

반복문

김지성 강사

반복문

- ✓ 프로그래밍을 하는 가장 큰 이유는 사람이 하기 어려운 반복을 기계가 대신 해주는 것.
- ✓ 만약 “Hello World”를 100번 출력하는 코드를 작성하고, Hello를 Hi로 변경해야 한다면 굉장히 비효율적으로 코드를 고쳐야 할 것이다.
- ✓ 이러한 반복적인 일을 대신 하기 위해서 파이썬에서는 반복을 하기 위한 문법이 존재한다.

반복문 - range()

- ✓ range() 함수는 연속되는 숫자 요소들을 만들 때 활용한다.
- ✓ range(시작, 끝, 증감크기)의 형식으로 사용한다.
 - list() 함수를 활용해 리스트로 변환 가능
 - range 객체는 iterate할 때 원하는 시퀀스 항목들을 순서대로 요소를 돌려주는 객체
 - 메모리에 공간을 미리 할당하는 것이 아니라 필요할 때 다음 요소를 반환해준다.

```
print(list(range(10)))  
print(list(range(0,10,2)))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
[0, 2, 4, 6, 8]
```

반복문 - for

- ✓ for문은 시퀀스 자료형의 항목들을 순서대로 하나씩 꺼낸다.
 - iterable 객체를 반복할 때 좋다.
 - 반복 횟수가 정해져 있을 때 주로 사용.

for **요소를 담을 변수** in **반복 가능한 객체**:

반복할 코드(띄어쓰기 4칸)

for **변수** in **range(횟수)**:

반복할 코드(띄어쓰기 4칸)

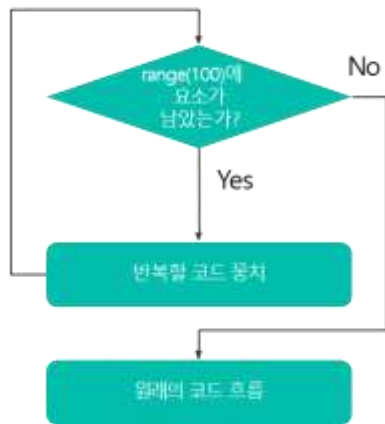
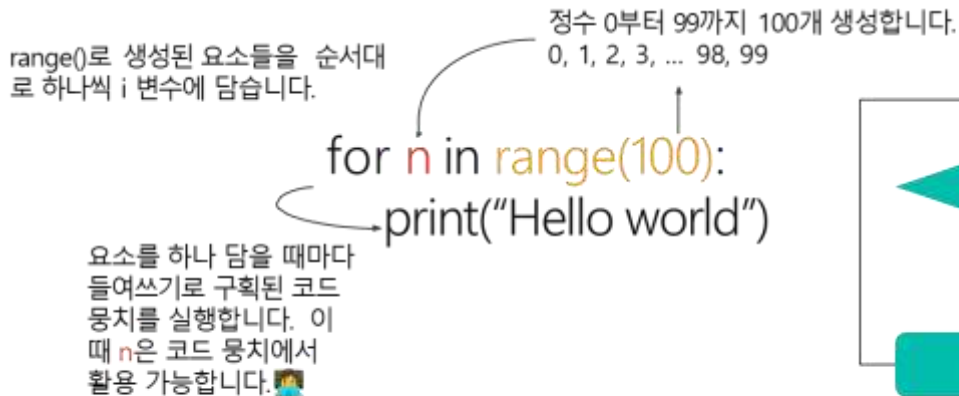
반복문 - for

- ✓ iterable: ‘반복 가능한’이란 사전적 의미를 갖는 이 단어는 파이썬에선 이런 뜻을 가진다.
 - 여러 개의 요소들로 구성되어 있으면서 요소를 한 번에 하나씩 돌려줄 수 있는 객체
 - 예를 들면 list, str, tuple, range와 같은 시퀀스 자료형이 있으며, 이와 상반되는 비 시퀀스 자료형으로는 dictionary가 있다.
- ✓ 객체: 우선은 메모리에 할당되어 존재하고 있는 데이터라고 생각하자. 클래스에서 더욱 자세히 배울 예정

반복문 - for

- ✓ for문을 통해 “Hello World”를 100번 출력하는 코드.
- ✓ range(100)을 통해 for문은 0~99까지 100번 반복을 하게된다. 이때마다 print()문을 실행.

