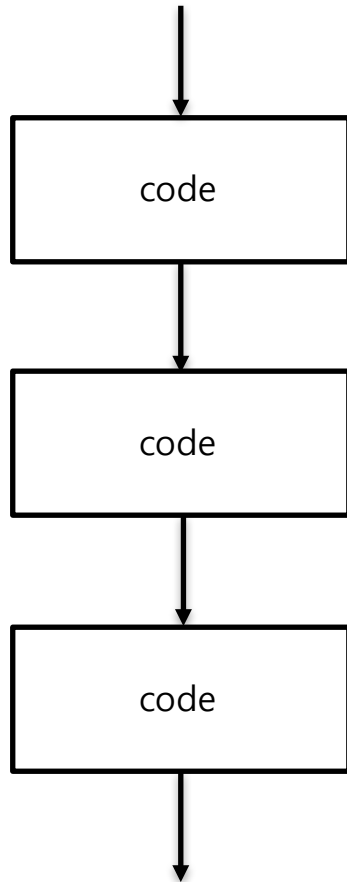


오늘의 강의 목표

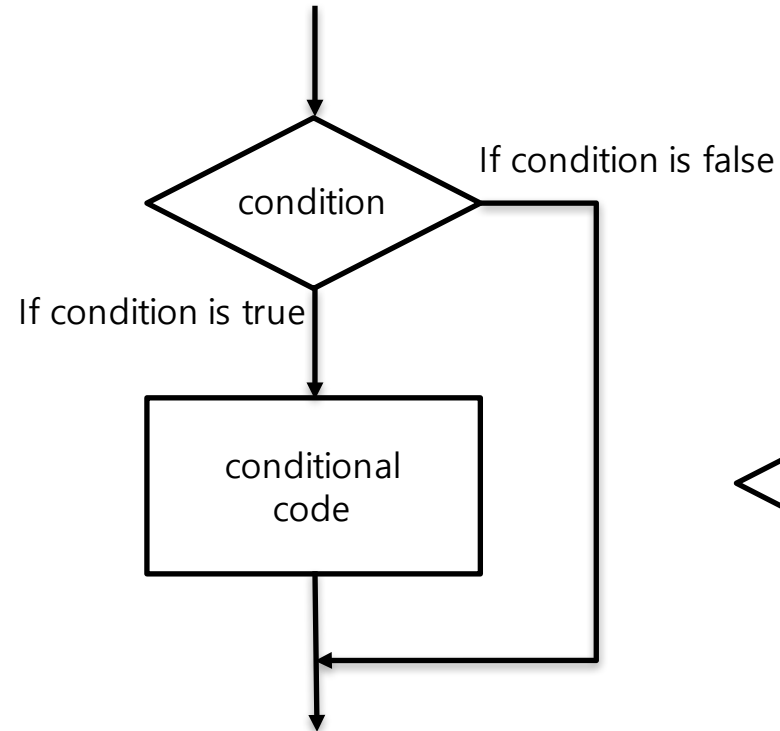
- 흐름 제어 (Flow Control)에 대한 이해
- if ~ else 에 대한 이해
- while에 대한 이해
- for에 대한 이해
- break, continue, pass에 대한 이해
- Loop else에 대한 이해

