PRACTICAL 3

STRUCTURED PROGRAMS

Presented by Qing Lyu / 吕晴

Adapted from Fall 2017 tutorial

WHAT WE'VE TALKED ABOUT LAST TIME

- What is function
- NLTK functions for text normalization

A quick test: with the function `quadratic(a, b, c)` you wrote last time, run

```
print(quadratic(1, 3, 1))
```

and then run

```
print(quadratic(1, 2, 3))
```

what's the output respectively?

Error? We'll talk about how to handle this today

STRUCTURED PROGRAMS/结构化程序: WHAT & WHY

- What?
  3 major structures:
  - Sequence/顺序结构
  - Selection/条件结构
  - Loop/循环结构
- Why?
  - A single command(statement/语句) usually cannot accomplish complex tasks
  - so we need multiple statements that are logically organized

SEQUENCE / 顺序结构

- The simplest structure - lines of code that are executed in order, w/o any branching or looping
- e.g.

```
import math
a, b, c = 1, 3, 1
delta = b ** 2 - 4 * a * c
x1 = (-b + math.sqrt(delta)) / (2 * a)
x2 = (-b - math.sqrt(delta)) / (2 * a)
print(x1, x2)
```

- N.B. x**y means x to the power of y

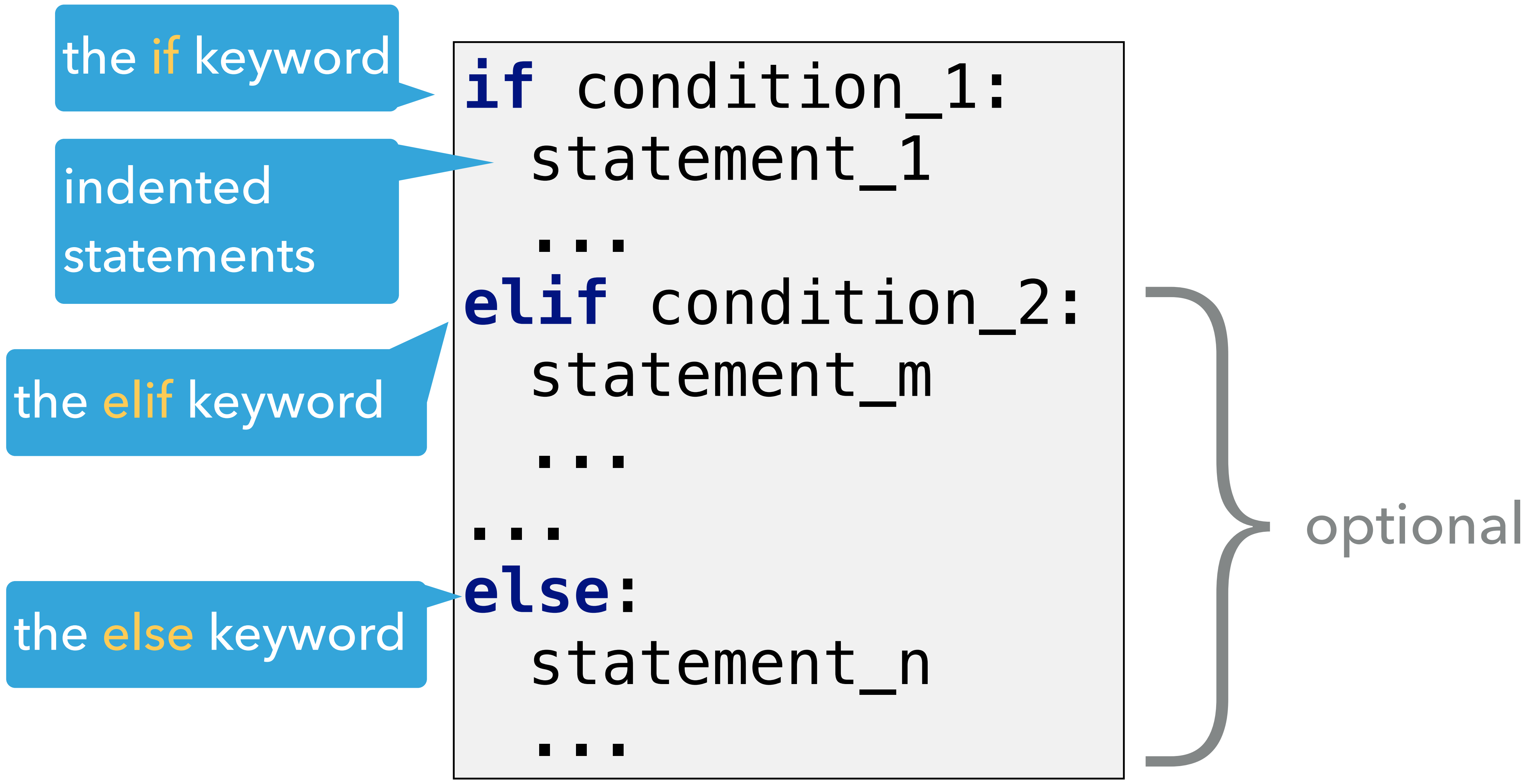
SELECTION / 选择结构

- Execute following statements only if a certain condition is met
- e.g.

```
import math
a, b, c = 1, 2, 3
delta = b ** 2 - 4 * a * c
if delta >= 0:
  x1 = (-b + math.sqrt(delta)) / (2 * a)
  x2 = (-b - math.sqrt(delta)) / (2 * a)
  print(x1, x2)
else:
  print('Error: delta < 0')
```

SELECTION / 选择结构

- The if statement in detail:



```
if condition_1:
  statement_1
  ...
elif condition_2:
  statement_m
  ...
...
else:
  statement_n
  ...
```

SELECTION / 选择结构

- A condition in the if statement is presented as an expression which evaluates to either True or False (Boolean Values/布尔值)
- Guess the value of the following Boolean expressions:

(1)	`1 != 0`	T	(6) `[0, 1] or []`	T
(2)	`2+3 == 5`	T	(7) `”Zhujieying”`	T
(3)	`2.333`	T	(8) `””`	F
(4)	`0`	F	(9) `None`	F
(5)	`True and False`	F	(10) `(9%3) ^ (not 0)`	T

