# Flow Control and Loops

講者：Isaac

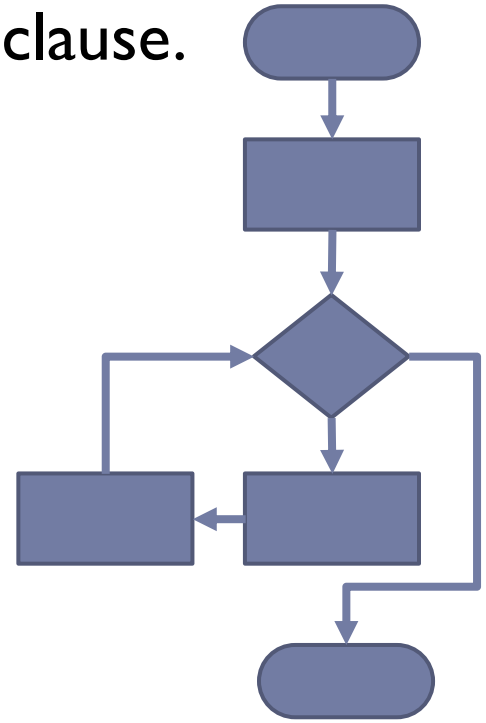
# Outline

- Flow Control
- If Statements
  - if
  - else
  - elif
- Loops
  - While
  - For
- Iterator
- Comprehensions

# Flow Control

# Flow Control

- In computer science, **control flow** (or **flow of control**) is the order in which individual statements, instructions or function calls of an imperative program are executed or evaluated.
- Start with condition and following by clause.
- Flowchart
  - are used in designing and documenting simple processes or programs.
  - help visualize what is going on and thereby help understand a process.



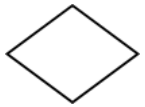
# Flowchart

- Building Blocks

| Shape | Name | Description |
|---|---|---|
| | Flowline Arrowhead | Shows the process's order of operation.A line coming from one symbol and pointing at another. Arrowheads are added if the flow is not the standard top-to-bottom, left-to right. |
| | Terminal | Indicates the beginning and ending of a program or sub-process. Represented as a stadium, oval or rounded (fillet) rectangle. They usually contain the word "Start" or "End", or another phrase signaling the start or end of a process, such as "submit inquiry" or "receive product". |
| | Process | Represents a set of operations that changes value, form, or location of data. Represented as a rectangle. |

# Flowchart

- Building Blocks

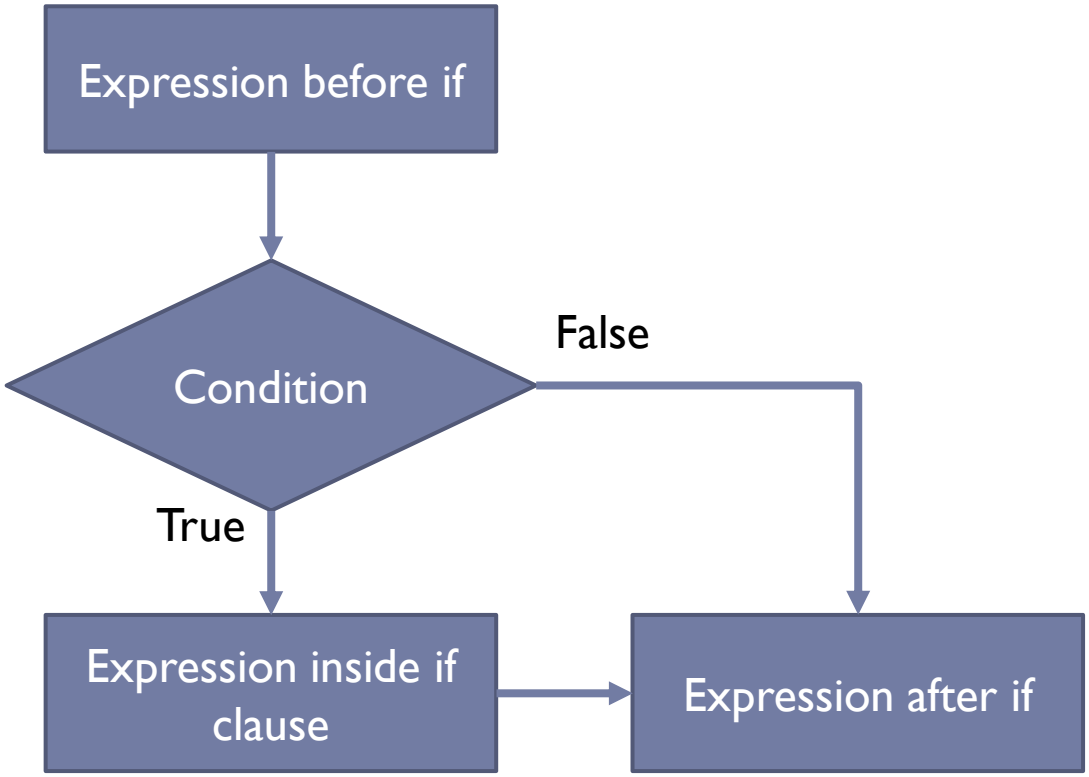
| Shape | Name | Description |
|---|---|---|
|  | Decision | Shows a conditional operation that determines which one of the two paths the program will take. The operation is commonly a yes/no question or true/false test. Represented as a diamond (rhombus). |
|  | Input Output | Indicates the process of inputting and outputting data, as in entering data or displaying results. Represented as a parallelogram. |