Program Flow Control

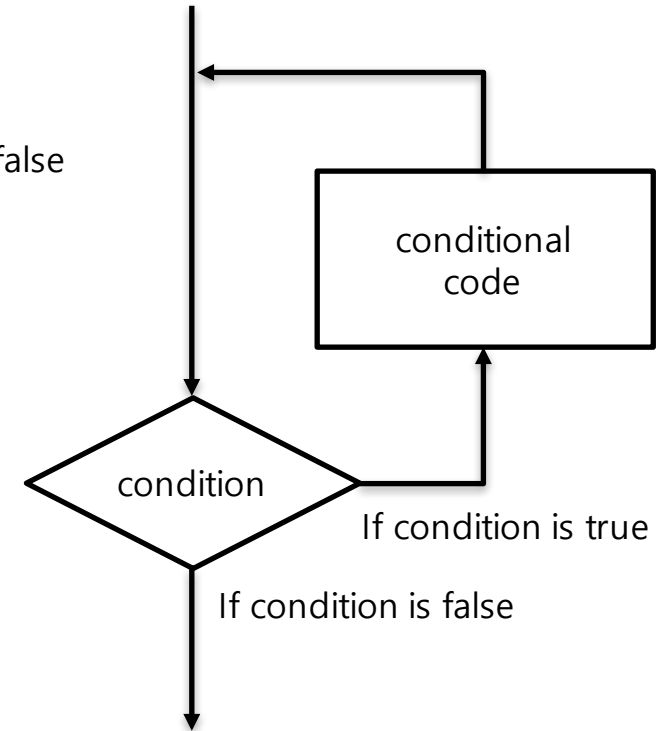
Sequential
Execution



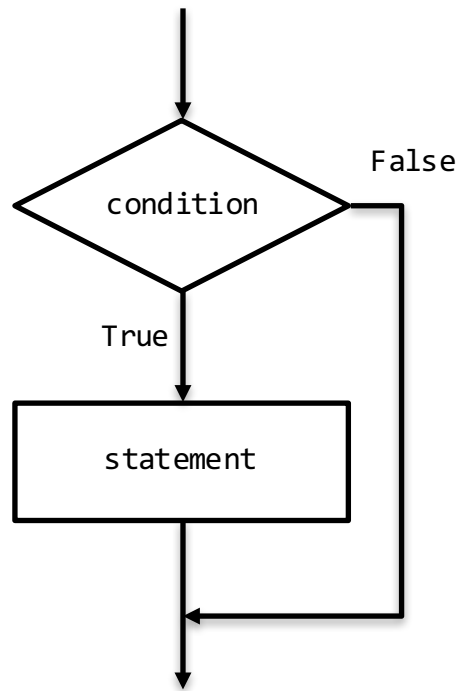
Conditional
Execution



Repeated
Execution



Conditional Execution

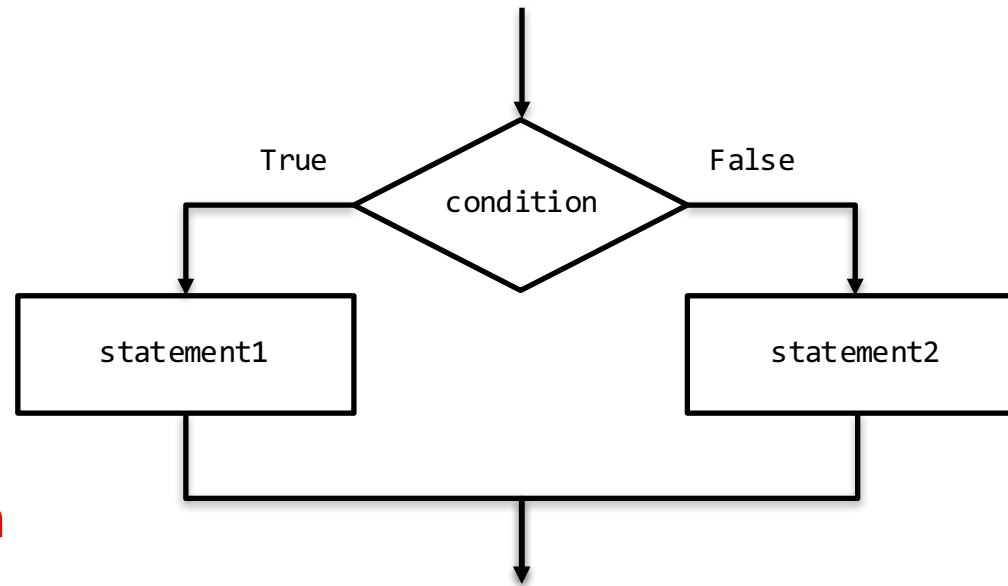


condition

```
if score > 90:  
    print("Grade = A+")
```

condition이 참일 경우에만 실행

Alternative Execution



condition

```
if score > 90:  
    print("PASS")  
else:  
    print("FAIL")
```

condition이 참일 경우에만 실행

condition이 거짓일 경우에만 실행

Compound Statement

- 여러 statement의 집합
- 같은 indentation level에 속해야 함

```
if score > 90:
```

```
    print("Grade = A+")  
    print("Congratulations!")  
    grade = "A+"
```

