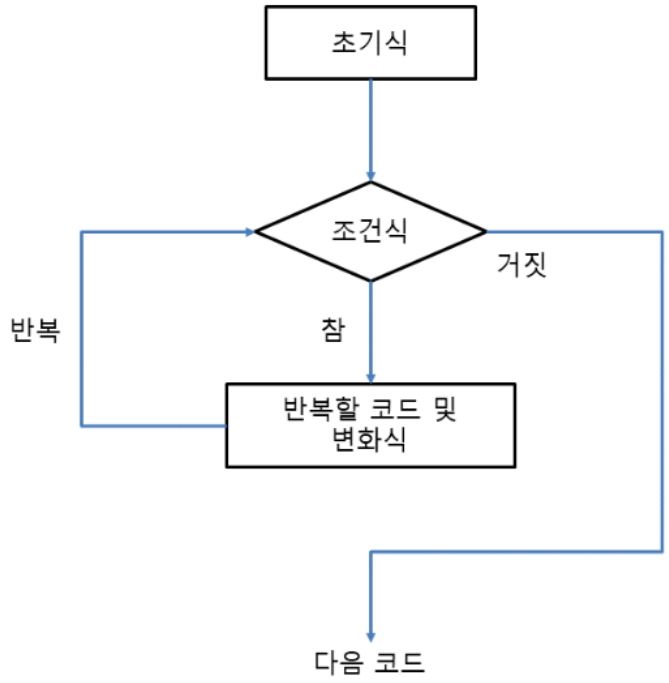
Ch7. 반복문

- **while 반복문으로 Hello, world! 100번 출력하기**
  - while 반복문은 조건식으로만 동작하며 반복할 코드 안에 조건식에 영향을 주는 변화식이 들어감

```
i = 0                       # 초기식
while i < 100:              # while 조건식
    print('Hello, world!')    # 반복할 코드
    i += 1                    # 변화식
```

  - while 반복문의 실행 과정임
  - 초기식부터 시작하여 조건식을 판별함
  - 조건식이 참(True)이면 반복할 코드와 변화식을 함께 수행함
  - 조건식을 판별하여 참(True)이면 코드를 계속 반복하고, 거짓(False)이면 반복문을 끝낸 뒤 다음 코드를 실행함

- **while 반복문으로 Hello, world! 100번 출력하기**
  - 조건식 → 반복할 코드 및 변화식 → 조건식으로 순환하는 부분이 루프(loop)임



- **while 반복문 사용하기**

```
초기식
while 조건식:
    반복할 코드
    변화식
```

  - while 다음 줄에 오는 코드는 반드시 들여쓰기를 해줌

```
>>> i = 0
>>> while i < 100:
...     print('Hello, world!')
...     i += 1
...
Hello, world!
... (생략)
Hello, world!
Hello, world!
Hello, world!
```

- **while 반복문 사용하기**

- **while 반복문 사용하기**
  - i에 0이 아닌 1을 할당하여 'Hello, world!'를 100번 출력해보자

```
>>> i = 1
>>> while i <= 100:
...     print('Hello, world!', i)
...     i += 1
...
Hello, world! 1
Hello, world! 2
Hello, world! 3
...  (생략)
Hello, world! 99
Hello, world! 100
```

- **while 반복문 사용하기**
  - i 초깃값을 감소시키기

```
>>> i = 100
>>> while i > 0:
...     print('Hello, world!', i)
...     i -= 1
...
Hello, world! 100
Hello, world! 99
Hello, world! 98
... (생략)
Hello, world! 2
Hello, world! 1
```

- **입력한 횟수대로 반복하기**
  - 입력한 횟수대로 반복을 해보자
  - IDLE의 소스 코드 편집 창에 입력하자

while_input_increment.py

```
count = int(input('반복할 횟수를 입력하세요: '))

i = 0
while i < count:     # i가 count보다 작을 때 반복
    print('Hello, world!', i)
    i += 1
```

실행 결과

```
반복할 횟수를 입력하세요: 3 (입력)
Hello, world! 0
Hello, world! 1
Hello, world! 2
```