# If Statements

# If Statements

- If
- Else
- Elif

# If flow control



# If Syntax

- Start with If (condition), then add ':' to end the sentence.
- Use space*4 to add indentation, then add clauses.
- Conditions usually comes with operators to define True or False(zero, empty, none).
- If the condition is true, then execute the following code block.
- Example:

Conditions

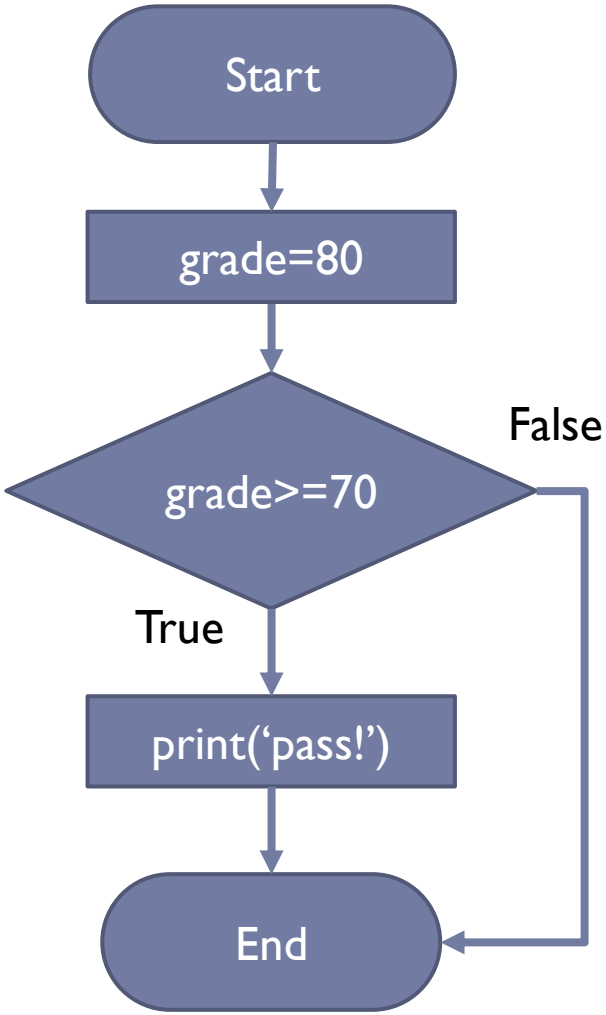
```
>>> grade = 80
>>> if (grade >= 70):
...     print('pass!')
```