Compound Statement



같은 indentation level

Conditional Expression

- Condition들은 logical operator로 연결될 수 있음 (and, or, not)

```
if score > 90 and absence < 3:  
    print("Grade = A+")
```

```
if score < 40 or absence > 6:  
    print("Grade = F")
```

```
if not score > 30:  
    print("Grade = F")
```

Common Error

```
if a == 0 or a == 1:  
    print("binary")
```

Correct

```
if a == 1 or a == 0:  
    print("binary")
```

```
if a == 0 or 1:  
    print("binary")
```

Wrong

```
if a == 1 or 0:  
    print("binary")
```

a가 0일 때와 1일 때
각각의 실행 결과는?
그 이유는?

Interval Operator

- 어떤 값이 어떤 범위 안에 있는지 판단
- C, JAVA 등 다른 언어에서는 사용 불가

```
if 70 <= score < 80:  
    print("Grade = B+")
```

참고: <https://docs.python.org/3/reference/expressions.html#not-in>

다른 언어의 경우

```
if (0.1 < 0.2 < 0.3) {  
    ...  
}
```



True == 1 in C

```
if ((0.1 < 0.2) < 0.3) {  
    ...  
}
```

1 < 0.3

Nested Conditionals

- if나 else 안에 다시 if나 else가 들어감

```
if x > 0:
```

```
    if y > 0:
```

```
        print("Both x and y are positive")
```

```
    else
```

```
        print("x is positive but y is not")
```

Compound
Statement

```
else:
```

```
    if y > 0:
```

```
        print("y is positive but x is not")
```

```
    else
```

```
        print("Neither x nor y is positive")
```

Compound
Statement

Nested Conditionals

- else 사용시 indentation에 주의

Match {

```
if a > b:
    if c > d:
        print("A")
    else:
        print("B")
```

Match {

```
if a > b:
    if c > d:
        print("A")
else:
    print("B")
```

Chained Conditions

- 어떤 값이 양수, 음수, 0 인지 확인하는 예제



```
if x > 0:
    print("Positive")
if x == 0:
    print("zero")
if x < 0:
    print("negative")
```

- 컴퓨터는 무조건 세 개의 조건을 한번씩 체크
 - 비효율적
- 세 조건이 서로 독립이 아닌데 독립으로 표현
 - 오해 유발

Chained Conditions

- 어떤 값이 양수, 음수, 0 인지 확인하는 예제



```
if x > 0:
    print("Positive")
else:
    if x == 0:
        print("zero")
    else:
        print("negative")
```

- 조건 체크 최소화
- 상호 배제 (mutually exclusive) 관계로 표현됨
- But, 복잡한 indentation

Chained Conditions

- 어떤 값이 양수, 음수, 0 인지 확인하는 예제



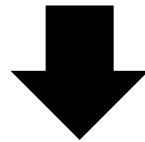
```
if x > 0:
    print("Positive")
elif x == 0:
    print("zero")
else:
    print("negative")
```

- 조건 체크 최소화
- 상호 배제 (mutually exclusive) 관계로 표현됨
- 명확한 indentation

Ternary Operator

[on_true] if [expression] else [on_false]

```
if x < y:  
    small = x  
else:  
    small = y
```