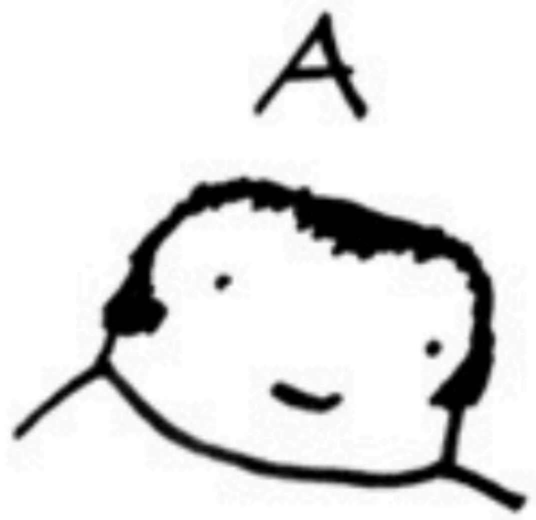
SELECTION / 选择结构

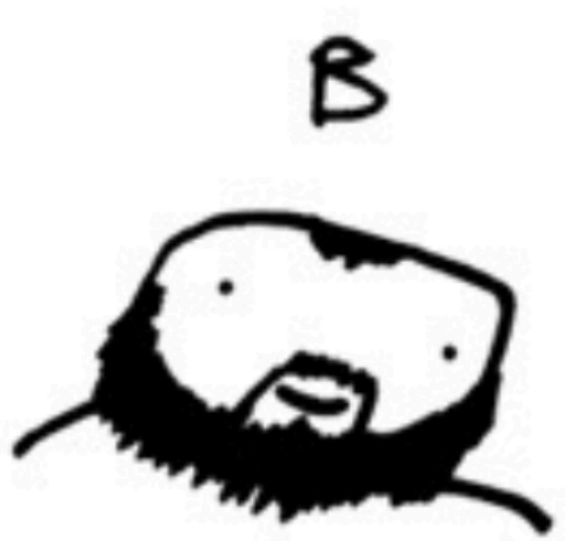
- Summary:
  - `and, or, not`: self-explanatory
  - any non-zero number, any non-empty list, or any non-empty string evaluates to True
  - 0 (also 0.0), empty list, or empty string evaluates to False
  - `==, !=, >, <, >=, <=:` self-explanatory
    `%:` remainder / 取余数
    N.B. == is not the same as = !!!
  - ^: XOR/异或(相同为False，不同为True)

SELECTION / 选择结构

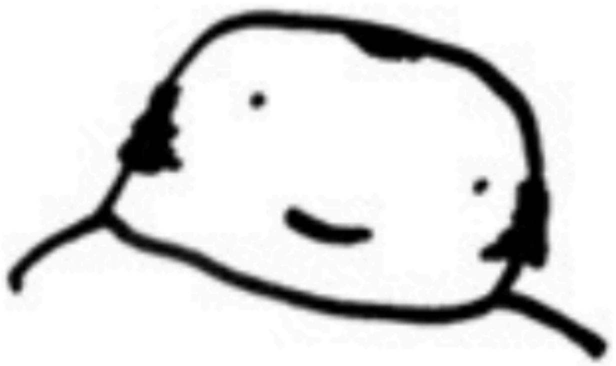
- Your turn:

Given A, B

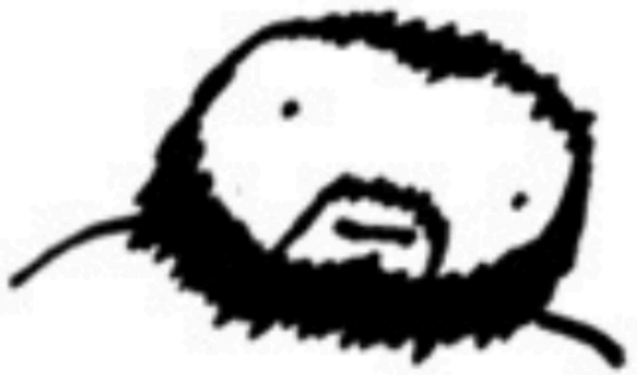




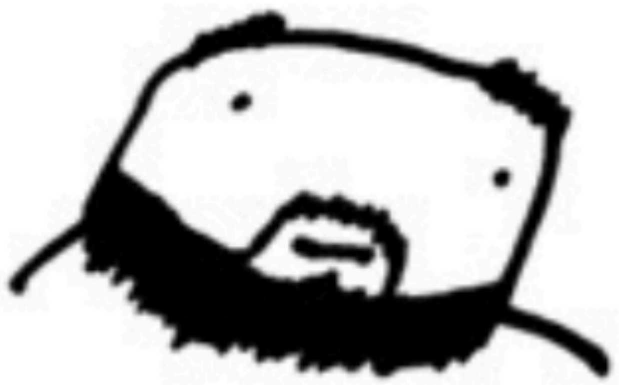
Write a Boolean expression for each of the following:



A and B



A or B



A ^ B

LOOP / 循环结构

- Two types: `for` and `while`
- e.g.

