도만 공부해도 신기함과 흥미를 느꼈을지도 모르겠습니다. 그러나 인간이 컴퓨터 프로그램을 만들어 사용하는 더 핵심적인 이유는 선택과 반복 기능이 아주 강력하고 편리하기 때문입니다. 어떤 일을 순차적으로 처리하는 것은 조금 번거롭기는 해도 세심한 주의를 기울이면 금방 숙달되는 경우가 종종 있습니다. 하지만, 그보다 더 인간을 괴롭히는 일은 단순한 작업과 선택을 끝없이 반복하는 것입니다. 조건문과 반복문은 그러한 문제를 소프트웨어를 통해 자동화하기 위해서 등장했습니다.

Ⅰ. 조건문

조건문에는 어떤 조건을 어떻게 설정할 것인가에 따라서 다양한 형태가 있을 수 있습니다. 그러나 분기점이 될 만한 특정한 조건을 충족했을 때(또는 충족하지 못했을 때) 실행할 명령을 지정해주는 기능을 한다는 점에서는 동일합니다. 어떤 형태를 띠는지 살펴보고 직접 사용해 보겠습니다.

1) if문

: if 문은 기본적으로 하나 이상의 조건문을 부여받고, 그 조건문이 참이면 그 아래에 있는 문장들을 수행하는 형태로 이루어집니다. 이때 주의할 점은 반드시 : (콜론)을 넣어주어야 하며, 정확한 들여쓰기를 사용해야 합니다.

- if문의 기본적 형태

```
if 조건 :
        실행할 명령1
        실행할 명령2
        ...
```

```
ex)
age = int(input("나이를 입력해주세요: "))
if age < 19 :
    print("이 사람은 미성년자입니다.")
if age >= 19 :
    print("이 사람은 성인입니다.")
```

※ 입력함수 input

: input 함수는 개발자인 여러분이 아니라 프로그램 사용자로부터 데이터를 입력받는 기능을 가진 함수입니다. 기본적으로 문자열 자료형으로 입력을 받기 때문에, 숫자를 저장하고 싶다면 int 형으로 변환해 주어야 합니다.

```
입력변수 = input(“안내문구”)
```

- 들여쓰기를 잘하자!

조건문과 반복문에서 들여쓰기에 주의해야하는 이유는?

(1) 프로그램 실행이 불가능함

각각의 프로그래밍 언어들에서 조건문과 그에 따른 실행문을 구별해주는 방법은 각기 다릅니다. 파이썬에서는 들여쓰기를 그 방법으로 사용합니다. 따라서 들여쓰기를 정확하게 줄에 맞춰서 해주지 않으면, 잘못된 실행결과가 나오거나 아예 오류로 프로그램이 실행되지 않을 수도 있습니다.

```
ex)
temperature = 39

if temperature >= 35 :
    print(“폭염 경보를 발령합니다.”)
    print(“에어컨 사고 싶다..”)


ex2)
temperature = 39

if temperature >= 35 :
    print(“폭염경보를 발령합니다.”)
print(“에어컨 사고 싶다..”)
```

(2) 코드 해석이 어려워짐

들여쓰기를 제대로 하지 않으면 프로그램이 어떻게 작성되어있는지 알아보기가 매우 힘들어집니다. 이 현상은 프로그램 길이가 길어지고 분기가 많아질수록 심해집니다. 심지어는 내가 작성한 프로그램조차 전혀 해석할 수 없는 경우도 생기게 됩니다.

Q. 그렇다면 들여쓰기는 어떻게 할까?

(1) 표준 들여쓰기 방법

: PEP-8에서 권장하고 있는 들여쓰기 방법은 하나의 코드 블럭 당 스페이스 바를 네 번(ㅅㅅㅅㅅ) 입력하는 것입니다. 이 방법은 특수문자에 해당하는 TAB키가 아닌 가장 기본적인 문자만 가지고도 입력할 수 있고 오류 가능성이 작아집니다.

(2) 비표준 들여쓰기 방법

: 스페이스바 네 번 입력하기 다음으로 가장 많이 사용되는 방법은 TAB키를 한 번 누르는 것입니다. 이 방법은 표준 들여쓰기보다 입력횟수가 적다는 장점이 있습니다. 그러나 기본 문자 외에 TAB이 사용되기 때문에 환경에 따라 오류가 발생할 수 있고, 스페이스 바와 혼용하면 들여쓰기 수정을 할 때 혼란을 일으킬 수 있습니다.

(3) 들여쓰기 방법은 통일하자!

: 들여쓰기를 할 때에는 어떤 방법을 택하느냐 보다는 하나의 프로그램 내에서 들여쓰기 방법을 통일하여 사용하는 것이 훨씬 더 중요합니다! 띄어쓰기를 사용하기 시작했다면 끝까지 띄어쓰기를, TAB키를 사용하기 시작했다면 끝까지 TAB키를 사용하면 됩니다.

