The equivalent of a GOTO in python [duplicate]

This question already has an answer here:
Is there a label/goto in Python? *10 answers*

I am self teaching myself python 2.7. I have some experience in using BATCH, which has a GOTO statement. How do I do that in python? For example, suppose I want to jump from line 5 to line 18.

I realize there have been previous questions regarding this topic, but I have not found them sufficiently informative or, are too high level in python for my current understanding.

python goto

edited Sep 18 '13 at 8:28
csgillespie
29.4k ●8 ●61 ●104

asked Sep 18 '13 at 3:21
Calder Hutchins
157 ●1 ●1 ●8

marked as duplicate by kojiro, mgilson, tpg2114, sebastian-c, csgillespie Sep 18 '13 at 8:26

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

7 @Makoto, because he learned using batch and doesn't know any better. – OmnipotentEntity Sep 18 '13 at 3:23

1 @kojiro Please read the whole topic. – Calder Hutchins Sep 18 '13 at 3:28

"One of thme mentioned a function, but i don't see how they would work." Understand how functions work, then you will see that you don't need `goto` . – Akavall Sep 18 '13 at 3:28

ok, I'll go do that Akavall – Calder Hutchins Sep 18 '13 at 3:30

4 Random: If you really do want to use a goto, somebody actually wrote a working version as part of an April Fool's joke in 2004. By no means do I recommend actually using it, but I thought I'd link it for amusement's sake. – Michael0x2a Sep 18 '13 at 3:51

6 Answers

`Goto` s are universally reviled in computer science and programming as they lead to very unstructured code.

Python (like almost every programming language today) supports structured programming which controls flow using if/then/else, loop and subroutines.

The key to thinking in a structured way is to understand how and why you are branching on code.

For example, lets pretend Python had a `goto` and corresponding `label` statement *shudder*. Look at the following code. In it if a number is greater than or equal to 0 we print if it

```
number = input()
if number < 0: goto negative
if number % 2 == 0:
   print "even"
else:
   print "odd"
goto end
label: negative
print "negative"
label: end
print "all done"
```

If we want to know when a piece of code is executed, we need to carefully traceback in the program, and examine how a label was arrived at - which is something that can't really be done.

For example, we can rewrite the above as:

```
number = input()
goto check

label: negative
print "negative"
goto end

label: check
if number < 0: goto negative
if number % 2 == 0:
   print "even"
else:
   print "odd"
goto end

label: end
print "all done"
```

Here, there are two possible ways to arrive at the "end", and we can't know which one was chosen. As programs get large this kind of problem gets worse and results in spaghetti code

In comparison, below is how you *would* write this program in Python:

```
number = input()
if number >= 0:
   if number % 2 == 0:
       print "even"
   else:
       print "odd"
else:
   print "negative"
print "all done"
```

I can look at a particular line of code, and know under what conditions it is met by tracing back the tree of `if/then/else` blocks it is in. For example, I know that the line `print "odd"` will be run when a `((number >= 0) == True) and ((number % 2 == 0) == False)`.

edited Sep 18 '13 at 6:36

answered Sep 18 '13 at 3:35
Lego Stormtroopr
16k ●7 ●34 ●63

5 thank-you for the effort. I believe that helped me greatly. – Calder Hutchins Sep 18 '13 at 3:39

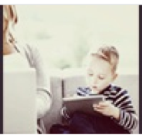
@kojiro Thanks! Fixed. – Lego Stormtroopr Sep 18 '13 at 3:39

2 +1, Your explanation was great, and all of the code looks good, but as the scripts get bigger and bigger, you may not be able to deal with a negative in a single line. if/else begins piling up with indents off the screen. It may be better then if in this example you take the input, call `check()`, and in `check()`, call `negative()` if we've found the number to be negative. Furthermore, this would show how easy it is to take `label`s and go straight to functions (which our OP doesn't seem to entirely grasp) – scohe001 Sep 18 '13 at 3:43

1 @LegoStormtroopr I understand, that's why I'm trying to tell you to teach the OP about using functions to avoid that now in a simple example like this before you send another one of *those* programmers out into the fray (as I said before, he doesn't seem to have very much of a grasp on functions) – scohe001 Sep 18 '13 at 3:49

1 “`goto`s are universally reviled in computer science and programming”—not quite. `goto` is perfect for error handling in C (eschewing RAII and exceptions), and for implementing structured programming constructs. Absolute statements are always—well, often—misguided. – Jon Purdy Sep 18 '13 at 7:16

Work on work you love. **From home.**



stackoverflow

Forgive me - I couldn't resist ;-)

```
def goto(linenum):
    global line
    line = linenum

line = 1
while True:
    if line == 1:
        response = raw_input("yes or no? ")
        if response == "yes":
            goto(2)
        elif response == "no":
            goto(3)
        else:
            goto(100)
    elif line == 2:
        print "Thank you for the yes!"
        goto(20)
    elif line == 3:
        print "Thank you for the no!"
        goto(20)
    elif line == 20:
        break
    elif line == 100:
        print "You're annoying me - answer the question!"
        goto(1)
```

answered Sep 18 '13 at 3:45
Tim Peters
22.7k ●4 ●35 ●53

1 Hmm interesting, +1 – Calder Hutchins Sep 18 '13 at 3:47

22 LOL! Calder, it's a joke. It's possible to write horrible code in **any** language ;-) – Tim Peters Sep 18 '13 at 3:47