Indentation

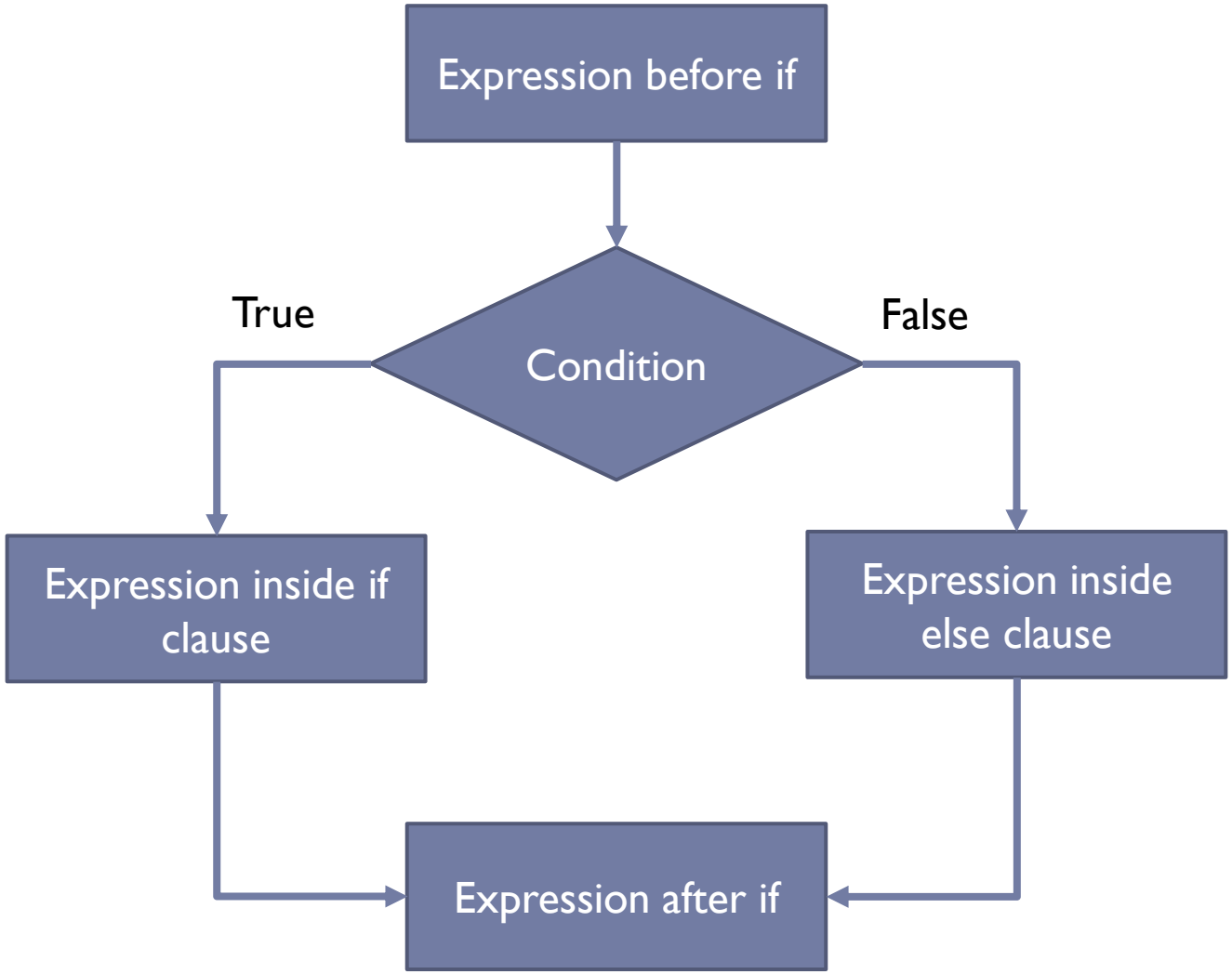
# If

- Example:
  - Code & Flowchart

```
>>> grade = 80
>>> if (grade >= 70):
...     print('pass!')
```



# else flow control



# else Syntax

- The else keyword catches anything which isn't caught by the preceding conditions.

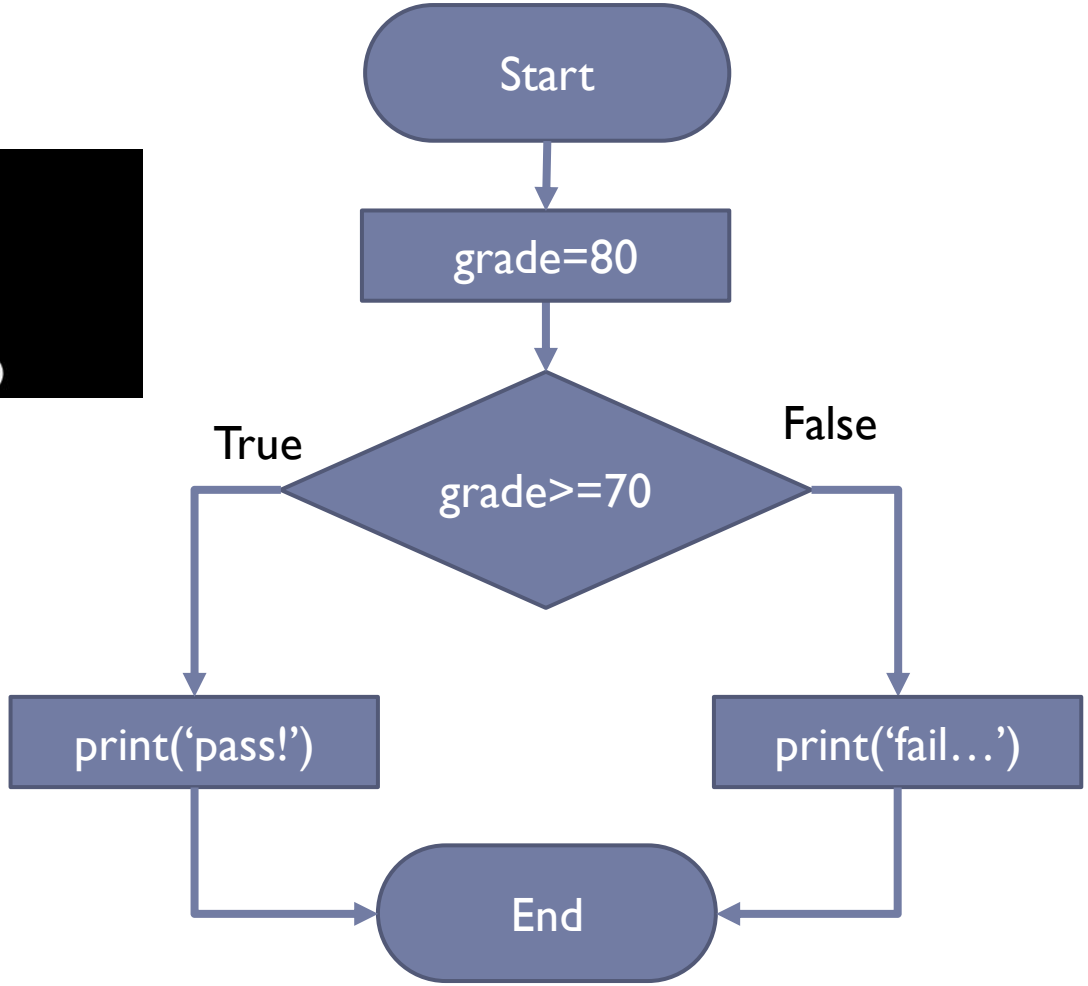
```
>>> grade = 80
>>> if (grade >= 70):
...     print('pass!')
... else:
...     print('fail...')
```