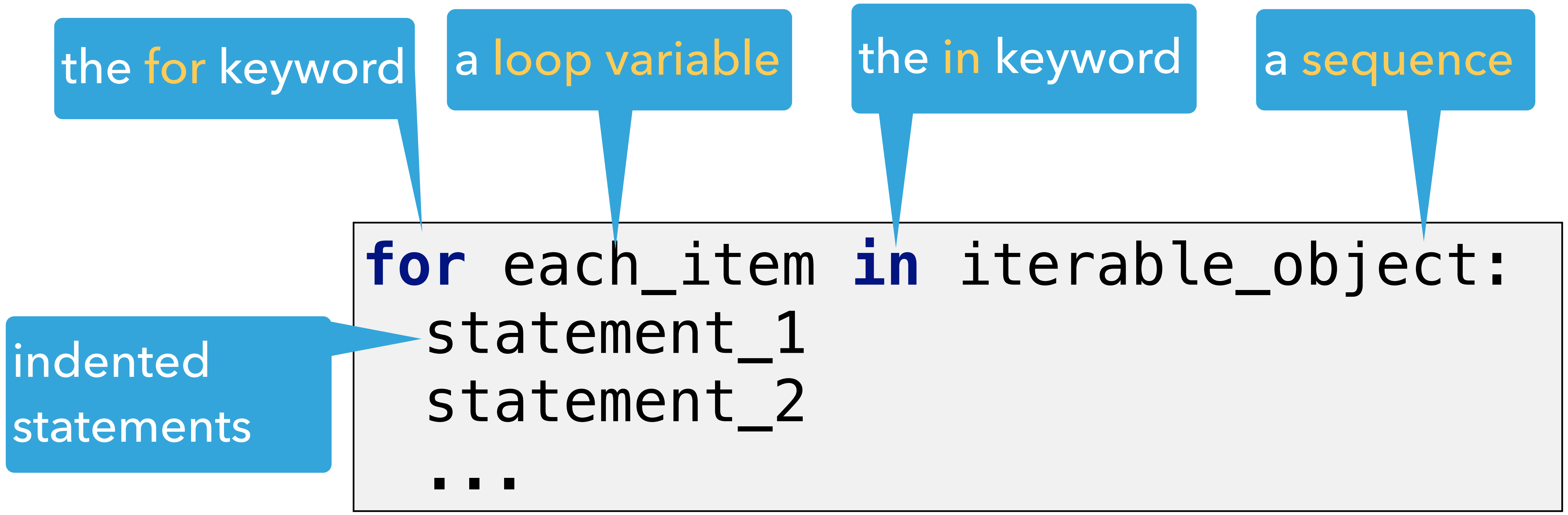
```
for i in range(5):
  print(i)
```

does the same thing as

```
i = 0
while i < 5:
  print(i)
  i += 1
```

LOOP / 循环结构

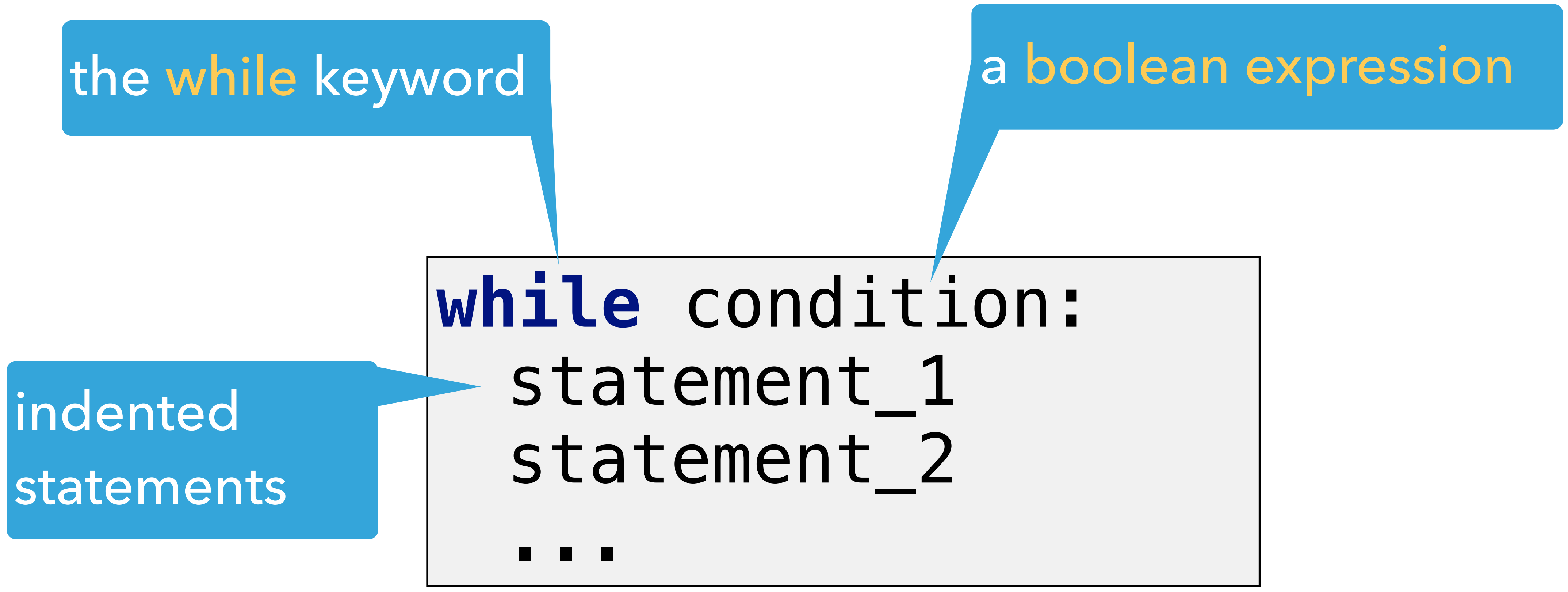
- The for loop in detail:



```
for each_item in iterable_object:
  statement_1
  statement_2
  ...
```