- **입력한 횟수대로 반복하기**

while_input_decrement.py

```
count = int(input('반복할 횟수를 입력하세요: '))

while count > 0:     # count가 0보다 클 때 반복
    print('Hello, world!', count)
    count -= 1       # count를 1씩 감소시킴
```

실행 결과

```
반복할 횟수를 입력하세요: 3 (입력)
Hello, world! 3
Hello, world! 2
Hello, world! 1
```

- **반복 횟수가 정해지지 않은 경우**
  - 난수를 생성해서 숫자에 따라 반복을 끝내 보자
  - 난수(random number)란 특정 주기로 반복되지 않으며 규칙 없이 무작위로 나열되는 숫자를 뜻함
  - 현실에서 쉽게 접할 수 있는 난수가 바로 주사위를 굴려서 나온 숫자임

- **반복 횟수가 정해지지 않은 경우**
  - 파이썬에서 난수를 생성하려면 random 모듈이 필요함
  - import 키워드를 사용하여 가져올 수 있음

• import 모듈

```
import random    # random 모듈을 가져옴
```

  - random.random()으로 random 모듈의 random 함수를 호출해보자

```
>>> random.random()
0.002383731799935007
>>> random.random()
0.3297914484498006
>>> random.random()
0.6923390064955324
```

  - random.random()을 실행할 때마다 계속 다른 실수가 출력되는 것은 바로 이 숫자가 바로 난수임

- **반복 횟수가 정해지지 않은 경우**
  - randint 함수는 난수를 생성할 범위를 지정하며, 범위에 지정한 숫자도 난수에 포함

    - `random.randint(a, b)`

  - randint 함수로 주사위를 만들어보자
  - 정육면체 주사위는 1부터 6까지 숫자는 random.randint(1, 6)처럼 1과 6을 넣으면 1과 6 사이의 난수가 생성됨

```
>>> random.randint(1, 6)
4
>>> random.randint(1, 6)
1
>>> random.randint(1, 6)
5
```

- **반복 횟수가 정해지지 않은 경우**
  - 1과 6 사이의 난수를 생성한 뒤 3이 나오면 반복을 끝냄

while_random.py

```
import random    # random 모듈을 가져옴

i = 0
while i != 3:    # 3이 아닐 때 계속 반복
    i = random.randint(1, 6)    # randint를 사용하여 1과 6 사이의 난수를 생성
    print(i)
```

실행 결과

```
5
1
4
1
1
3
```

- **while 반복문으로 무한 루프 만들기**

while_infinite_loop.py

```
while True:    # while에 True를 지정하면 무한 루프
    print('Hello, world!')
```

실행 결과

```
... (생략)
Hello, world!
Hello, world!
Hello, world!
Hello, world!
... (계속 반복)
```

  - while에 조건식 대신 True를 지정하면 무한히 반복하는 무한 루프가 만들어짐
  - 조건식이 항상 참(True)이므로 변화식도 필요 없음
  - 스크립트 파일을 실행한 상태로 두면 'Hello, world!'는 끝나지 않고 계속 출력됨
  - IDLE이나 콘솔(터미널, 명령 프롬프트)에서 Ctrl+C를 입력하여 무한 루프를 끝냄
  - while에 True 대신 True로 취급하는 값을 사용해도 무한 루프로 동작

- **while 반복문으로 무한 루프 만들기**

```
while 1:    # 0이 아닌 숫자는 True로 취급하여 무한 루프로 동작
    print('Hello, world!')
```

```
while 'Hello':    # 내용이 있는 문자열은 True로 취급하여 무한 루프로 동작
    print('Hello, world!')
```

- **for 반복문으로 Hello, world! 100번 출력하기**
  - 'Hello, world!' 문자열을 100번 출력하려면?
  - 가장 간단한 방법은 print를 100번 사용해서 출력하는 것임

```
# print 100번 사용
print('Hello, world!')
print('Hello, world!')
print('Hello, world!')
print('Hello, world!')
# ... (생략)
print('Hello, world!')
print('Hello, world!')
print('Hello, world!')
print('Hello, world!')
```

