Programming for Data Science: Iterations

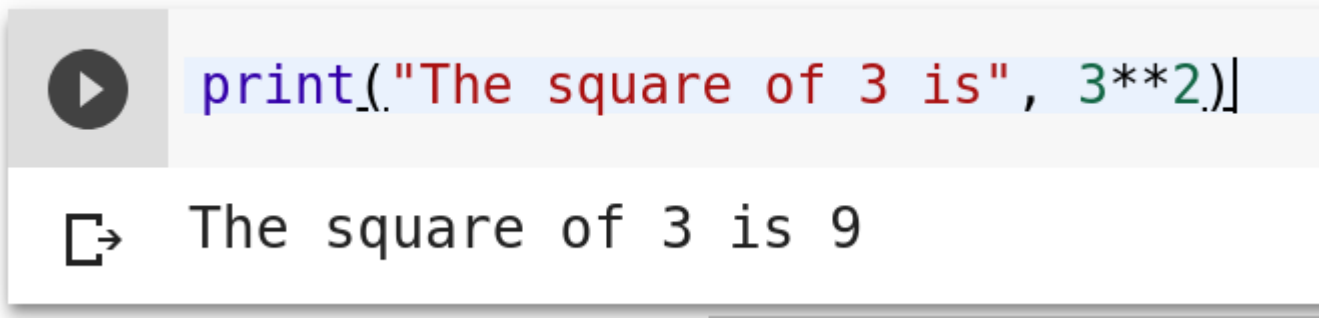
Instructor: Jakramate Bootkrajang

Outlines

- Motivation
- Counter-controlled loop
  - For loop
- Condition-controlled loop
  - While loop
- Break and continue statements

Motivation

- Suppose we want to calculate and print the square of 3
- How do we do that ?



Motivation [2]

- What if we want to do the same for 4 and 5 ?

```
print("The square of 3 is", 3**2)
print("The square of 4 is", 4**2)
print("The square of 5 is", 5**2)
```

```
The square of 3 is 9
The square of 4 is 16
The square of 5 is 25
```

Motivation [3]

- How about from 3 to 100 ?

Motivation [4]

- It is cumbersome to write the print statements repeatedly
- In fact, the statements are almost identical with few differences

```
print("The square of 3 is", 3**2)
print("The square of 4 is", 4**2)
print("The square of 5 is", 5**2)
```

```
The square of 3 is 9
The square of 4 is 16
The square of 5 is 25
```

Motivation [5]

- In many cases, we find ourself repeating the some operations multiple times
- Python provides program structure called iteration or loop that performs the same operations for a number of times
- There are two types of iterations
  - Counter-controlled iteration
  - Condition-controlled iteration

Counter-controlled loop

```
for item in Iterator:
    statement0
    statement1
    statement2
```

- iterator: data object that can be iterated
  - String, List, Generator
- item: loop variable, draws item from iterator one at a time, and takes its value
- The statements in the loops are repeated until no item can be drawn from iterator

Range function

- Widely used iterator that generates sequence of numbers
- Usage: range(start:stop:step)
  - Start: the beginning of sequence
    - optional [default == 0]
  - Step: the increment of number
    - optional [default == 1]
  - Stop: the endpoint of sequence (not included in the sequence)

Examples

- range(5) → 0,1,2,3,4
- range(1,10) → 1,2,3,4,5,6,7,8,9
- range(1,10,2) → 1,3,5,7,9
- range(10,1,-2) → 10,8,6,4,2

For loop and range()

- range() can be used to write for loop with predefined number of repetitions

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

```
0
1
2
3
4
```

- ‘i’ takes value in number seq generated by range()

Visualising loop and Caution

- First, see external material
- The loop body must be indented appropriately (4 whitespaces)

Condition-controlled loop

- In counter-controlled loop, we know exactly how many iterations to perform
- This is not always possible in real-world
  - We might not know how many times to loop
- Instead, we can write loop which relies on some condition and execute the loop until the condition is False

The while loop

```
while boolean_expression:
    statement1
    statement2
        ...
    boolean_update
```

- Ingredients
  - bool_exp: continue the loop if True
  - statement(s): some block of codes
  - boolean_update: at least there is a line of code than change the truth value of boolean_expression at some point