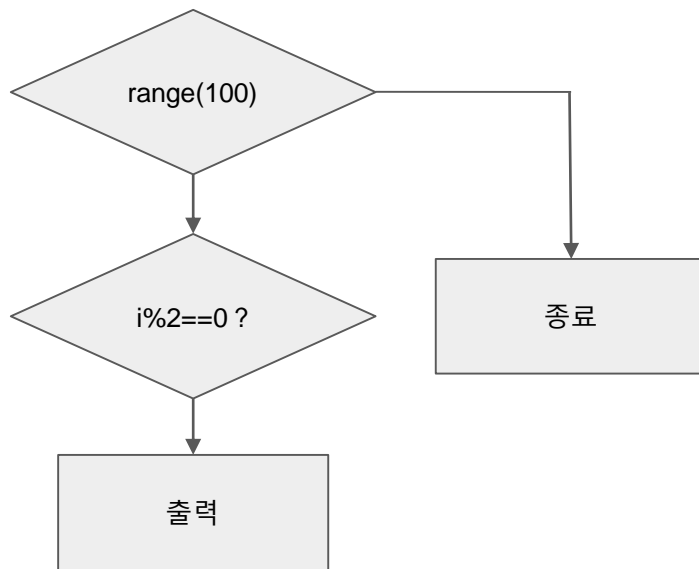
```
for i in range(100):  
    print("Hello World")
```



반복문 - for

- ✓ for문 안에 조건문 혹은 다시 반복문이 들어갈 수 있다.

```
for i in range(100):  
    if i%2==0: print(i,end=' ')
```



반복문 - for

- ✓ for문 안에 조건문 혹은 다시 반복문이 들어갈 수 있다.

```
for i in range(4):  
    for j in range(5):  
        print(f"{i} * {j} = {i*j}")
```

- ✓ 다중 반복문의 구조는 다음과 같다.
1. 가장 바깥에 있는 반복문에 들어간다.
 2. 다음 안쪽 반복문에 들어간다. (1-2를 반복하여 가장 안쪽 반복문까지 진행)
 3. 안쪽 반복문을 다 반복할 때까지 진행.
 4. 안쪽 반복문을 다 반복했다면 그 바깥 반복문을 반복.
 5. 3-4를 반복하여 가장 바깥 반복문의 반복이 끝날때까지 진행.

반복문 - for

✓ 반복문 연습문제

- 사용자로부터 자연수를 입력받고 입력받은 횟수만큼 “Hello world”를 출력해보세요.

실행 결과:

인사 몇 번 해드릴까요? 3

Hello world

Hello world

Hello world

반복문 - for

✓ 반복문 연습문제

- 아래의 실행 결과처럼 나오도록 list()와 range()를 활용해서 구현해보세요.

실행 결과:

[5, 6, 7, 8, 9]

실행 결과:

[0, 3, 6, 9]

실행 결과:

[-10, -40, -70]

반복문 - for

✓ 반복문 연습문제

- 아래의 실행 결과처럼 나오도록 for(), if(), range()를 활용해서 구현해보세요.
- '*'의 개수는 10개입니다.

실행 결과:

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

반복문 - for

✓ 반복문 연습문제

- 1부터 사용자로부터 입력 받은 자연수까지의 홀수의 합을 구합니다. 1부터 입력 받은 수까지 홀수 전체를 각각 더하는 산식을 표현하고 그 계산식의 결과를 아래 실행결과와 같이 출력되도록 구해보세요.

실행 결과:

1부터 입력한 자연수까지의 모든 홀수의 합을 구합니다 100

$1 + 3 + 5 + 7 + 9 + 11 + 13 + 15 + 17 + 19 + 21 + 23 + 25 + 27 + 29 + 31 + 33 + 35 + 37 + 39 + 41 + 43 + 45 + 47 + 49 + 51 + 53 + 55 + 57 + 59 + 61 + 63 + 65 + 67 + 69 + 71 + 73 + 75 + 77 + 79 + 81 + 83 + 85 + 87 + 89 + 91 + 93 + 95 + 97 + 99 = 2500$

반복문 - for

✓ 반복문 연습문제

○ 계승(Factorial)을 계산해보려합니다. 사용자로부터 정수를 하나 입력받습니다. 그리고 입력받은 숫자까지의 계승을 for문을 활용하여 구한 뒤 출력해주세요.