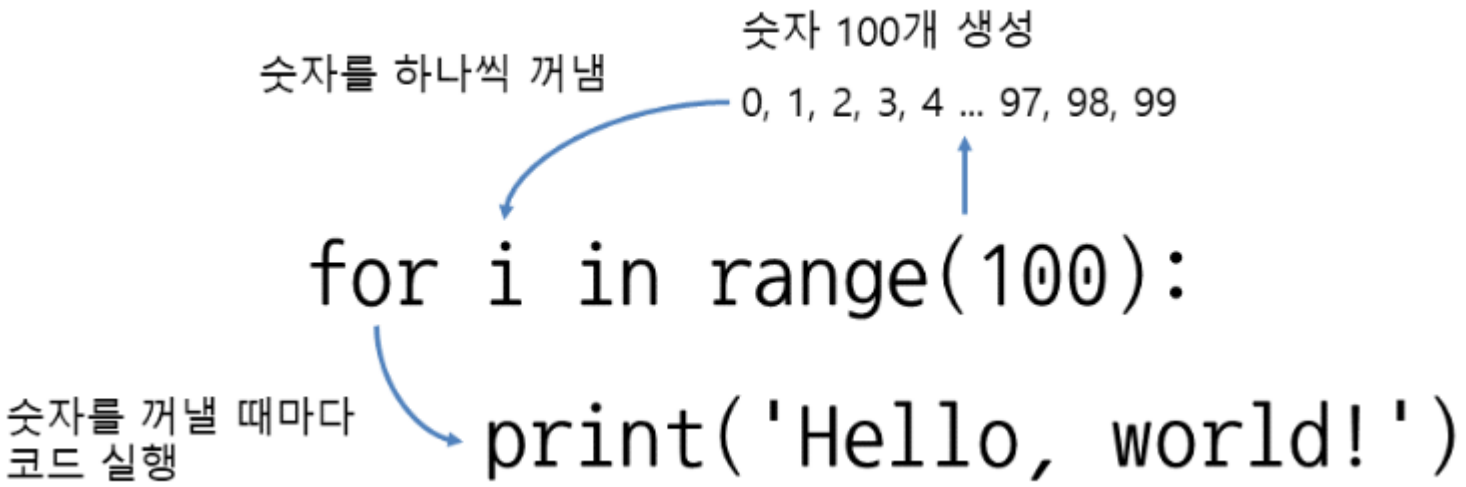
- **for와 range 사용하기**
  - 복사, 붙여 넣기로 print('Hello, world!')를 100번 붙여 넣으면 어렵지 않게 완성
  - 1,000번 또는 10,000번을 출력한다면?
  - 코드를 붙여 넣는데 시간이 너무 오래 걸림, 프로그래밍 측면에서도 비효율적임
  - 대부분의 프로그래밍 언어에서는 반복되는 작업을 간단하게 처리하기 위해 반복문이라는 기능을 제공
  - 반복문은 반복 횟수, 반복 및 정지 조건을 자유자재로 제어

- **for와 range 사용하기**
  - for 반복문은 range에 반복할 횟수를 지정하고 앞에 in과 변수를 입력
  - 끝에 :(콜론)을 붙인 뒤 다음 줄에 반복할 코드를 넣음

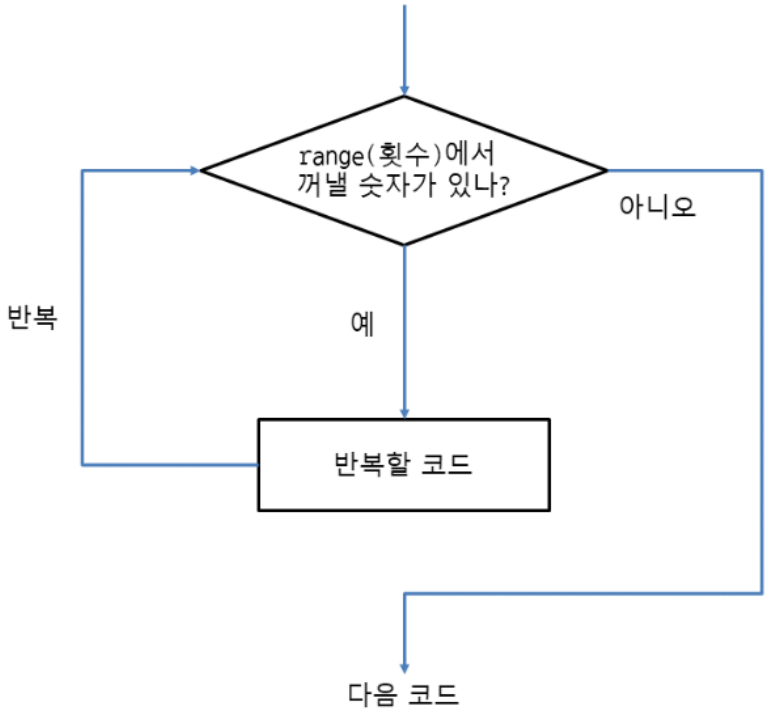
```
for 변수 in range(횟수):
    반복할 코드
```

```
>>> for i in range(100):
...     print('Hello, world!')
...
Hello, world!
... (생략)
Hello, world!
Hello, world!
Hello, world!
```

