PYTHON LOOPS - WHILE, FOR, BREAK, CONTINUE, RANGE

CSC 109 - Introduction to programming in Python - Fall 2023

Marcus Birkenkrahe

December 2, 2023

Contents

1 Introduction **3**

2 While loops **3**

3 Annoying while it lasts **5**

4 Breaking out of a loop **6**

5 Continuing a loop **7**

6 Non-Boolean truth values **9**

7 For loops **10**

8 Equivalence of while and for **11**

9 Starting, stopping and stepping with `range` **12**

10 Summary **13**

11 Glossary **13**

12 References **14**

Figure 1: A Fokker looping (1915-20), Library of Congress@Flickr.com

1 Introduction

- Loops are clauses that repeat any number of times.
- Like all clauses, a condition is tested to check if the clause ought to be entered.
- But once it's been entered, the condition is tested again after performing some action:

```
loop [condition]:
   do something
   go back to the loop condition
```

- Python knows two loop types, `while` and `for`, and two loop exit strategies, `break` and `continue`.

2 While loops

- A `while` statement always consists of:
  1. The `while` keyword
  2. A condition (an expression that evaluates to `True` or `False`)
  3. A colon (`:`)
  4. An indented block (`while` clause) on the next line.
- At the end of an `if` statement, execution continues. At the end of a `while` statement, execution jumps back to test the condition again.
- Here is an `if` statement to print 'Hello, world.'

```
spam = 0
if spam < 5:
    print('Hello, world.')
    spam = spam + 1
```

```
Hello, world.
```

- Replacing `if` by `while`: when executed, this prints 'Hello, world.' five times.

```
spam = 0
while spam < 5:
    print('Hello, world.')
    spam = spam + 1

Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
```

- Look at the BPMN process diagrams and the code side by side (slide):

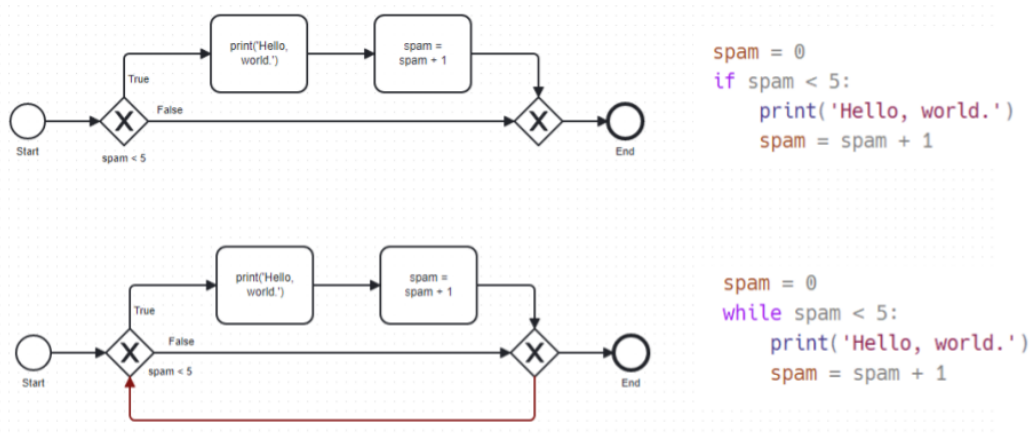

Figure 2: if vs. while statement

- In the `while` statement, the condition is checked at the start of each *iteration* (that is one loop execution).
- Production BPMN diagrams with loops do not have any lines running back: instead, the loop tasks are overloaded as shown here. To save time, we're not using a production BPMN tool (like signavio.com) but a simpler, free online tool like bpmn.io.