LOOP / 循环结构

- The while loop in detail:



```
while condition:
  statement_1
  statement_2
  ...
```

LOOP / 循环结构

- break and continue statements can alter the flow of a normal loop
    - break: exit the loop containing it, i.e. execute from the line immediately after the body of the loop
    - continue: skip the rest of the code inside a loop for the current iteration only, i.e. loop does not terminate but continues on with the next iteration
- Try:
  I want to print the first number in [0,10) that's divisible by 3, should I write break or continue here?

```
for n in range(10):
  if n % 3 == 0:
    print(n)
    break________
```

- What if I write continue instead?

LOOP / 循环结构

- A compact way to write for loops:

```
list_of_sqs = []
for i in range(5):
  list_of_sqs.append(i**2)
```

does the same thing as

```
list_of_sqs = [i**2 for i in range(5)]
```

LOOP / 循环结构

- Nested loops: a loop within a loop
    - e.g. Create a 5*5 null matrix (represented as a list of lists)

```
matrix = []
for i in range(5):
  matrix.append([])
  for j in range(5):
    matrix[i].append(0)
```

- Can you write it in the compact way (using just 1 line of code)?

```
matrix = [[0 for i in range(5)] for j in range(5)]
```

- Note: In practice, we can also use the `*` operator or the `numpy` package to do this more elegantly, without using `for` loops.
  – reminder from a student (not sure about his name, sorry)

PRACTICE

Try on your own, and we'll ask you about it next week!

- Find what's wrong with the following function:

```python
def find_age_group(age):
  if age >= 6:
    return 'teenager'
  elif age >= 18:
    return 'adult'
  else:
    return 'kid'
```

PRACTICE

- Consider the following list:

```
strs = ['3.14', '-24.2', '53', '3.8e10',
‘4,530.00', ‘1024p’]
```

Complete the function to take it as input, and return a filtered list of decimal numbers, using regular expression:

```
import re
def filter(strs):
  filtered_strs = []
  for ……
    if ……
      ……
  return filtered_strs
```

Hint: Only ‘1024p’ isn’t a decimal number.

PRACTICE

- Following the pseudocode in Figure 2.15, J&M 2.5, complete the function for computing minimum edit distance:

```
def min_edit_distance(source, target):
    ......
    return ......
```

Hint:

- Use the line of code to create a matrix in the last slide.
- Use a cost of 1 for all operations.
- Do NOT use recursion.

THAT’S IT!
CONGRATS!