- **for와 range 사용하기**
  - 파이썬의 for 반복문은 range에서 in으로 숫자를 하나하나 꺼내서 반복하는 방식임
  - for는 숫자를 꺼낼 때마다 코드를 실행함



- **for와 range 사용하기**
  - for 반복문의 동작 과정 그림으로 표현하면 다음과 같음
  - for 변수 in range(횟수) → 반복할 코드로 순환하는 것을 루프(loop)라고 부름



- **반복문에서 변수의 변화 알아보기**
  - range에서 꺼낸 숫자를 눈으로 확인해보자

```
>>> for i in range(100):
...     print('Hello, world!', i)
...
Hello, world! 0
Hello, world! 1
Hello, world! 2
... (생략)
Hello, world! 98
Hello, world! 99
```

  - range에서 꺼낸 숫자는 변수 i에 저장되며 반복할 코드에서 사용할 수 있음

- **for와 range 응용하기**
  - 시작하는 숫자와 끝나는 숫자 지정하기
  - range에 횟수만 지정하면 숫자가 0부터 시작하지만, 다음과 같이 시작하는 숫자와 끝나는 숫자를 지정해서 반복할 수도 있음

• for 변수 in range(시작, 끝):

```
>>> for i in range(5, 12):    # 5부터 11까지 반복
...     print('Hello, world!', i)
...
Hello, world! 5
Hello, world! 6
Hello, world! 7
Hello, world! 8
Hello, world! 9
Hello, world! 10
Hello, world! 11
```

- **증가폭 사용하기**
  - range는 증가폭을 지정해서 해당 값만큼 숫자를 증가시킬 수 있음
  - 0부터 9까지의 숫자 중에서 짝수만 출력

• for 변수 in range(시작, 끝, 증가폭):

```
>>> for i in range(0, 10, 2):    # 0부터 8까지 2씩 증가
...     print('Hello, world!', i)
...
Hello, world! 0
Hello, world! 2
Hello, world! 4
Hello, world! 6
Hello, world! 8
```

- **숫자를 감소시키기**
  - for와 range는 숫자가 증가하면서 반복함
  - 그럼 숫자를 감소시킬 수는 없을까?

```
>>> for i in range(10, 0):    # range(10, 0)은 동작하지 않음
...     print('Hello, world!', i)
...
```

  - 실행을 해보면 아무것도 출력되지 않음
  - range는 숫자가 증가하는 기본 값이 양수 1이기 때문임

```
>>> for i in range(10, 0, -1):    # 10에서 1까지 1씩 감소
...     print('Hello, world!', i)
...
Hello, world! 10
Hello, world! 9
Hello, world! 8
... (생략)
Hello, world! 2
Hello, world! 1
```

- **숫자를 감소시키기**
  - **증가폭을 음수로 지정하는 방법 말고도 reversed를 사용하면 숫자의 순서를 반대로 뒤집을 수 있음**

- for 변수 in reversed(range(횟수))
- for 변수 in reversed(range(시작, 끝))
- for 변수 in reversed(range(시작, 끝, 증가폭))

```
>>> for i in reversed(range(10)):    # range에 reversed를 사용하여 숫자의 순서를 반대로 뒤집음
...     print('Hello, world!', i)    # 9부터 0까지 10번 반복
...
Hello, world! 9
Hello, world! 8
Hello, world! 7
... (생략)
Hello, world! 1
Hello, world! 0
```