■ 계승이란 1부터 n까지의 자연수를 모두 곱하는 것을 의미합니다.

■ $n! = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$

실행 결과:

계승을 구할 숫자를 입력하세요. 4

24

실행 결과:

계승을 구할 숫자를 입력하세요. 7

5040

반복문 - for

✓ 반복문 연습문제

- 아래의 실행 결과처럼 나오도록 for문과 range()를 활용해서 구현해보세요.

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

반복문 - for

✓ 반복문 연습문제

- 아래의 실행 결과처럼 나오도록 for문과 range()를 활용해서 구현해보세요.
- - 숫자와 숫자 사이에 \t를 사용합니다.

실행 결과:

자연수 하나 입력해주세요: 6

```
6      6      6      6      6      6
5      5      5      5      5
4      4      4      4
3      3      3
2      2
1
```

반복문 - for

✓ 반복문 연습문제

- 아래의 실행 결과처럼 나오도록 for문과 range()를 활용해서 구현해보세요.
- 높이는 10.

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


반복문 - for

✓ 반복문 연습문제

- 아래의 실행 결과처럼 나오도록 for문과 range()를 활용해서 구현해보세요.
- 높이는 10.

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


반복문 - for

- ✓ for문에서 range() 대신 시퀀스 객체를 넣으면 요소를 하나씩 꺼내와서 반복을 진행한다.

```
str1 = "가나다라마바사"  
for s in str1:  
    print(s)
```

```
li1 = [1,2,3,4,5]  
for l in li1:  
    print(l)
```

```
tu1 = (1,2,3,4,5)  
for l in li1:  
    print(l)
```

반복문 - for

- ✓ for문에서 range() 대신 시퀀스 객체를 넣으면 요소를 하나씩 꺼내와서 반복을 진행한다.
 - 시퀀스 객체를 거꾸로 반복하려면 reversed() 함수를 통해서 역순으로 반복이 가능.

```
str1 = "가나다라마바사"  
for s in reversed(str1):  
    print(s)
```

반복문 - for

✓ 반복문 연습문제

- 제시된 리스트의 값을 for문을 활용하여 10배씩 증가시킨 후 아래의 실행 결과와 같이 출력해보세요.
- `x = [10, -42, 25, 103, 21, 77, 24]`

실행 결과:

100 -420 250 1030 210 770 240

반복문 - for

✓ 반복문 연습문제

- 사용자로부터 자연수를 입력받아옵니다. 그 숫자의 구구단을 출력하는 프로그램을 구현해 보세요. 아래의 실행 결과를 참고하세요.(for문 활용)

실행 결과:

구구단을 알려드립니다. 숫자를 입력하세요 9

9 * 1 = 9

9 * 2 = 18

9 * 3 = 27

9 * 4 = 36

9 * 5 = 45

9 * 6 = 54

9 * 7 = 63

9 * 8 = 72

9 * 9 = 81

반복문 - for

✓ 반복문 연습문제

- 사용자로부터 자연수를 입력받아옵니다. 1부터 입력된 자연수까지의 합을 구하는 프로그램을 구현해보세요.

실행 결과:

자연수를 입력해주세요: 100

5050

반복문 - while

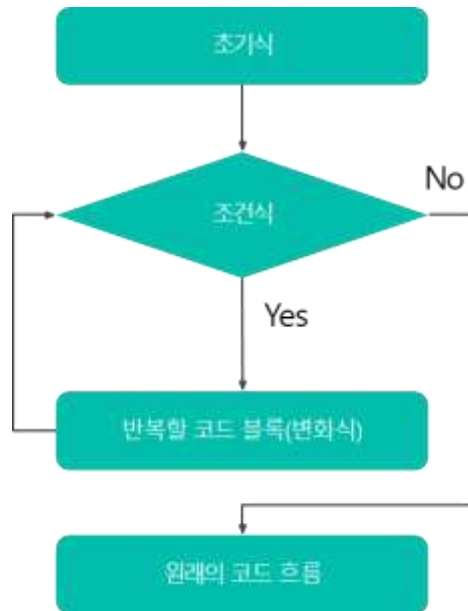
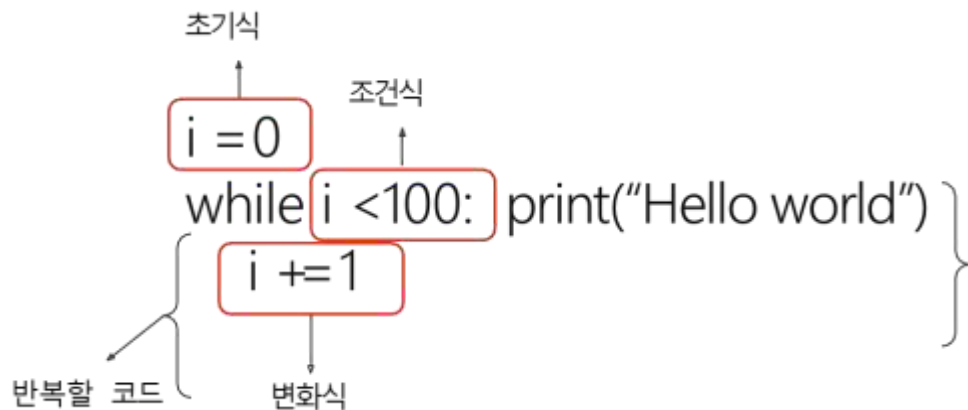
- ✓ while 반복문은 조건식으로만 동작하는 반복문이다.
 - 조건식을 판별한다.
 - 조건식이 True인 동안 while문 내에 있는 코드 블록을 실행
 - 조건식이 False가 되면 while문을 빠져나가 종료

```
i = 0
while i < 10:
    print(i, end=" ")
    i += 1
```

0 1 2 3 4 5 6 7 8 9

반복문 - while

- ✓ while 반복문은 조건식으로만 동작하기 때문에 반드시 조건식을 멈출 변화식이 필요하다.
- ✓ 아래 그림에서는 i값을 1씩 증가시켜 100이 될때까지 반복을 진행한다.



반복문 - while

- ✓ while 반복문과 for 반복문의 차이는 정확한 반복 횟수를 지정하느냐 지정하지 않느냐이다.
 - while 반복문은 조건으로 반복을 해야할 때 사용한다.
 - for 반복문은 정확한 반복 횟수에 따라서 반복을 해야할 때 사용한다.
- ✓ 기본적으로 while 반복문으로 만들 수 있는 코드는 for 반복문으로도 만들 수 있다. 그 역으로도 마찬가지로 가능하다. 하지만 정확한 용도에 따라서 사용하는 것이 가독성, 퍼포먼스에서 좋다.

반복문 - while

✓ 반복문 연습문제

○ while문을 사용하여 1이상 100 이하의 짝수를 '\t'를 구분자로 해서 출력해보세요.

실행 결과:

2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36
	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70
	72	74	76	78	80	82	84	86	88	90	92	94	96	98	100		

2	4	6	8	10	12	14	16
18	20	22	24	26	28	30	32
34	36	38	40	42	44	46	48
50	52	54	56	58	60	62	64
66	68	70	72	74	76	78	80
82	84	86	88	90	92	94	96
98	100						

반복문 - while

✓ 반복문 연습문제

- 사용자로부터 자연수를 입력받아옵니다. 그 숫자의 구구단을 출력하는 프로그램을 구현해 보세요. 아래의 실행 결과를 참고하세요.(while문 활용)

실행 결과:

구구단을 알려드립니다. 숫자를 입력하세요 9

9 * 1 = 9

9 * 2 = 18

9 * 3 = 27

9 * 4 = 36

9 * 5 = 45

9 * 6 = 54

9 * 7 = 63

9 * 8 = 72

9 * 9 = 81

반복문 - while

✓ 반복문 연습문제

- 사용자로부터 학생의 숫자와 학생 성적을 입력받습니다. 학생 성적 평균을 구하는 프로그램을 while문으로 구현해보세요.

실행 결과:

학생의 수 : 3

1번 학생의 점수 : 77

2번 학생의 점수 : 88

3번 학생의 점수 : 99

평균 : 88.0점

반복문 - while

✓ 반복문 연습문제

- while문을 사용하여 학생의 이름을 출력해보세요.
- `student = ["김", "이", "박", "최", "공"]`

실행 결과:
김 이 박 최 공

반복문 - while

✓ 반복문 연습문제

- A학급의 수학 점수가 다음과 같을 때 while문 조건식에 len() 함수를 사용하여 이 학급의 평균 점수를 구하는 프로그램을 구현해보세요.
- `class_a = (70, 60, 55, 75, 95, 90, 80, 80, 85, 100)`

실행 결과:
79.0

반복문 제어

김지성 강사

반복문 제어

- ✓ 파이썬에서의 반복문은 반복을 멈추거나, 특정 코드블록을 실행하지 않고 다시 반복문으로 돌아가게 하거나 하는 등 반복문을 제어할 수 있는 문법이 존재한다.
- ✓ `break`는 반복문을 멈추게 한다.
- ✓ `continue`는 다시 반복문으로 돌아가게 한다.