2) else문

: else문은 바로 앞에 나오는 if문의 조건이 충족되지 않았을 경우 실행될 코드를 지정할 수 있도록 해줍니다. else문 내부에 다시 if문을 사용하면 다중 조건문을 사용할 수 있습니다.

- else문의 기본적 형태

```
if 조건 :
        실행할 명령1
        실행할 명령2
        ...
else:
        실행할 명령a
        실행할 명령b
        ...
```

```
ex)
age = int(input("나이를 입력해주세요: "))
height = int(input("키를 입력해주세요: "))

if(age >=14) and (height >= 120) :
        print("이 놀이기구에 탑승하실 수 있습니다!")
else:
        print("다음에 다시 오세요!")


ex2)
user_id = 'ID'
user_password = 'PASSWORD'
id_input = input('ID를 입력해주세요: ')
password_input = input('PASSWORD를 입력해주세요: ')

isAuth = (user_id == id_input) and (user_password == password_input)
if(isAuth):
        print("로그인 되었습니다.")
else:
        print("ID와 PASSWORD를 다시 확인해주세요.")
```

3) elif문

: if 와 else 만으로도 조건문을 사용하기에는 큰 문제가 없습니다. 하지만 필요한 논리적 판단이 많아질수록, 다시 수많은 반복 작업을 중복하여 나열하는 형태를 띠게 됩니다. 이때 elif를 사용하면 조건문 관리를 좀 더 쉽고 직관적으로 할 수 있습니다.

ex) - elif를 사용하지 않았을 때

```
score = int(input('성적을 입력해주세요: '))

if score >= 90:
    print('A')
else:
    if score >= 80:
        print('B')
    else:
        if score >= 70:
            print('C')
        else:
            print('F')
```

ex2) - elif를 사용했을 때

```
score = int(input('성적을 입력해주세요: '))

if score >= 90:
    print('A')
elif score >= 80:
    print('B')
elif score >= 70:
    print('C')
else:
    print('F')
```

4) 예외처리와 try, except문

: 조건문과 반복문을 활용한 프로그램을 개발하다 보면 생각지도 못한 부분에서 다양하고도 수많은 오류를 마주치게 됩니다. 물론 연습용으로 작성한 프로그램에서 발생하는 오류나 버그는 크게 문제가 되지 않을 것입니다. 그러나 실제 사용을 목적으로 하는 프로그램이라면 발생할 수 있는 오류를 최대한 미리 방지하고, 어쩔 수 없이 오류가 발생하더라도 검출과 대응이 가능하도록 만들어 둘 필요가 있습니다. 그러한 작업을 예외처리라고 하며 파이썬에서는 try, except, finally 등의 문법을 활용해서 구현할 수 있습니다.

```
try:
        실행할 문장1
        실행할 문장2
        ...
except () :
        오류가 났을 때 실행할 문장1
        오류가 났을 때 실행할 문장2
        ...
```

ex) score에 문자를 입력하면 오류가 발생한다.

```
try:
    score = int(input('성적을 입력해주세요: '))

    if score >= 90:
        print('A')
    elif score >= 80:
        print('B')
    elif score >= 70:
        print('C')
    else:
        print('F')

except ValueError:
    print("문자가 아닌 숫자를 입력해주세요")
```

Ⅱ. 반복문

1) for문

: for문은 반복문의 가장 기본적인 형태입니다. for문을 사용할 때 가장 신경써야할 부분은 내가 원하는 반복실행회수를 논리적으로 정확히 정해주는 것입니다. 그렇지 않으면 실행회수가 모자라거나 혹은 무한루프에 빠지게 되어 의도대로 작동하지 않습니다. 앞에서 배운 if문과 결합하여 사용하면 조건부 반복문을 만들 수도 있습니다. for문은 일반적으로 다음과 같은 형태로 되어있습니다.

```
for 변수 in 이터레이터:
    실행문1
    실행문2
    ....

ex)
menu = ['coke', 'fanta', 'cider']
for beverage in menu :
    print(beverage)

ex2)
for letter in 'The quick brown fox jumps over the lazy dog':
        print(letter)
```

- range 함수

: range(시작점, 끝점+1) 함수는 이름 그대로 시작점부터 끝점까지의 범위를 한정해주는 함수입니다. 반환 하는 데이터가 따로 없는 함수라는 점을 알고 사용하면 됩니다.

```
ex)
for num in range(6):
        print(num)

ex2)
for num in range(0, 100, 5):
        print(num)
```