- **입력한 횟수대로 반복하기**

for_range_input.py

```
count = int(input('반복할 횟수를 입력하세요: '))

for i in range(count):
    print('Hello, world!', i)
```

실행 결과

```
반복할 횟수를 입력하세요: 3 (입력)
Hello, world! 0
Hello, world! 1
Hello, world! 2
```

- **시퀀스 객체로 반복하기**
  - for에 range 대신 시퀀스 객체를 넣어도 됨
  - for는 리스트, 튜플, 문자열 등 시퀀스 객체로 반복할 수 있음
  - for에 range 대신 리스트를 넣으면 리스트의 요소를 꺼내면서 반복함

```
>>> a = [10, 20, 30, 40, 50]
>>> for i in a:
...     print(i)
...
10
20
30
40
50
```

- **시퀀스 객체로 반복하기**
  - 튜플도 마찬가지로 튜플의 요소를 꺼내면서 반복함

```
>>> fruits = ('apple', 'orange', 'grape')
>>> for fruit in fruits:
...     print(fruit)
...
apple
orange
grape
```

  - 참고로 여기서는 for 반복문의 변수를 i 대신 fruit로 사용함
  - for에서 변수 i는 다른 이름으로 만들어도 상관없음
  - 문자열도 시퀀스 객체임
  - for에 문자열을 지정하면 문자를 하나씩 꺼내면서 반복함

```
>>> for letter in 'Python':
...     print(letter, end=' ')
...
P y t h o n
```

- **시퀀스 객체로 반복하기**
  - 문자열 'Python'을 뒤집어서 문자를 출력할 수는 없을까?
  - 이때는 앞에서 배운 reversed를 활용하면 됨

  - reversed(시퀀스객체)

```
>>> for letter in reversed('Python'):
...     print(letter, end=' ')
...
n o h t y P
```

  - 문자열 'Python'에서 문자 n부터 P까지 출력됨
  - reversed는 시퀀스 객체를 넣으면 시퀀스 객체를 뒤집어 줌
    (원본 객체 자체는 바뀌지 않으며 뒤집어서 꺼내줌)
  - **for 반복문은 반복 개수가 정해져 있을 때 주로 사용함**
  - **for 반복문은 range 이외에도 시퀀스 객체를 사용할 수 있다는 점이 중요**

- **for 반복문으로 요소 출력하기**

```
for 변수 in 리스트:
    반복할 코드
```

```
>>> a = [38, 21, 53, 62, 19]
>>> for i in a:
...     print(i)
...
38
21
53
62
19
```

  - print로 i를 출력하면 모든 요소를 순서대로 출력할 수 있음
  - 물론 in 다음에 리스트를 직접 지정해도 상관 없음

```
for i in [38, 21, 53, 62, 19]:
    print(i)
```

- ## 인덱스와 요소 함께 출력하기

  - for 인덱스, 요소 in enumerate(리스트):

```
>>> a = [38, 21, 53, 62, 19]
>>> for index, value in enumerate(a):
...     print(index, value)
...
0 38
1 21
2 53
3 62
4 19
```

```
>>> for index, value in enumerate(a):
...     print(index + 1, value)
...
1 38
2 21
3 53
4 62
5 19
```

- **인덱스와 요소 함께 출력하기**

  - for 인덱스, 요소 in enumerate(리스트, start=숫자):

```
>>> for index, value in enumerate(a, start=1):
...     print(index, value)
...
1 38
2 21
3 53
4 62
5 19
```

  - 반복문 사용 시 몇 번째 반복문인지 확인

- **while 반복문으로 요소 출력하기**

```
>>> a = [38, 21, 53, 62, 19]
>>> i = 0
>>> while i < len(a):
...     print(a[i])
...     i += 1
...
38
21
53
62
19
```

```
>>> a = [38, 21, 53, 62, 19]
>>> i = 0
>>> while i <= len(a):
...     print(a[i])
...     i += 1
...
38
21
53
62
19
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
IndexError: list index out of range
```

- **while 반복문으로 요소 출력하기**

```
while i < len(a):
    print(a[i])
    i += 1
```