Indentation

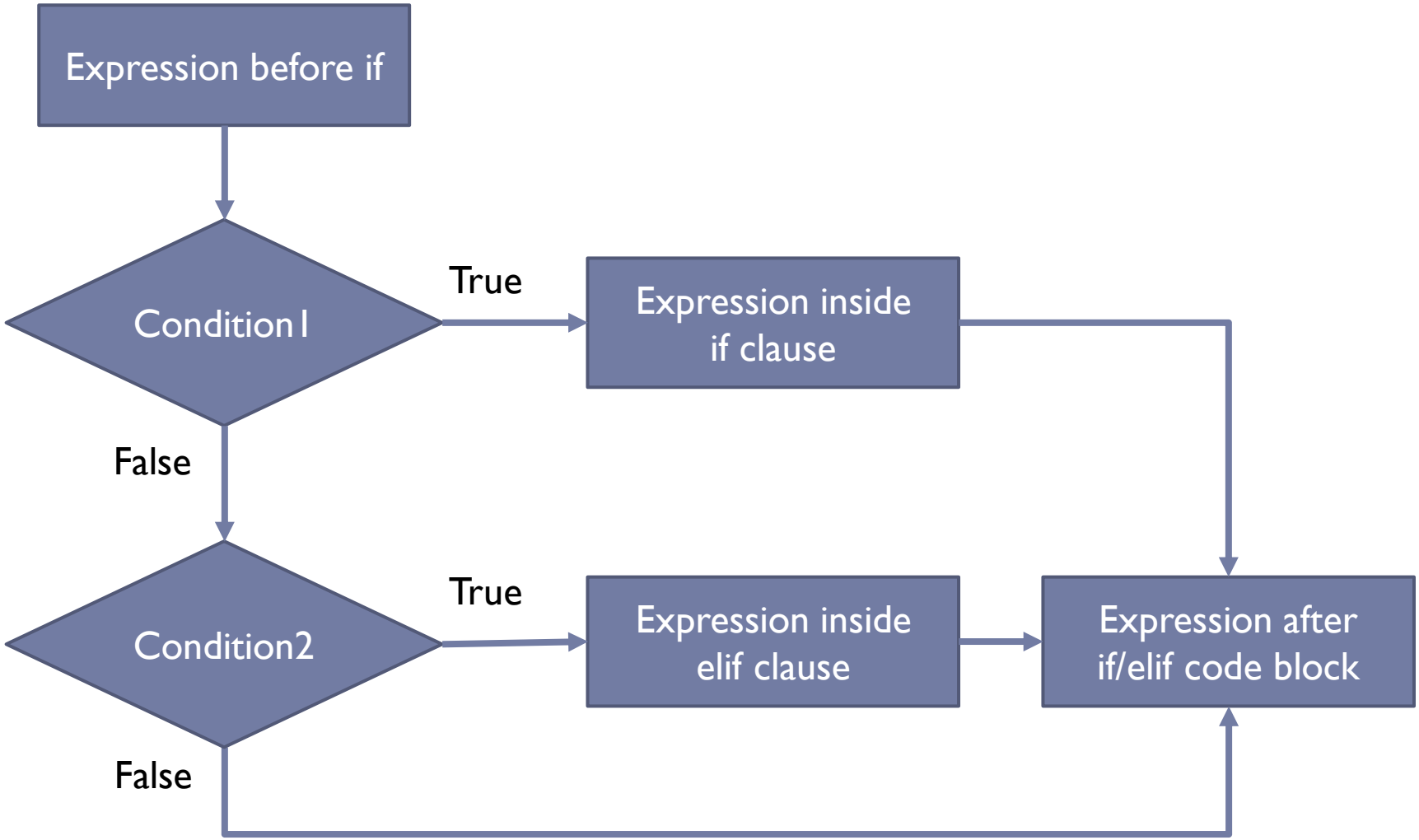
# else

- Example:
  - Code & Flowchart

```
>>> grade = 80
>>> if (grade >= 70):
...     print('pass!')
... else:
...     print('fail...')
```