- 중첩 for문

: for문 안에 for문이 들어갈 수도 있습니다. 이론적으로는 중첩회수에 제한이 없지만 for문이 한 번 중첩될 때마다 처리해야할 작업량은 기하급수적으로 늘어나게 되는 특성이 있습니다. 따라서 원하는 기능을 하는 프로그램을 만들되 가급적 for문의 중첩을 줄이는 방향으로 설계해야 실행시간이 줄고 성능이 향상됩니다.

```
ex)
for i in range(5):
    for j in range(5):
        for k in range(5):
            for l in range(5):
                for m in range(5):
                    print(i, j, k, l, m)

ex2)
for x in range(5):
        for y in range(x+1):
                print('*', end="")
        print('', x)
```

- 리스트 내포

: 리스트를 만들 때 리스트 안에 if문과 for문을 처음부터 포함시켜서 규칙에 맞는 요소를 가진 리스트를 자동으로 생성할 수도 있습니다. 리스트 내포라고 부르는 이 방법은 처음에는 어렵지만 일단 익숙해지면 직관적이고 편리하게 코드를 짜는데에 도움이 됩니다. 파이썬 코드들에서 꽤 자주 등장하니 알아두시면 좋습니다.

```
리스트 변수명 = [표현식 for 변수 in 이터레이터 if 조건문]
```

```
ex)
years = [year for year in range(2000, 2022)]
print(years)

ex2)
even_years = [year for year in range(2000, 2022) if year % 2 == 0]
print(even_years)
```

2) while문

: while문은 if 문이나 for 문과 다르게 조건문이 참인 동안에 계속해서 반복 실행되는 특징을 가지고 있습니다.

```
while 조건문 :
    실행문1
    실행문2
    ...
```

```
ex1)
num = 0

while num < 10 :
    print(num)
    num+=1
```

```
ex2) - 문자열포매팅을 사용한 반복출력
ahae = 1
print(f'제{ahae}의 아해가 무섭다고 그리오.')
while(ahae < 13):
    ahae += 1
    print(f‘제{ahae}의 아해도 무섭다고 그리오.’)
```

- 무한루프

반복문에서 무한루프가 성립하면 프로그램은 강제종료하기 전까지 같은 내용이 끝없이 반복해서 실행됩니다. 무한루프가 무조건 나쁜 것은 아니고 오히려 꼭 필요할 때도 있지만 의도되지 않은 무한루프는 고쳐야 할 버그에 해당합니다. 따라서 반복문이 무한루프에 빠지는 것을 막으려면 실행조건을 꼼꼼하게 설정해 루프에서 탈출할 수 있도록 해야 합니다.

```
ex)
num = 0
while True:
        print(num)
        num += 1
```

3) break문

: 반복문 실행 중에 break문을 만나면 그 즉시 반복문 실행이 중단되고 코드 블럭 밖으로 빠져나가 명령 실행을 이어나가게 됩니다. 그렇기 때문에 반복문 실행을 멈추고 싶은 부분에 적합한 조건을 넣어 break를 사용하면 프로그램이 무한루프에 빠지지 않도록 제어할 수 있습니다.

```
ex1)
quantity = 100
buyer = 1

while True:
    print(f'{buyer}번 고객님 주문해주셔서 감사합니다.')
    quantity -= 1
    buyer += 1
    if quantity <= 0:
        print('더 이상 구매할 수 없는 상품입니다.')
```

```
ex2)
quantity = 100
buyer = 1

while True:
    print(f'{buyer}번 고객님 주문해주셔서 감사합니다.')
    quantity -= 1
    buyer += 1
    if quantity <= 0:
        print('더 이상 구매할 수 없는 상품입니다.')
        break
```

4) continue문

: continue는, 실행 순서를 코드 블록의 바깥으로 탈출시키는 break와는 다르게 반복문 처음의 조건 검사 단계로 돌아가도록 만듭니다. 반복문 실행 중 continue를 만나게 되면 지금까지 진행되던 반복 실행 절차가 멈추고 현재 상황에서 다음 회차 반복문 실행조건을 다시 검사합니다. 이 때 조건이 충족되면 다시 반복 실행이 시작되고, 충족되지 않으면 실행 순서가 다음 줄로 넘어가게 됩니다.

```
ex1)
counting = ['한', '두', '세', '네', '다섯', '여섯', '일곱', '여덟', '아홉', '열']
for number in counting:
    print(f'{number} 꼬마')
    idx = counting.index(number)
    if (idx == 2) or (idx == 5) or (idx == 8) or (idx == 9):
        print('인디언\n')

ex2)
counting = ['한', '두', '세', '네', '다섯', '여섯', '일곱', '여덟', '아홉', '열']
for number in counting:
    print(f'{number} 꼬마')
    idx = counting.index(number)
    if (idx == 2) or (idx == 5) or (idx == 8) or (idx == 9):
        continue
        print('인디언\n')
```