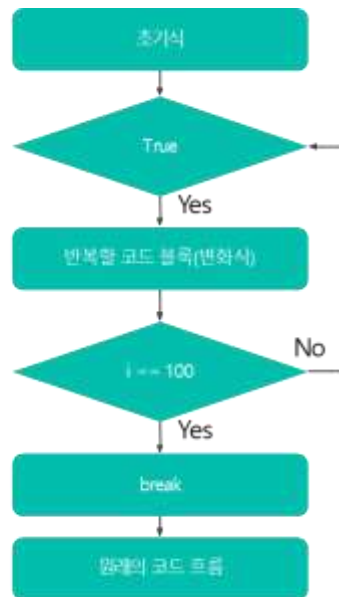
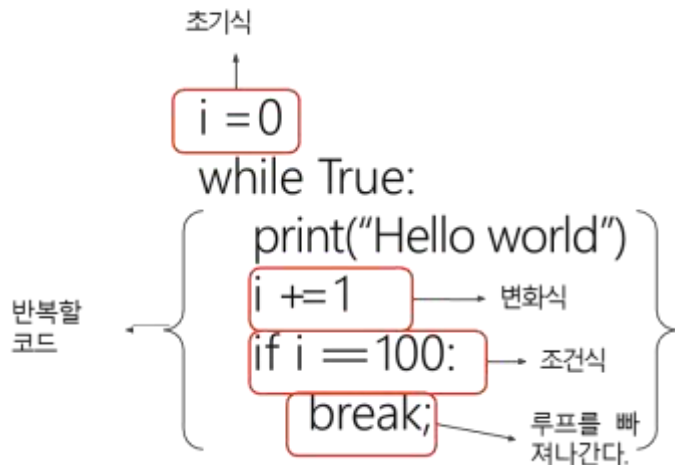
반복문 제어 - break

- ✓ for, while문에서 break는 해당 반복문을 완전히 빠져나간다.
- ✓ 보통 반복문 안에서 특정 조건에 부합할 때 해당 반복문을 빠져나가기 위해 사용.
 - 따라서 break는 if문과 함께 쓰이는게 보통이다.

```
# 10000회 반복
for _ in range(10000):
    print(_)
    if _ == 100:
        break
```

반복문 제어 - break

- ✓ while 조건식은 True이기 때문에 항상 반복을 진행한다. (무한 반복)
- ✓ 하지만 i의 값이 100일 때 if 조건문이 참이기 때문에 break를 만나 반복문이 종료된다.



반복문 제어 - break

- ✓ 0부터 100까지 자연수의 총합을 구하는 프로그램을 구현해보세요.
 - while문과 break를 활용해서 작성해보세요.

```
i = 0  
total = 0  
while True:
```

이 부분을 완
성해보세요.

```
print(total)
```

5050

반복문 제어 - break

- ✓ 0부터 100까지 자연수의 총합을 구하는 프로그램을 구현해보세요.
 - while문과 break를 활용해서 작성해보세요.

```
i = 0
total = 0
while True:
    total += i
    i = i + 1
    if i > 100:
        break
print(total)
```

5050

반복문 제어 - break

- ✓ 무한 반복문은 사용자로부터 원하는 값을 입력받을 때까지 반복을 해야할 때 유용하게 사용한다.
- ✓ 아래 예제는 사용자가 'q'를 입력받을 때까지 반복을 진행한다.

```
while True:
    input_value = input("q를 입력하면 종료됩니다.")
    if input_value == "q": break
print("프로그램을 종료합니다.")
```

q를 입력하면 종료됩니다. 진짜?

q를 입력하면 종료됩니다. 정말?

q를 입력하면 종료됩니다. q

프로그램을 종료합니다.