# elif flow control



# elif Syntax

- "if the previous conditions were not true, then try this condition".

```
>>> grade = 80
>>> if (grade >= 70):
...     print('pass!')
... elif (grade < 60):
...     print('fail!')
```

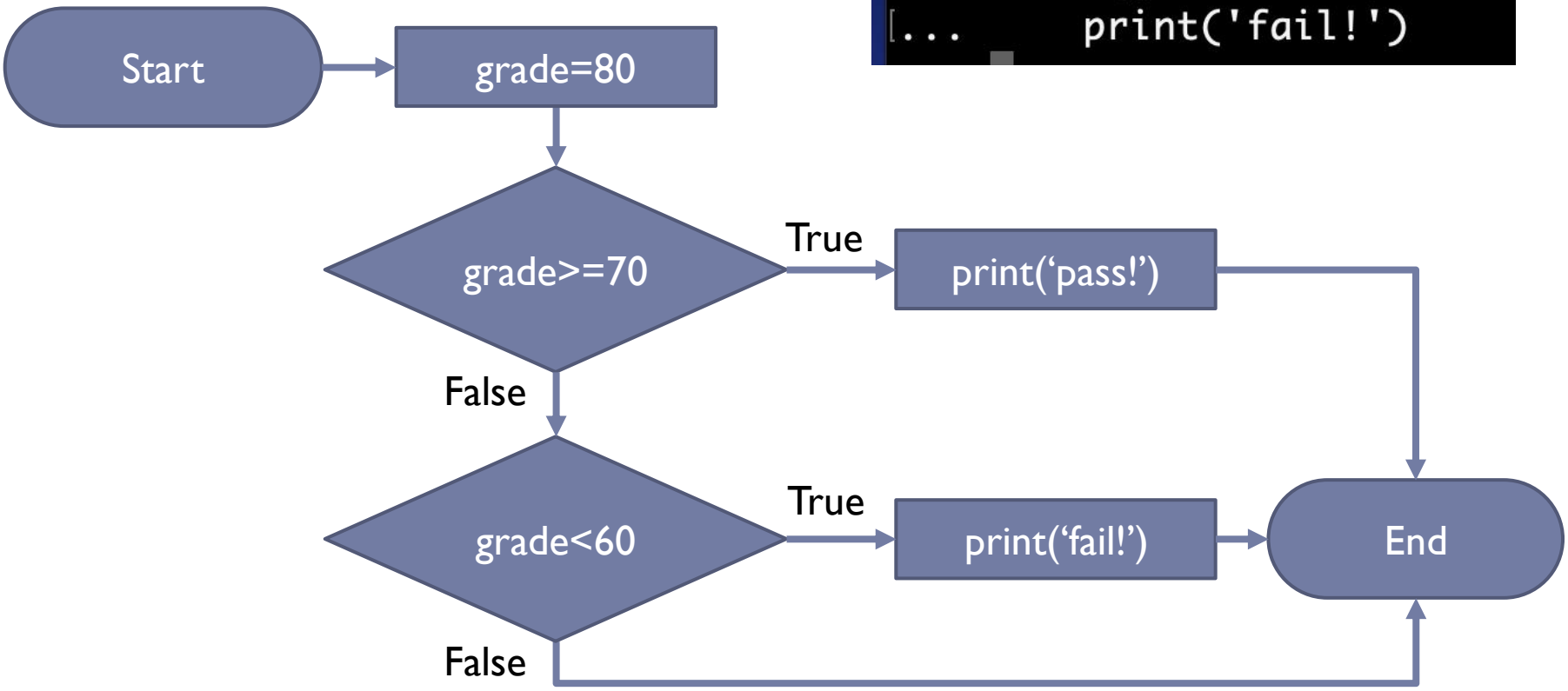
Elif Conditions

Indentation

# elif

- Example:
  - Code & Flowchart

```
>>> grade = 80
>>> if (grade >= 70):
...     print('pass!')
... elif (grade < 60):
...     print('fail!')
```



# Nested if Statement

- if statements inside if statements
- Syntax Example:

```
If (condition 1):
    if (condtion A):
        code block A
    elif (condition B):
        code block B
    else:
        code block C
else:
    code block 2
```

# Example

```python
print('Is Leap Year or not.')
year = input('Please enter Year (ex:2019) : ')

leap4 = int(year) % 4
leap100 = int(year) % 100
leap400 = int(year) % 400

if leap4 == 0:
    if leap100 != 0:
        print(f'{year} is leap year.')

    elif leap400 == 0:
        print(f'{year} is leap year.')

    else:
        print(f'{year} is not leap year.')
else:
    print(f'{year} is not leap year.')

