

Introduction to Python Day 3

Verjina Metodieva and Daniel Parthier

2025-02-18

Index based *for loops* - `range()`

- generates integer sequences
- `range(n)` generates the series of n values from 0 to $n - 1$

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

```
0  
1  
2  
3  
4
```

```
# looping through data indices. find the max  
B = [1, 4, 6, 7, 89, 54]  
big_indx = 0  
for i in range(len(B)):  
    if B[i] > B[big_indx]:  
        big_indx = i  
print('The max value in B is', B[big_indx], 'found on position', big_indx)
```

The max value in B is 89 found on position 4

Index based *for loops* - `enumerate()`

- assigns a count to each item within an iterable and returns it as an enumerate object
- one way to avoid nested loops

```
import numpy as np

array_a = np.arange(20, 25)
for indx, val in enumerate(array_a):
    print('the index is', indx)
    print('the value is', val)
```

```
the index is 0
the value is 20
the index is 1
the value is 21
the index is 2
the value is 22
the index is 3
the value is 23
the index is 4
the value is 24
```

! range() and enumerate() - none of the two returns a list of objects!

-
- motivation: limitation in 'simple' for loops - we don't know the position of an element within a sequence, as we experienced in the last example
 - range(n) generates the series of n values from 0 to $n - 1$
 - precisely the series of valid indices into a sequence of length n
 - range() - returns a range object that is iterable
 - enumerate() - returns an enumerate object that is also iterable
 - they are mainly used in loops

Break and continue statements

- break - immediately terminates the loop
- continue - skips whatever is after it and continues with the next iteration
 - mostly used after a conditional statement

-
- a break statement that immediately terminates a while or for loop when executed within its body
 - a continue statement - skips the rest of the statements in the current iteration of the loop and it returns control to the beginning of the loop

While loops

- Perform a task **while** something is **True**
- Be careful:
 - Some loops never finish (get stuck)
 - Make sure that condition for ending the loop can be fulfilled

```
while check_condition:  
    perform_task()
```

If your python terminal gets stuck at one point you can try a `KeyboardInterrupt` using `Ctrl+C`, which will kill the running script.

Let's wait while we wait

- Start a little counter

```
import time  
counter = 0  
while counter < 10:  
    time.sleep(1)  
    counter += 1  
    print("You waited for " + str(counter) + " seconds...")
```

- Good for keeping processes running

Try to avoid **while** loops as much as possible. They can be useful if you do not have information how long it should run, but know it will at one point finish.

Errors and how to read them

There are useful resources regarding errors

- Simply googling works surprisingly well
- You will often end up on [stackoverflow](#)
 - There is no question which was not already asked¹

¹if that is not true open up a question

stackoverflow About Products OverflowAI Log in Sign up

Home Questions Tags Users Companies LABS Jobs Discussions COLLECTIVES TEAMS

Search Results

Results for if clause python
Search options not deleted

500 results

Relevance Newest More

4281 votes **Manually raising (throwing) an exception in Python**
Problem 1: Hiding bugs raise Exception("I know Python!") # Don't! If you catch, likely to hide bugs. ... Best Practices: except **clause** When inside an except **clause**, you might want to, for example, log th...
python exception Aaron Hall 395k answered Jun 5, 2014 at 16:30

778 votes **Why does python use 'else' after for and while loops?**
I'm wondering how Python coders read this construct in their head (or aloud, if you like). Perhaps I'm missing something that would make such code blocks more easily decipherable? ... See also Else...
python if-statement for-loop for-else Kent Boogaart 179k asked Apr 2, 2012 at 16:18

2200 votes **What does if __name__ == "__main__": do?**
When your script is run by passing it as a command to the Python interpreter, python myscript.py all of the code that is at indentation level 0 gets executed. ... If your script is being imported into another...
python namespaces program-entry-point python-module idioms Adam Rosenfield 400k answered Jan 7, 2009 at 4:28

-1 votes **Python Dictionary if clause [duplicate]**
as you can tell, I just recently started to learn python. While practising, I stumbled upon an issue, for which I don't have an answer to why this is the case. ... Saigon are not in the dictionary Python no...
python if-statement david0711 3 asked Jul 27, 2021 at 15:58

248 votes **How to execute multi-line statements within Python's own debugger (PDB)**
Now, when I get to the debugger I want to execute a multi-line statement such as an **if clause** or a for loop but as soon as I type if condition: and hit the return key, I get the error message *** SyntaxError ...
python debugging multiline pdb Mike 2,481 asked May 11, 2011 at 16:01

-1 votes **executing wrong if-clause in python [duplicate]**

Hot Network Questions

- What can a bear superhero use as a projectile?
- Looking for an old fantasy book about dragons. Small ones are kept as pets but others are killed. Main character is from an underground society
- Why can pressure be identified as partial of energy with respect to volume?
- How can the Director of National Intelligence be unaware of IMF?
- mks.ext4 to loop: 128-byte inodes cannot handle dates beyond 2038 and are deprecated
- Is it appropriate to abbreviate authors' names in function names, even with proper attribution?
- Is 13 minutes enough time to change platforms in

Types of errors

1. SyntaxErrors
2. IndentationError
3. NameError
4. TypeError
5. IndexError
6. AttributeError
7. etc.

Fix errors

- Breath
- Don't panic
 - Identify the error by checking the terminal output

- Look at the line provided
- Go backwards if error is nested