5 PLEASE DO IMPLEMENT COME FROM NEXT? – msw Sep 18 '13 at 3:48

1 lol tim i know it was a joke :p – Calder Hutchins Sep 18 '13 at 4:06

1 Wouldn't it be more pythonic to keep a dict of line numbers to functions? Like this maybe. Or would it be better to make `end` raise something? – abarnert Sep 24 '13 at 0:15

I entirely agree that `goto` is poor poor coding, but no one has actually answered the question. There **is** in fact a goto module for Python (though it was released as an April fool joke and is not recommended to be used, it *does* work).

edited Jul 20 '15 at 17:50

answered Sep 18 '13 at 3:53
scohe001
2,665 ●9 ●29

as of Python 2.6.9 it does not work anymore... – fortran Aug 12 '14 at 15:34

Continuation passing style requires goto (in the guise of function calls). – Erik Haliewicz Feb 11 '15 at 20:56

There's no `goto` instruction in the Python programming language. You'll have to write your code in a structured way... But really, why do you want to use a `goto`? that's been considered harmful for decades, and any program you can think of can be written without using `goto`.

Of course, there are some cases where an unconditional jump might be *useful*, but it's never *mandatory*, there will always exist a semantically equivalent, structured solution that doesn't need `goto`.

edited Sep 18 '13 at 3:28

answered Sep 18 '13 at 3:23
Óscar López
130k ●14 ●145 ●222

Thank-you, I wanted to use it to: python askes the user a question and if the raw input is not yes or no it will ask another question about what the user meant. Then it would go back to the line before the question was asked. – Calder Hutchins Sep 18 '13 at 3:27

@CalderHutchins Can you explain what you mean by if it's not yes or not it asks another question? – C0deH4cker Sep 18 '13 at 3:29

@CalderHutchins: That's trivial to do in a structured flow. There's no need to create any jumps to arbitrary lines. – Makoto Sep 18 '13 at 3:30

1 @CalderHutchins you don't need a `goto` for that, put the question inside a loop, if the answer is not valid then loop again until the input is right – Óscar López Sep 18 '13 at 3:30

I mean if you do not asnswer the question with yes or no, python will ask you if the answer you gave is similar to yes, or to no, basically i want it to learn by itself, of which i will have to find out how to make python edit it's own code, like BATCH can. – Calder Hutchins Sep 18 '13 at 3:32

Disclaimer: I have been exposed to a significant amount of F77

The modern equivalent of `goto` (arguable, only my opinion, etc) is explicit exception handling:

Edited to highlight the code reuse better.

Pretend pseudocode in a fake python-like language with `goto`:

```
def myfunc1(x)
    if x == 0:
        goto LABEL1
    return 1/x

def myfunc2(z)
    if z == 0:
        goto LABEL1
    return 1/z

myfunc1(0)
myfunc2(0)

:LABEL1
print 'Cannot divide by zero'.
```

Compared to python:

```
def myfunc1(x):
    return 1/x

def myfunc2(y):
    return 1/y


try:
    myfunc1(0)
    myfunc2(0)
except ZeroDivisionError:
    print 'Cannot divide by zero'
```

Explicit named exceptions are a *significantly* better way to deal with non-linear conditional branching.

edited Sep 18 '13 at 3:57

answered Sep 18 '13 at 3:39
cjrh
5,318 ● 16 ● 34

1 That's not a bad way of thinking about the *best possible* use of non-local gotos – msw Sep 18 '13 at 3:51

Truth: >95% of GOTOs I have seen in the wild have been used for preemptive error handling. The other 5% have been clever tricks that resulted in horrific bugs, and the code, as-is, couldn't even **be** translated to a different goto-less language without requiring changes to the algorithm. – cjrh Sep 18 '13 at 3:55

@crjh: I'm very interested in seeing codes with `goto` s that cannot be directly translated to `goto` -less language. Can you give me example? – justhalf Sep 18 '13 at 3:57

1 @justhalf I can't find the exact paper, but there is a proof that **all** programs that use `goto` can be rewritten to use while loops instead. Not cleanly mind you, but you can remove all `goto` s. – Lego Stormtroopr Sep 18 '13 at 4:19 ✎

1 @LegoStormtroopr: you should be addressing that to cjrh, haha But yeah, I guess the "not cleanly" part is what cjrh referred to as "requiring changes to the algorithm" – justhalf Sep 18 '13 at 4:24

```
answer = None
while True:
    answer = raw_input("Do you like pie?")
    if answer in ("yes", "no"): break
    print "That is not a yes or a no"
```

Would give you what you want with no goto statement.

answered Sep 18 '13 at 4:30
Paul Becotte
1,706 ● 6 ● 15

This is not a good idea. While it technically correct, it makes the logic very difficult to untangle. – Lego Stormtroopr Sep 18 '13 at 5:10

2 It's better than goto. – William Shipley Mar 12 '14 at 21:31