반복문 제어 - break

- ✓ 아래 소스 코드를 완성하여 0부터 100 사이의 숫자 중 3으로 끝나는 숫자만 출력되게 만드세요.

```
i = 0  
while True:
```

이 부분을 완
성해보세요.

```
    print(i, end=' ')  
    i += 1
```

3 13 23 33 43 53 63 73 83 93

반복문 제어 - break

✓ 아래 소스 코드를 완성하여 0부터 100 사이의 숫자 중 3으로 끝나는 숫자만 출력되게 만드세요.

```
i = 0
while True:
    if i % 10 != 3:
        i += 1
        continue
    if i > 100:
        break
    print(i, end=' ')
    i += 1
```

3 13 23 33 43 53 63 73 83 93

반복문 제어 - continue

- ✓ continue는 제어흐름을 유지하고, continue가 실행된 반복 횟수의 코드 실행만 건너뛰는다.
- ✓ 즉 continue를 만나면 가장 가까운 반복문으로 다시 돌아간다.

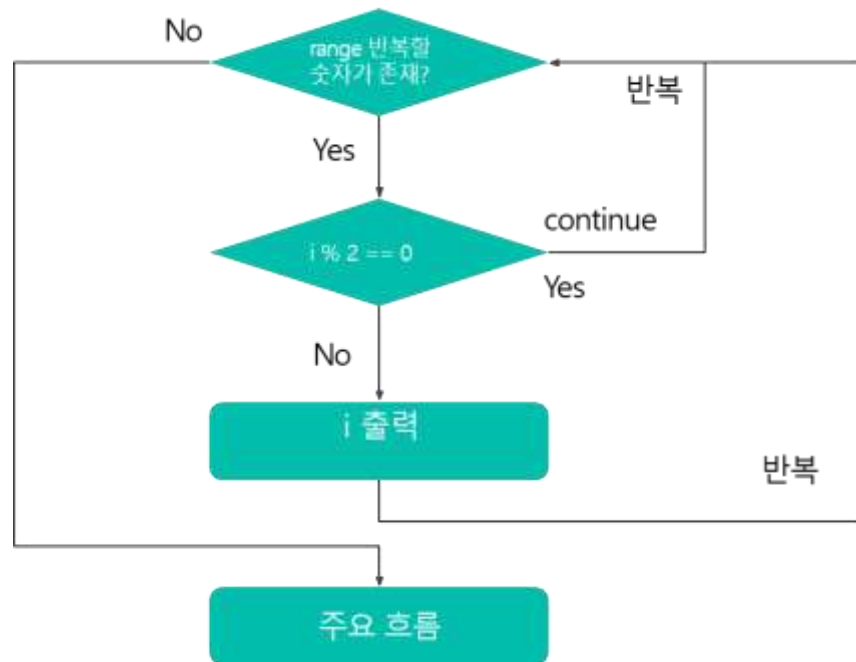
```
for i in range(100):  
    if i % 2 == 0:  
        continue  
    print(i, end="\t")
```

0부터 99까지 증가하면서 100번 반복
i를 2로 나누었을 때 나머지가 0면 짝수
아래 코드를 실행하지 않고 건너뛰

1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37
39	41	43	45	47	49	51	53	55	57	59	61	63	65	67	69	71	73	75
77	79	81	83	85	87	89	91	93	95	97	99							

반복문 제어 - continue

- ✓ continue는 제어흐름을 유지하고, continue가 실행된 반복 횟수의 코드 실행만 건너뛰는다.
- ✓ 즉 continue를 만나면 가장 가까운 반복문으로 다시 돌아간다.