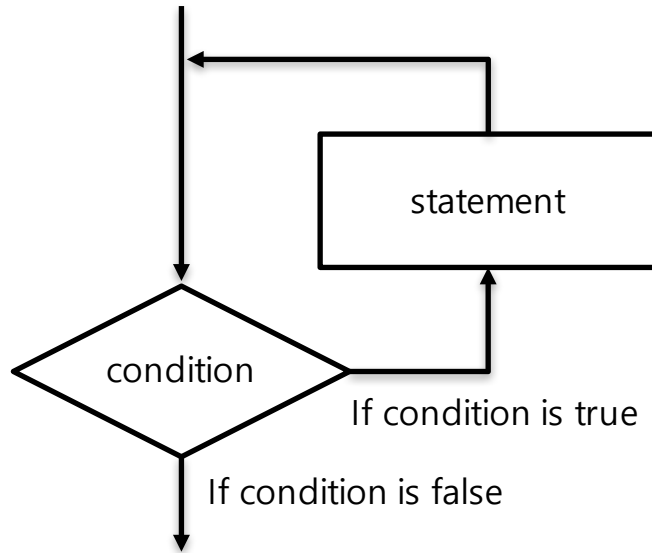
```
small = x if x < y else y
```

조건이 True일 경우

조건

조건이 False일 경우

While Loop



- 반복적인 일을 시키는 방법
- 특정 조건이 만족되면 반복

```
i = 1
sum = 0
while i <= 100:
    sum += i
    i += 1
print("Adding from 1 to 100 yields " + str(sum))
```

1에서 100까지의 합 계산

Nested While Loop

- While loop 안에 while loop
- Loop variable은 i, j, k를 사용하는 것이 관례

```
i = 1
while i <= 100:
    sum = 0
    j = i
    while j <= 100:
        sum += j
        j += 1
    print("Adding from " + str(i) + " to 100 yields " + str(sum))
    i += 1
```

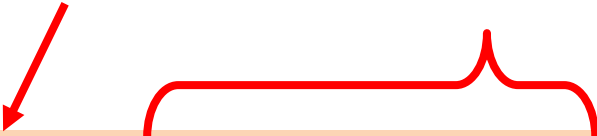


```
Adding from 1 to 100 yields 5050
Adding from 2 to 100 yields 5049
...
Adding from 100 to 100 yields 100
```


For Loop

- Sequence에 대해서 차례차례 실행

Loop index variable Sequence



```
for i in [0, 1, 2, 3, 4]:  
    print("i = ", i)
```

Sequence of integers

```
for i in ["red", "green", "blue"]:  
    print(i)
```

Sequence of strings

```
for i in ["red", 1, "green", 2, "blue", 3]:  
    print(i)
```

Sequence of mixed types

Nested For Loop

- For loop 안에 for loop

```
for i in [0, 1, 2]:  
    for j in [0, 1, 2]:  
        print("(i, j) = ", "(" , i, ", ", j, ")")
```



```
(i, j) = ( 0 , 0 )  
(i, j) = ( 0 , 1 )  
(i, j) = ( 0 , 2 )  
(i, j) = ( 1 , 0 )  
(i, j) = ( 1 , 1 )  
(i, j) = ( 1 , 2 )  
(i, j) = ( 2 , 0 )  
(i, j) = ( 2 , 1 )  
(i, j) = ( 2 , 2 )
```

range()

- range(n)
 - $[0, 1, 2, \dots, n - 1]$
 - n번 반복
- range(n1, n2)
 - $[n1, n1 + 1, n1 + 2, \dots, n2 - n1]$
 - $n2 - n1$ 번 반복
- range(n1, n2, n3)
 - $[n1, n1 + n3, n1 + 2 * n3, \dots, n1 + k * n3]$
 - $n1 + k * n3 < n2$ 일 때까지 ($n3 > 0$ 인 경우)
 - $n1 + k * n3 > n2$ 일 때까지 ($n3 < 0$ 인 경우)

range()