```

# Pass Statement

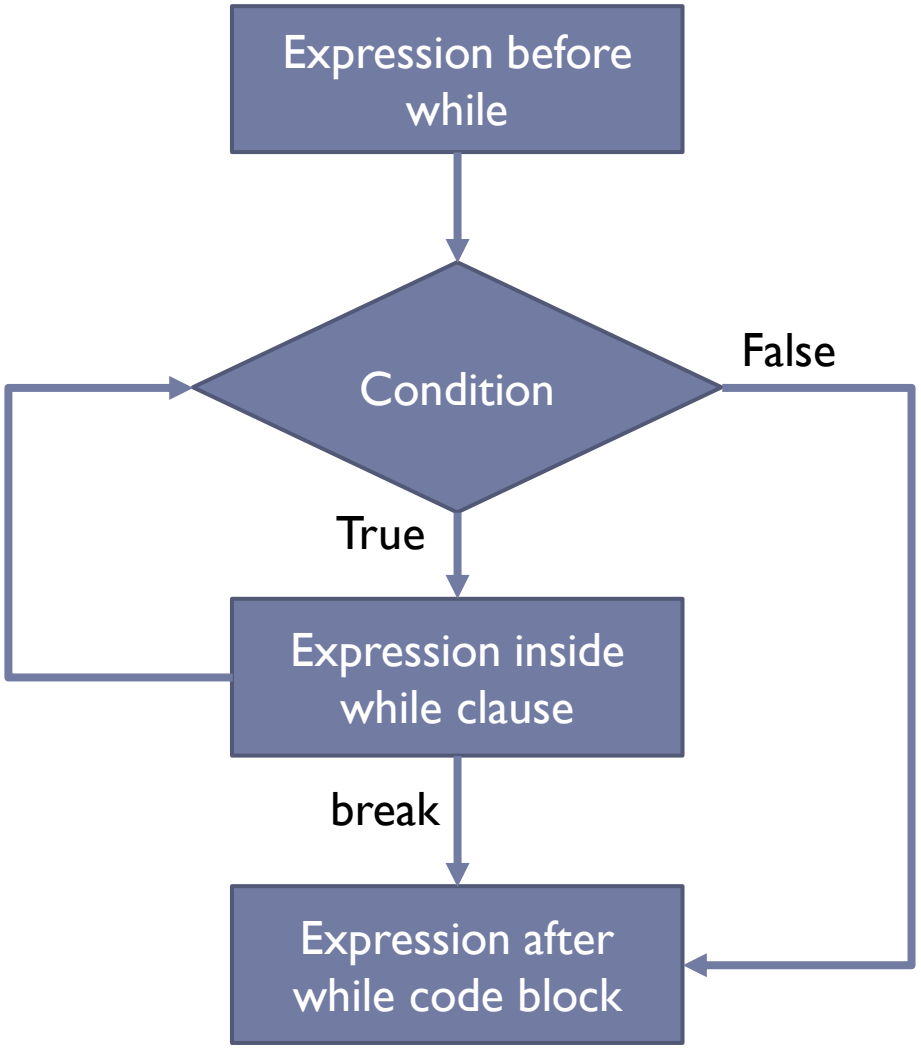
- if statements cannot be empty, put in the pass statement to avoid getting an error.
- Syntax Example:

```
if (condition A):
    pass
```

```
>>>
>>> a = 3
>>> b = 2
>>> if a < b:
...     pass
...
```

# While Loops

# while flow control



# While Loops

- With the while loop we can execute a set of statements as long as a condition is true.
- Under terminal, use CTRL+C to escape infinite loop.
- Syntax Example:

```
while (condition):
    code block
```

# Example

```python
answer = 3
guess = 0

while guess != answer:
    guess = int(input('Please make guess during 1~6. : '))
    if guess > answer:
        print('Hint: bigger than the answer.')
    elif guess < answer:
        print('Hint: smaller than the answer.')
    else:
        print('Bingo!')
```

```
Please make guess during 1~6. : 1
Hint: smaller than the answer.
Please make guess during 1~6. : 2
Hint: smaller than the answer.
Please make guess during 1~6. : 3
Bingo!
```

# break Statement

- With the break statement we can stop the loop even if the while condition is true:
- Syntax Example:

```
while (condition A):
    code block
    if (condition B):
        break
```