반복문 제어 - continue

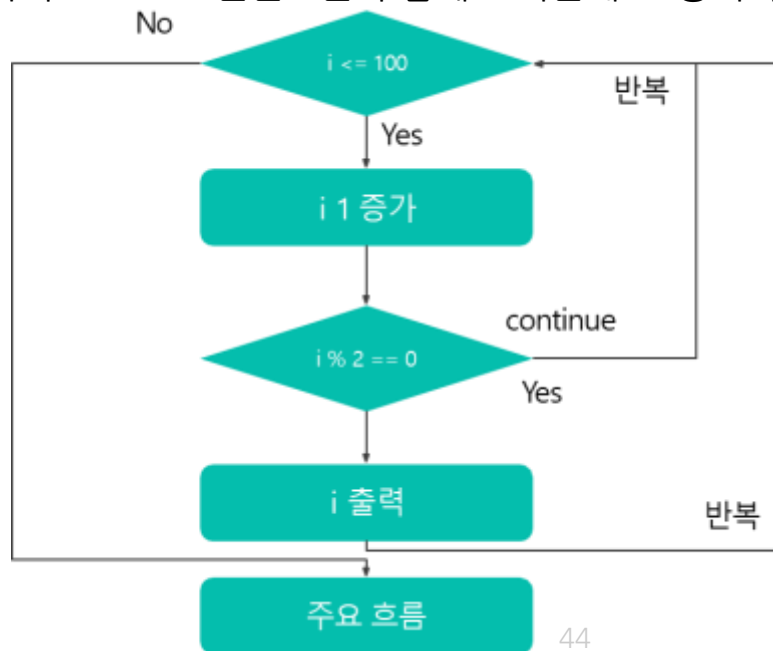
- ✓ while문에서도 continue 동작은 동일.
- ✓ 보통 반복문 안에서 특정 조건에 부합할 때 해당 반복 회차만 빠져나가기 위해 사용.
 - 따라서 continue문은 if문과 함께 쓰이는게 보통이다.

```
i = 0
while i < 100:      # i가 100보다 작을 때 반복. 0부터 99까지 증가하면서 100번 반복
    i += 1          # i를 1씩 증가시킴
    if i % 2 == 0:  # i를 2로 나누었을 때 나머지가 0이면 짝수
        continue   # 아래 코드를 실행하지 않고 건너뛰
    print(i, end="\t")
```

1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37
39	41	43	45	47	49	51	53	55	57	59	61	63	65	67	69	71	73	75
77	79	81	83	85	87	89	91	93	95	97	99							

반복문 제어 - continue

- ✓ while문에서도 continue 동작은 동일.
- ✓ 보통 반복문 안에서 특정 조건에 부합할 때 해당 반복 회차만 빠져나가기 위해 사용.
 - 따라서 continue문은 if문과 함께 쓰이는게 보통이다.



반복문 제어 - 연습 문제

✓ 반복문 연습문제

- 1-100까지의 숫자 중 3의 배수이면서 5의 배수인 3번째 숫자를 찾으세요.
- for/while 2개 모두 작성.

3번째 3과 5의 최소 공배수: 45

반복문 제어 - 연습 문제

✓ 반복문 연습문제 해답

```
count = 0
num = 1
while True:
    if num % 3 == 0 and num % 5 == 0:
        count += 1
        if count == 3: break
    num += 1
print(f"3번째 3과 5의 최소 공배수: {num}")
```

반복문 제어 - 연습 문제

✓ 반복문 연습문제

- 문자열에서 모음을 제거하세요.
- for/while 2개 모두 작성.

```
text = "This is an example string"  
vowels = "aeiou"  
result = ""  
....  
print(result)
```

Ths s n xmpl strng

반복문 제어 - 연습 문제

✓ 반복문 연습문제 해답

```
text = "This is an example string"
vowels = "aeiou"
result = ""
i = 0
while i < len(text):
    if text[i].lower() in vowels: # 문자가 모음일 경우 건너뛰기
        i += 1
        continue
    result += text[i]
    i += 1
print(result)
```


반복문 제어 - 연습 문제

✓ 반복문 연습문제 - 영화 티켓 예매하기

- 사용자는 영화 티켓을 예약할 때 이름을 입력해야 합니다.
- 이름에 **공백이 포함**되어 있으면 예약이 불가능하며, 다시 입력해야 합니다.
- 사용자가 **q** 또는 **Q**를 입력하면 프로그램이 종료됩니다.
- 프로그램 종료 시 지금까지 예약된 모든 이름을 출력합니다.

```
영화 티켓 예약 프로그램 (q/Q를 누르면 종료)
이름에 공백이 포함되어 있습니다. 다시 입력해주세요.
홍길동님, 티켓 예약이 완료되었습니다!
kim님, 티켓 예약이 완료되었습니다!
```

```
프로그램을 종료합니다. 안녕히 가세요!
```

```
예약된 이름 목록:
- 홍길동
- kim
```

리스트 컴프리헨션(list comprehension)

김지성 강사

리스트 컴프리헨션

- ✓ 파이썬의 자료구조(list, dictionary, set)을 쉽고, 짧게 한 줄로 만들 수 있는 파이썬 문법으로 가독성과 [속도 측면](#)에서 이점이 있다.
- ✓ 조건문을 걸거나 다중 반복문을 사용할 수 있다.
 - 하지만 기본적으로 가독성을 위해서 복잡한 코드는 리스트 컴프리헨션 사용을 지양한다.

```
numbers = []  
for n in range(1, 11):  
    numbers.append(n)  
print(numbers)
```

```
print([x for x in range(1,11)])
```