Sequence를 list로 변환



```
>>> list(range(3))  
[0, 1, 2]  
>>> list(range(3, 10))  
[3, 4, 5, 6, 7, 8, 9]  
>>> list(range(10, 3))  
[]  
>>> list(range(-5, 10, 3))  
[-5, -2, 1, 4, 7]  
>>> list(range(10, -5, -3))  
[10, 7, 4, 1, -2]  
>>> list(range(-5, 10, -3))  
[]
```

For vs While

- For와 While 비교

```
for i in range(1, 101):  
    sum = 0  
    for j in range(i, 101):  
        sum += j  
    print("Adding from " + str(i) + " to 100 yields " + str(sum))
```

For

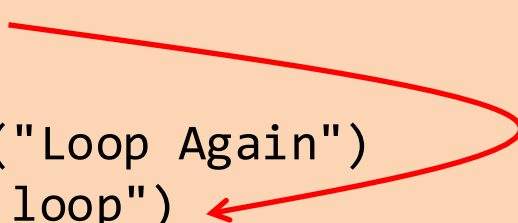
```
i = 1  
while i <= 100:  
    sum = 0  
    j = i  
    while j <= 100:  
        sum += j  
        j += 1  
    print("Adding from " + str(i) + " to 100 yields " + str(sum))  
    i += 1
```

While

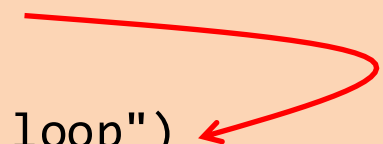
break

- Loop 밖으로 탈출 (While, For 모두 사용 가능)

```
while True:
    key = input("Enter q to quit: ")
    if key == 'q':
        break
    else:
        print("Loop Again")
print("Out of loop")
```



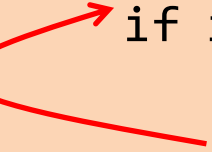
```
for i in ['apple', 'strawberry', 'blueberry', 'pineapple']:
    if i == 'blueberry':
        break
    print(i)
print("Out of loop")
```



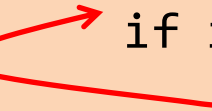
continue

- 다음 루프 시작으로 (이후 skip의 의미)

```
i = 0
while i < 10:
    if i % 3 == 0:
        i += 1 수동으로 index 증가 필요
        continue
    print(i)
    i += 1
```



```
for i in ['apple', 'strawberry', 'blueberry', 'pineapple']:
    if i == 'blueberry':
        continue
```



자동으로 print(i)
다음 item으로
이동

pass

- 아무것도 하지 않음을 명시적으로 표현

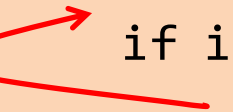
```
while True:  
    pass
```

```
for i in ['apple', 'strawberry', 'blueberry', 'pineapple']:  
    if i == 'blueberry':  
        pass  
    else:  
        print(i)
```

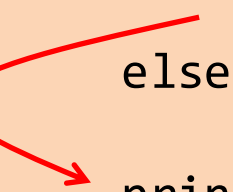

continue vs pass

- continue와 pass 차이점 비교

```
for i in ['apple', 'strawberry', 'blueberry', 'pineapple']:
    if i == 'blueberry':
        continue
    print(i)
    print('-----')
```



```
for i in ['apple', 'strawberry', 'blueberry', 'pineapple']:
    if i == 'blueberry':
        pass
    else:
        print(i)
    print('-----')
```



위 두 코드의 차이점은?

Loop else

- Loop 정상 종료 때 실행
- break로 종료된 경우는 실행하지 않음

```
count = 0
while count < 10:
    key = input("Enter q to quit: ")
    if key == 'q':
        print("Break before finish")
        break
    else:
        count += 1
        print("Loop Again")
else:
    print("Finishing 10 times loop")
```

HW

- 이전 숙제에 더해서 아래 두 가지 기능 추가
 - s : suspend & resume
 - q : quit
- 단, break와 continue를 사용해야 함

Questions



메모

- Loop 전까지 진도 나감
- 전반적으로 너무 dry