The while loop

```
06 x = 3
07 ans = 0
08 itersLeft = x
09
10 while (itersLeft != 0):        # Square an integer, the hard way
11     ans = ans + x
12     itersLeft = itersLeft - 1
13
14 print(str(x) + '*' + str(x) + ' = ' + str(ans))
```

- เราสามารถจำลองการ run ของ loop ได้โดยการเขียน

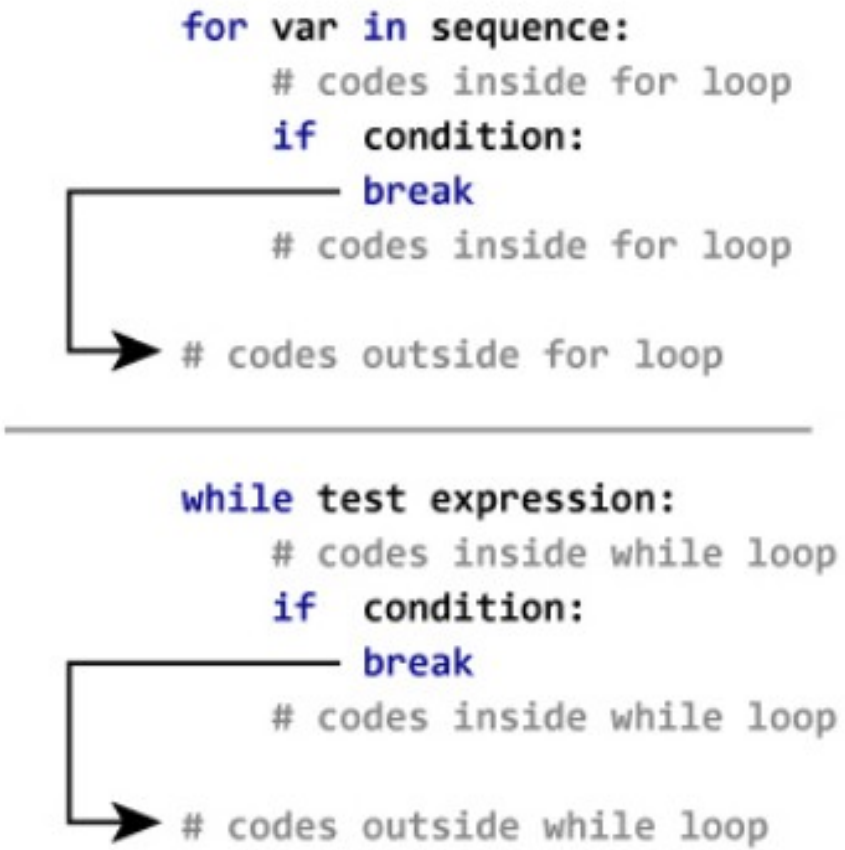
test#	x	ans	itersLeft
1	3	0	3
2	3	3	2
3	_____	_____	_____
4	_____	_____	_____

Infinite loop

- Be careful that your boolean expression does turn to False during your loop execution.
- Otherwise you’ll stuck in the loop forever

Break statement

- We can use break to prematurely left the loop



Infinite loop and break

- Seeing how break can be used to exit loop
- We can write an infinite loop and if statement to check when to exit the loop

```
while True:
  s = input(">")
  if len(s) == 0:
    break
  else:
    print(s)
```

```
>Hi
Hi
>How
How
>dy?
dy?
>
```

Continue statement

- To execute the next iteration without executing the remaining part of the loop

```
for i in range(5):
    print(i)
    continue
    print(i**2)
```

```
0
1
2
3
4
```

Continue statement [2]

```
n = 5
while n > 0:
  print(i)
  n = n - 1
  continue
  print(i**2)
  # don't put n=n-1 here!!!
```

```
0
1
2
3
4
```

Exercise 1

- Write a program to calculate the sum of even numbers from 2 to 100

Exercise 2

- Write a program which repeatedly reads numbers until the user enters "done". Once "done" is entered, print out the total, count, and average of the numbers. If the user enters anything other than a number, detect their mistake using try and except and print an error message and skip to the next number.