리스트 컴프리헨션 - 속도 측정

- ✓ 1-9999개의 리스트를 만드는데 1000번 반복 실행했을 때의 속도 차이를 보면 실제로 큰 차이로 속도 차이가 존재한다.

```
import timeit

# 반복문 방식
loop_time = timeit.timeit(
    stmt="""
numbers = []
for n in range(1, 10000):
    numbers.append(n)
""",
    number=1000 # 1000번 반복 실행
)

# 리스트 컴프리헨션 방식
comprehension_time = timeit.timeit(
    stmt="[x for x in range(1, 10000)]",
    number=1000 # 1000번 반복 실행
)

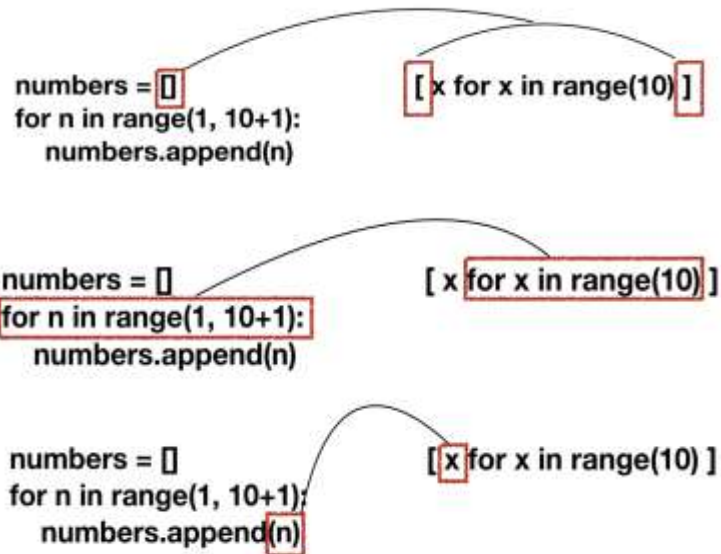
loop_time, comprehension_time

(0.3206181999994442, 0.21216620004270226)
```

리스트 컴프리헨션

✓ 대괄호[]를 통해 생성하는 방법은 동일

1. 반복문을 리스트 내부에 작성하여 반복
2. for문에서 반복되는 변수를 가장 앞에 작성




리스트 컴프리헨션 조건

✓ 1개의 if조건문을 사용하면 반복문 뒤에 위치한다.

1. 조건이 참이라면 리스트에 추가가 된다.
2. 조건이 거짓이라면 리스트에 추가되지 않고 다시 반복한다.

```
even_numbers = []  
for n in range(1, 10+1):  
    if n % 2 == 0:  
  
even_numbers.append(n)  
print(even_numbers)
```



```
print([x for x in range(1, 10+1) if x % 2 == 0])
```

리스트 컴프리헨션 중첩 반복문

- ✓ 2개 이상의 반복문을 사용할때는 [변수 for [표현식] for [표현식]]의 형태로 표현한다.
- ✓ 3개이상의 반복문은 가독성이 떨어지기 때문에 사용을 지양한다.

```
[(x, y) for x in [1, 2, 3] for y in [10, 20, 30]]
```

```
[(x, y) for x in [1, 2, 3] if x%2==1 for y in [10, 20, 30]]
```

```
[(x, y) for x in [1, 2, 3] for y in [11, 22, 33] if y%2==1]
```

```
[(x, y) for x in [1, 2, 3] for y in [10, 20, 30]]
```

```
[(1, 10),  
(1, 20),  
(1, 30),  
(2, 10),  
(2, 20),  
(2, 30),  
(3, 10),  
(3, 20),  
(3, 30)]
```

```
[(x, y) for x in [1, 2, 3] if x%2==1 for y in [10, 20, 30]]
```

```
[(1, 10), (1, 20), (1, 30), (3, 10), (3, 20), (3, 30)]
```

```
[(x, y) for x in [1, 2, 3] for y in [11, 22, 33] if y%2==1]
```

```
[(1, 11), (1, 33), (2, 11), (2, 33), (3, 11), (3, 33)]
```