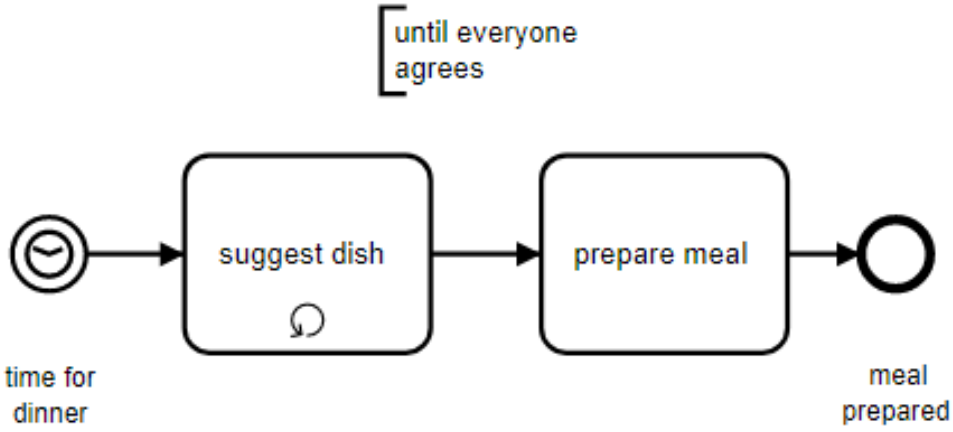

Figure 3: Loop task in BPMN (camunda.com)

3 Annoying while it lasts

- Create a new code block (or a new notebook) in Colab and enter the following code, replacing `'your name'` by your actual name:
  - collapse the print + input lines into one input + prompt line!

```
name = ''
while name != 'your name':
    print('Please type your name')
    name = input()
print('Thank you!')
```

- Run the file and try to understand what's going on:
  1. The `name` variable is initialized as the empty string `''`
  2. The condition is tested for the first time: it's `True`
  3. The clause is entered and you're asked to enter your name
  4. The `input` function assigns what you entered to `name`
  5. The condition is tested again (and again and again...)
  6. As soon as you enter your name, with the correct spelling, the condition becomes `False` and the program jumps to the last line.

- Go to autbor.com/yourname/ to run the file. `Edit this code` to change `'your name'` in the code to your own name.
- Here is the process diagram for this code:

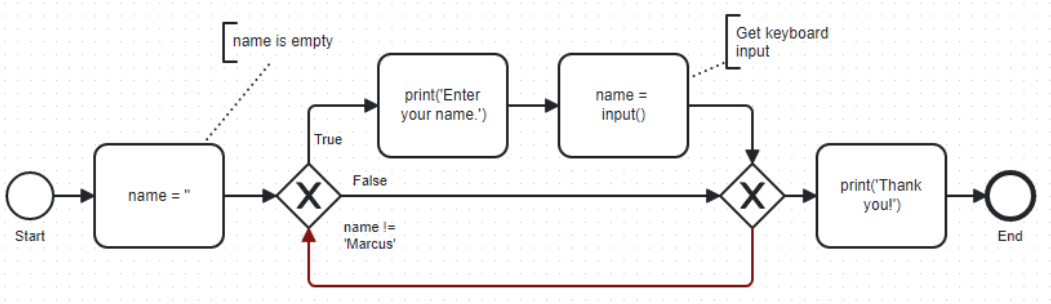

Figure 4: While loop that waits until you enter your name

4 Breaking out of a loop

- Loops are entered when the condition evaluates to `True` and they are left only if it evaluates to `False` - it's easy to go "infinite".
- When execution hits the `break` statement, it exits the current clause immediately. In nested loops, it exits the innermost loop only.
- Let's create an infinite loop and `break` it when a condition is met:

```
while True:
    name = input('Please type your name: ')
    if name == 'your name':
        break
print('Thank you!')
```

- Go to autbor.com/yourname2/ to run the file. `Edit this code` to change `'your name'` in the code to your own name.
- Here is the process diagram for this and the previous loop:
- **Exercise:** what happens if you use `break` outside of a loop clause? Can you fool Python by indenting the `break`? Write a one-line "Hello, world!" program followed by a `break` statement:

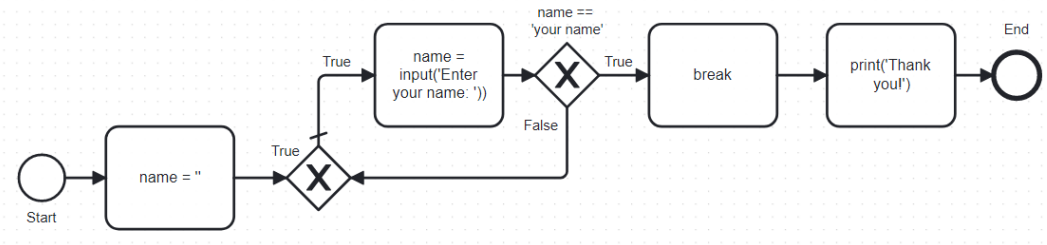

Figure 5: Infinite while loop that must be broken out of

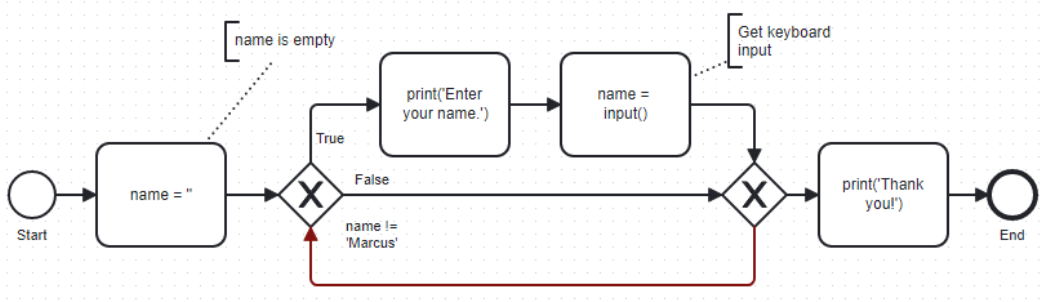

Figure 6: Infinite while loop that must be broken out of

```
print("Hello, world!")
break
```

- For fun, create an infinite loop *without* break condition that prints "Infinity!" forever until you break the execution with CTRL-D:

```
while True:
    print('Infinity')
```

- In Colab, delete the "infinite" output by clicking on the "X".

5 Continuing a loop

- Like `break`, the `continue` statement is only used inside a loop.
- When a `continue` statement is reached, the program jumps back to the start of the loop and re-evaluates the loop condition.
- Write a program that:

1. starts with an infinite `while` loop (always `True`)
2. asks for `input` of a `name`.
3. `if` the `name` is not equal to `'Joe'` it executes `continue`
4. otherwise, it asks for `input` of the `password`
5. `if` the `password` is `'swordfish'` it executes `break`.
6. confirms `'Access granted'` when you're done.

- Solution:

```
while True:
    name = input("Who are you? ")
    if name != 'Joe':
        continue
    password = input("What's the password? ")
    if password == 'swordfish':
        break
print('Access granted')
```

- Copy the code, open pythontutor.com and paste the program code.
- Run the program inside the pythontutor.com visualization tool.
- The BPMN diagram shows the two break points clearly:

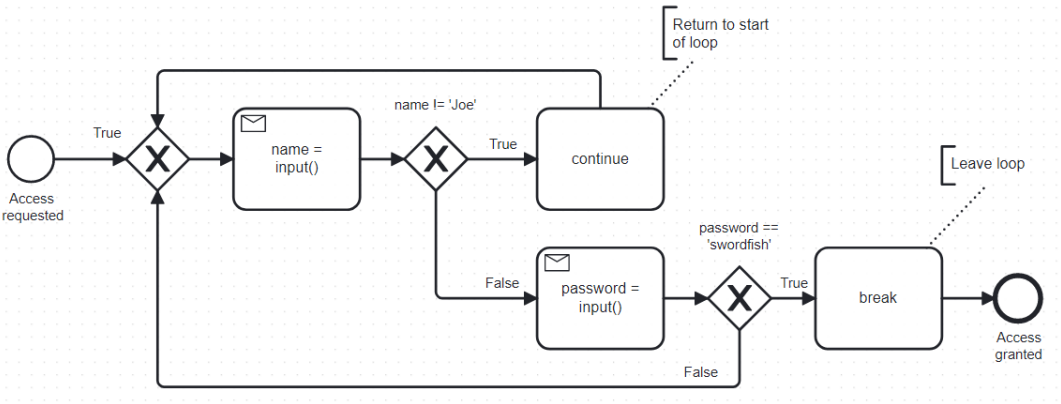

Figure 7: Continue and break with an infinite loop

- Solution (pythontutor.com)

6 Non-Boolean truth values

- Recall: to the computer, when used in conditions, `0`, `0.0` and `''` (empty string) are `False`, while all other values are `True`.
- What does the following program do?

```
name = ''
while not name:    # until you enter a name!
    name = input('Enter your name: ')
    guests = input('How many guests will you have? ')
if int(guests):    # if you have non-zero guests
    print('Make sure to have enough room')
print('Done')
```

- Try it in pythontutor: autbor.com/howmanyguests/
- You could have entered `while not name != ''` instead of `while not name` and you could have used `if guests != 0` instead of `if guests`
- The BPMN diagram:

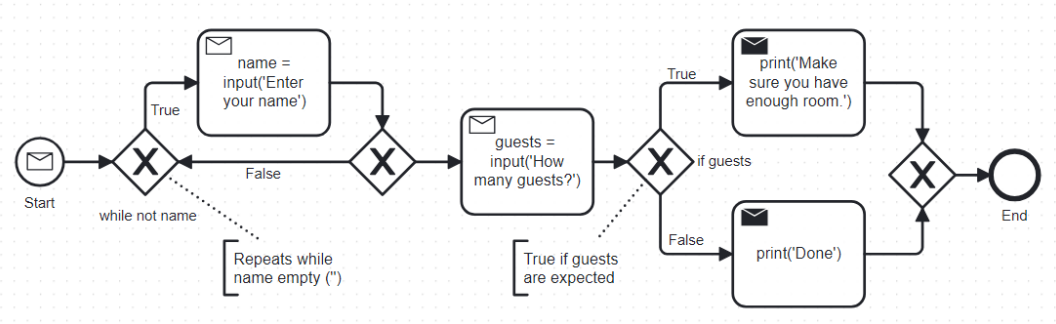

Figure 8: Non-Boolean truth values

- If you enter `''` in response to the second question (number of `guests`), an error is generated, because `int` cannot convert empty space or whitespace to an integer. To fix this, you must handle it as an *exception*:

```
name, guests = '','' 
while not name:
```

```
    name = input('Enter your name: ')
    guests = input('How many guests will you have? ')
if guests == '':
    guests = 0
    print('okay')
if int(guests):
    print('Make sure to have enough room')
    print('Done')
```

7 For loops

- The `for` statement allows you to repeat a block of code a certain number of times.
- A `for` statement includes:
  1. the `for` keyword
  2. a loop variable
  3. a call to the `range` function with up to 3 integers
  4. a colon `:`
  5. An indented clause starting on the next line
- Simple example: the program `fiveTimes.py` executes the statement in its clause five times while `i` is counting up from 0 to 4:

```
print('My name is')
for i in range(5):
    print('Jimmy Five Times (' + str(i) + ')')
```

```
My name is
Jimmy Five Times (0)
Jimmy Five Times (1)
Jimmy Five Times (2)
Jimmy Five Times (3)
Jimmy Five Times (4)
```

- Challenge: how would the `print` statement look like with an f-string?
- Solution:

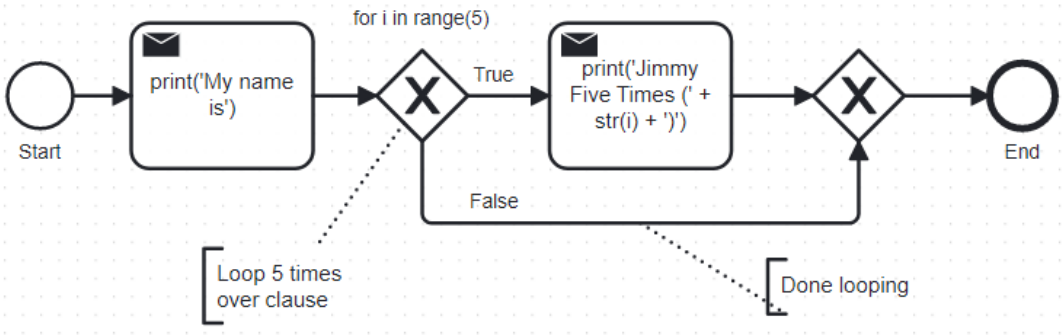

Figure 9: For loop with range 5

```
print('My name is')
for i in range(5):
    print(f'Jimmy Five Times ({i})')
```

- `for` loops are great for counting up and down in regular in- or decrements. This program adds up all numbers from 1 to 100 and stores the result in `total`:

```
total = 0
for i in range(101):
    total = total + i
print(total)
```

```
5050
```

- This last result relates to a story: when the mathematician Carl Friedrich Gauss was a boy, he found a way to add up all the numbers from 0 to 100. He noticed that there are 50 pairs of numbers that add up to 101: 1 + 100, 2 + 99,. . . ,50 + 51, and 50 * 101 = 5,050.
- Check this program in pythontutor.com.

8 Equivalence of while and for

- `for` loops and `while` loops are fully equivalent but the former are more concise than the latter.

- **Exercise**: rewrite `fiveTimes.py` as `fiveTimes2.py` with a `while` loop instead of a `for` loop.
- Solution:

```
print('My name is')
i = 0
while i < 5:
    print('Jimmy Five Times (' + str(i) + ')')
    i = i + 1
```

- Once you're done, run the program at pythontutor.com.

9 Starting, stopping and stepping with `range`

- The shortest documentation can be had on the IPython shell with the keyword (variable or function) `?`
- The `range` function ('constructor') e.g. is documented online in the Python standard library:
  1. All parameters must be integers only, keywords are not allowed
  2. Only the `stop` parameter is mandatory: `range(5)`
  3. The other paramaters: `range(start, stop[, step])` with defaults `start=0` and `step=1`.
- For example, `range(12,16)` starts at 12 and stops at 16:

```
for i in range(12,16):
    print(i)

12
13
14
15
```

- Counting up from 2 in steps of 2:

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

```
2
4
6
8
```

- You can use a negative number for `step` to make the loop count down:

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

```
5
4
3
2
1
0
```

- What `type` is `range(5)`?

```
print(type(range(5))
```

10 Summary

- Code can be executed repeatedly in a loop while their conditions evaluate to `True` using `while` or `for`.
- The `range` function constructs a sequence of integers. Its parameters are `start`, `stop` and `step` values, with default `start=0`, `step=1`.
- `break`, `continue` and `sys.exit` can exit a loop, jump back to the start, or terminate the execution.

11 Glossary

TERM/COMMAND	MEANING
`while`	Conditional loop
`for`	Conditional loop with counter
`range`	Create sequence of integers
`break`	Exit loop
`continue`	Go to start of loop

12 References

- Sweigart, A. (2019). Automate the Boring Stuff with Python. NoStarch. URL: automatetheboringstuff.com
- Yunits, B. (2019). Which programming languages use indentation? URL: pldb.com.