# Example

```python
answer = 3
guess = 0

while True:
    guess = int(input('Please make guess during 1~6. : '))
    if guess > answer:
        print('Hint: bigger than the answer.')
    elif guess < answer:
        print('Hint: smaller than the answer.')
    else:
        print('Bingo!')
        break
```

```
Please make guess during 1~6. : 1
Hint: smaller than the answer.
Please make guess during 1~6. : 2
Hint: smaller than the answer.
Please make guess during 1~6. : 3
Bingo!
```

# continue Statements

- With the continue statement we can stop the current iteration, and continue with the next:
- Syntax Example:

```
while (condition A):
    code block
    if (condition B):
        continue
```

# Example

```python
while True:
    x = input('Please enter a digit from 1~6: ')
    if x == '1':
        print('continue statement\n')
        continue
    elif x == '0':
        break
    else:
        print('Enter else block.')
    print('Still in while loop.\n')
print('Out of While Loop.')
```

```
Please enter a digit from 1~6: 1
continue statement

Please enter a digit from 1~6: 2
Enter else block.
Still in while loop.

Please enter a digit from 1~6: 3
Enter else block.
Still in while loop.

Please enter a digit from 1~6: 4
Enter else block.
Still in while loop.

Please enter a digit from 1~6: 0
Out of While Loop.
```

# else Statements

- With the else statement we can run a block of code once when the condition no longer is true:
- Syntax Example:

```
while (condition A):
    code block A
else:
    code block B
```

# Example

- While + continue + else

```python
n = 5
while n > 0:
    n = n - 1
    if n == 2:
        continue
    print(n)
else:
    print("Loop is finished")
```

```
4
3
1
0
Loop is finished
```

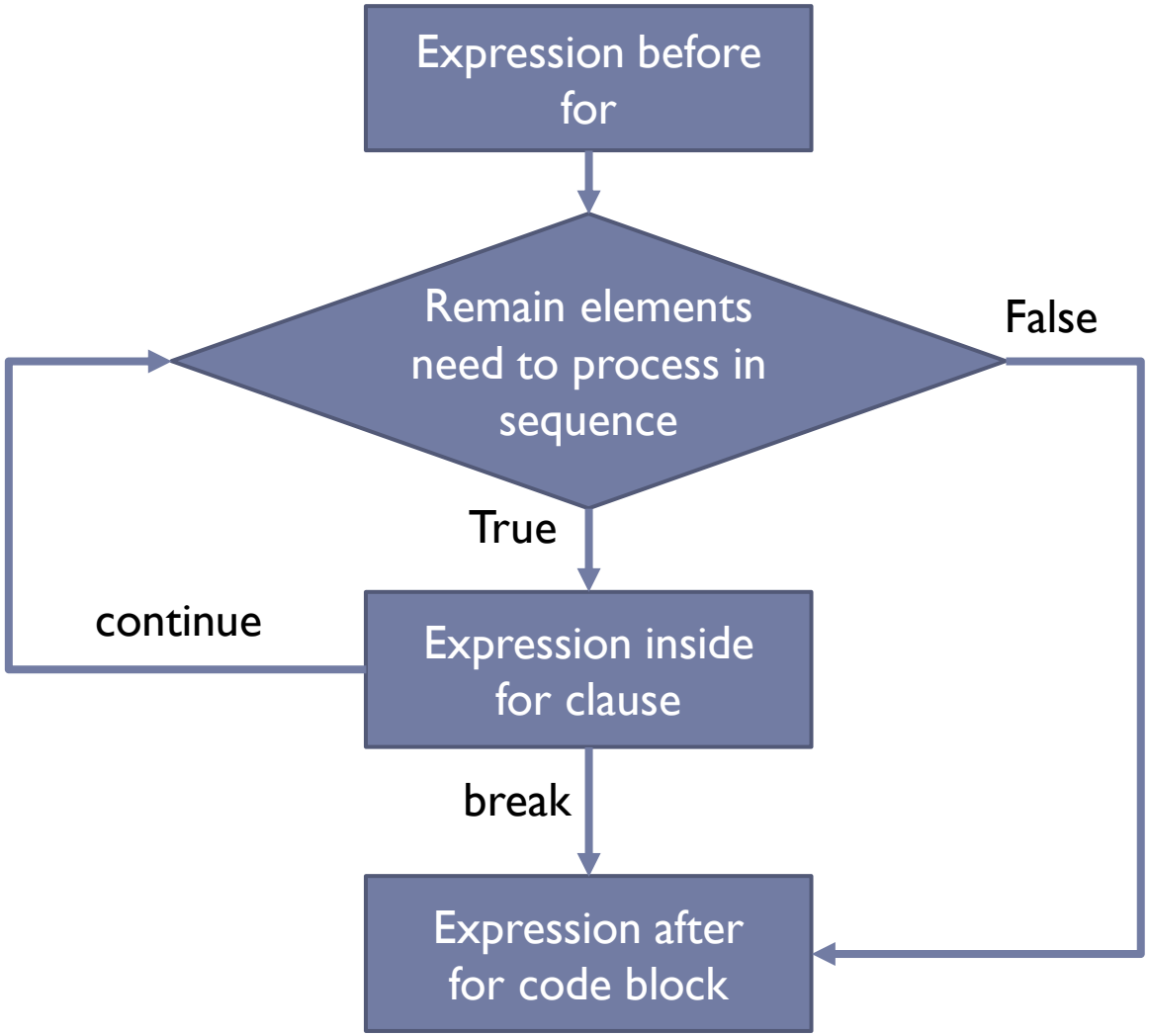
- While + break + else

```python
n = 5
while n > 0:
    n = n - 1
    if n == 2:
        break
    print(n)
else:
    print("Loop is finished")
```

```
4
3
```