▲실습

[문제1] 코드 리터러시

다음 코드를 실행했을 때 출력되는 결과는 무엇일까요?

```
>>> a = "Life is too short, you need python"
>>> if 'wife' in a:
...     print('wife')
... elif 'python' in a and 'you' not in a:
...     print('python')
... elif 'shirt' not in a:
...     print('shirt')
... elif 'need' in a:
...     print('need')
... else:
...     print('none')
```

[문제2] 조건문 사용하기

친구의 결혼식에 간 고길동 씨는 하객들에게 행운권을 나누어 주고 추첨을 통해 경품을 주는 이벤트를 보고 바로 응모했습니다. 그가 받은 행운권 번호는 23번이었습니다. 다음이 행운권의 당첨번호 리스트라고 할 때, 고길동 씨가 당첨되었다면 “야호!”라는 문자열을 출력하는 프로그램을 작성해보세요.

```
>>> lucky_list = [1, 9, 23, 46, 72]
```

[문제3] 1부터 100까지 출력

1부터 100까지의 숫자를 for문을 이용하여 출력해보세요.

[문제4] 5의 배수의 총합

for문을 이용하여 1부터 1000까지의 자연수 중 5의 배수에 해당하는 자연수들의 총합을 구해 출력해보세요.

[문제5] 학급의 평균 점수

A반 학생들의 국어 성적이 다음과 같다고 할 때, for문을 이용하여 A 학급의 평균 점수를 계산해 보세요.

```
A = [70, 60, 55, 75, 95, 90, 80, 80, 85, 100]
```

[문제6] 혈액형

다음은 오늘 헌혈한 사람들의 혈액형(A, B, AB, O)을 기록한 데이터입니다.

```
blood_type = ['A', 'B', 'A', 'O', 'AB', 'AB', 'O', 'A', 'B', 'O', 'B', 'AB']
```
for 문을 이용하여 각 혈액형 별 사람 수의 합계를 구해보세요.

[문제7] 리스트 내포

리스트 중에서 홀수에만 2를 곱하여 저장하는 다음과 같은 코드가 있습니다.

```
numbers = [1, 2, 3, 4, 5]

result = []
for n in numbers:
    if n % 2 == 1:
        result.append(n*2)
```

위 코드를 리스트 내포를 이용하여 표현해보세요.

[문제8] 1부터 100까지 더하기

1부터 100까지의 자연수를 모두 더하고 그 결과를 출력하세요.

[문제9] 3의 배수의 합

1부터 1000까지의 자연수 중 3의 배수의 합을 구하시오.

[문제10] 50점 이상의 평균

다음은 A반 학생들의 사회 성적을 나타낸 리스트입니다. 50점 미만을 받은 사람은 낙제에 해당하여 평균 점수 계산에서 제외되어야 한다고 할 때, A반의 사회 성적 평균 점수를 구해 보세요.

```
A = [20, 55, 67, 82, 45, 33, 90, 87, 100, 25]
```

[문제11] 별 표시하기1

while문을 이용하여 아래와 같이 별(*)을 표시하는 프로그램을 작성해 보세요.

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

[문제12] 별 표시하기2

while문을 이용하여 아래와 같이 별(*)을 표시하는 프로그램을 작성해 보세요.

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

[문제13] 구구단

for문과 range함수를 이용해서 2단부터 13단까지 구구단을 출력해보세요.

[문제14] 따라 쳐보기 - 포츈쿠키 자판기

```
import random

fortune = 10

tell = ["오늘은 로또를 사보세요~", "당신에게 사랑이 찾아옵니다!","얼른 집으로 돌아 가세요", "슬플 땐 슬퍼해도 됩니다", "오늘은 반가운 사람을 만나겠군요!", "답답해도 조금만 참아 봐요", "과자를 너무 믿지 마세요", "가족들과 시간을 보내보세요", "이번 주에는 지출이 많겠네요"]

while True:
    money = int(input("\n동전을 넣어 주세요: "))
    if money == 500:
        num = random.randint(0, 8)
        print('='*15, "~운명의 포츈쿠키~", "="*15)
        print('\n', tell[num], '\n')
        print('='*49)
        fortune = fortune - 1
        print("남은 쿠키는 %d개 입니다." % fortune)
    elif money > 500:
        print("거스름돈 %d원과 포츈쿠키를 드립니다." % (money - 500))
        num = random.randint(0, 8)
        print('=' * 15, "~운명의 포츈쿠키~", "=" * 15)
        print('\n', tell[num], '\n')
        print('=' * 49)
        fortune = fortune - 1
        print("남은 쿠키는 %d개 입니다." % fortune)
    else:
        print("돈을 다시 돌려드릴게요.")
        print("남은 쿠키는 %d개 입니다." % fortune)
    if not fortune:
        print("포츈쿠키가 다 떨어졌습니다. 판매를 중지 합니다.")
        break
```