# for Loops

# for flow control



# For Loops

- for loop is used for iterating over a sequence (list, tuple, dictionary, set or string).
- Syntax Example:

```
for <variable> in (sequence):
    code block
```

- Similar with while loops, usage usually with flow control statements(if/continue/break…).

# Example

```python
for c in 'Python':
    print(f'current character:{c}')
```

```
current character:P
current character:y
current character:t
current character:h
current character:o
current character:n
```

```python
fruits = ['watermelon', 'guava',  'strawberry']
for f in fruits:
    print(f'fruits: {f}')
```

```
fruits: watermelon
fruits: guava
fruits: strawberry
```

# for Loops

- Syntax Example:
  - Nested for statements.
  - Combined with continue/break/if-else statements.

```
for <variable> in <sequence>:
    code block
    for <variable> in <sequence>:
        if (condition A):
            continue
    if (condition B):
        break
else:
    code block
```

**Nested for statements**

**continue/break/if-else statements**

# Example

- Nested For loops

```python
i = 2
while(i < 100):
    j = 2
    while(j <= (i / j)):
        if not( i % j ):
            break
        j = j + 1
    if j > i / j:
        print(f'{i} is prime')
    i = i + 1

print('end...')
```

```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
```

# Iterator

# iterator

- object that contains a countable number of values which can be all traversed through.
- Example:
  - iter(), next()

List = [‘A’, ‘B’, ‘C’, ‘D’]

```
>>> it = iter(list)
>>> for x in it:
...     print(x, end='')
...
ABCD>>>
```

```
>>> it = iter(list)
>>> print(next(it))
A
>>> print(next(it))
B
>>> print(next(it))
C
>>> print(next(it))
D
```

# range Statement

- To loop through a set of code a specified number of times, we can use the range() function, The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.
- Syntax Example:

```
for <variable> in range(number) :
    code block
```

```
for <variable> in range(start, end, step) :
    code block
```

# Example

```python
for i in range(1, 3):
    print(i)
```

```
1
2
```

```python
for i in range(1, 5):
    print(i)
```

```
1
2
3
4
```

```python
for i in range(1, 10, 2):
    print(i)
```

```
1
3
5
7
9
```

# enumerate Statements

- The enumerate() function takes a collection (e.g. a tuple, list, set) and returns it as an enumerate object.
- The enumerate() function adds a counter as the key of the enumerate object.
- Usually combine with for loops.
- Syntax Example:

```
enumerate(<sequence>, <start>)
```

# enumerate Statements

**Example-1**

```
>>> fruits = ['Apple', 'Orange', 'Melon']
>>> for fruit in enumerate(fruits):
...     print(fruit)
...
(0, 'Apple')
(1, 'Orange')
(2, 'Melon')
```

**Example-2**

```
>>> for fruit in enumerate(fruits, 10):
...     print(fruit)
...
(10, 'Apple')
(11, 'Orange')
(12, 'Melon')
```

**Example-3**

```
>>> for count, fruit in enumerate(fruits):
...     print(count, fruit)
...
0 Apple
1 Orange
2 Melon
```

# Comprehension

# List comprehension

- To generate list in an elegance way.
- Syntax:
  - list = [(expression) (for loop) (if statement)]
- Example:

**list generated by for loop**

```
>>> x =[]
>>> for i in range(10):
...     x.append(i)
...
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

**list generated by list comprehension**

```
>>> x = [i for i in range(10)]
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

# List Comprehension

- Example:
  - Combine with if/else statements:

**for loop**

```
>>> x = []
>>> for i in range(20):
...     if i % 5 == 0:
...         x.append(i)
...
>>> x
[0, 5, 10, 15]
```

**list comprehension**

```
>>> x = [i for i in range(20) if i % 5  == 0]
>>> x
[0, 5, 10, 15]
>>>
```