UnboundLocalError: local variable referenced before assignment when reading from file

I have also tried searching for the answer but I don't understand the answers to other people's similar problems...

```
tfile= open("/home/path/to/file",'r')

def temp_sky(lreq, breq):
    for line in tfile:
        data = line.split()
        if (    abs(float(data[0]) - lreq) <= 0.1
            and abs(float(data[1]) - breq) <= 0.1):
            T= data[2]
    return T
print temp_sky(60, 60)
print temp_sky(10, -10)
```

I get the following error

```
7.37052488
Traceback (most recent call last):
File "tsky.py", line 25, in <module>
  print temp_sky(10, -10)
File "tsky.py", line 22, in temp_sky
  return T
UnboundLocalError: local variable 'T' referenced before assignment
```

The first print statement works correctly but it won't work for the second. I have tried making T a global variable but this makes both answers the same! Please help!

python

edited Jan 19 at 1:44
icedwater
2,532 ● 3 ● 17 ● 39

asked Mar 12 '13 at 17:14
user1958508
84 ● 1 ● 1 ● 7

to get rid of UnboundLocalError, the if statement has to run, so try give T a default value so that T is defined, refer to @shx2 answer – PurityLake Mar 12 '13 at 17:20

1 Also, you are running an entire loop to get a single value. This way your loop will always return the last matching instance of data. You can make your code more efficient by reading it in reverse order and instead of assigning, `return T` – ferrix Mar 12 '13 at 17:28

6 Answers

Your `if` statement is always false and T gets initialized only if a condition is met, so the code doesn't reach the point where `T` gets a value (and by that, gets defined/bound). You should introduce the variable in a place that always gets executed.

Try:

```
def temp_sky(lreq, breq):
    T = <some_default_value> # None is often a good pick
    for line in tfile:
        data = line.split()
        if ( abs(float(data[0]) - lreq) <= 0.1 and abs(float(data[1]) - breq) <=
0.1):
            T= data[2]
    return T
```

edited Mar 12 '13 at 17:25
ferrix
517 ● 2 ● 8

answered Mar 12 '13 at 17:17
shx2
25.3k ● 4 ● 37 ● 69

Thanks! This has fixed the error message – user1958508 Mar 12 '13 at 18:31

The other answers are correct: You don't have a default value. However, you have another problem in your logic:

You read the same file twice. After reading it once, the cursor is at the end of the file. To solve this, you can do two things: Either open/close the file upon each function call:

```
def temp_sky(lreq, breq):
    with open("/home/path/to/file",'r') as tfile:
        # do your stuff
```

This hase the disadvantage of having to open the file each time. The better way would be:

```
tfile.seek(0)
```

You do this after your `for line in tfile:` loop. It resets the cursor to the beginning to the next call will start from there again.

answered Mar 12 '13 at 17:37

javex
3,154 ● 14 ● 36

After my error was resolved I was still not getting the right answer and this was due to the file cursor not being in the right place. I have amended this as you suggested and is now working. Thanks – user1958508 Mar 12 '13 at 18:32

Before I start, I'd like to note that I can't actually test this since your script reads data from a file that I don't have.

'T' is defined in a local scope for the declared function. In the first instance 'T' is assigned the value of 'data[2]' because the conditional statement above apparently evaluates to True. Since the second call to the function causes the 'UnboundLocalError' exception to occur, the local variable 'T' is getting set and the conditional assignment is never getting triggered.

Since you appear to want to return the first bit of data in the file that matches your conditonal statement, you might want to modify you function to look like this:

```
def temp_sky(lreq, breq):
    for line in tfile:
        data = line.split()
        if ( abs(float(data[0]) - lreq) <= 0.1 and abs(float(data[1]) - breq) <=
0.1):
            return data[2]
    return None
```

That way the desired value gets returned when it is found, and 'None' is returned when no matching data is found.

answered Mar 12 '13 at 17:28

brandonsimpkins
163 ● 1 ● 7

1 That is actually not identical behavior. The example given returns the last matching instance and you are returning the first instance. Reading the file from the end would work but a simple `reversed()` will not do if we don't want the entire file in memory. – ferrix Mar 12 '13 at 17:31

FWIW: I got the same error for a different reason. I post the answer here not for the benefit of the OP, but for the benefit of those who may end up on this page due to its title... who might have made the same mistake I did.

I was confused why I was getting "local variable referenced before assignment" because I was calling a FUNCTION that I knew was already defined:

```
def job_fn(job):
  return job + ".job"

def do_something():
  a = 1
  b = 2
  job_fn = job_fn("foo")

do_something()
```

This was giving:

```
UnboundLocalError: local variable 'job_fn' referenced before assignment
```

Took me a while to see my obvious problem: I used a local variable named `job_fn` which masked the ability to see the prior function definition for `job_fn`.

answered Nov 24 '14 at 19:33

Dan H

3,727 ● 2 ● 12 ● 22

Contributing to ferrix example,

```
class Battery():

def __init__(self, battery_size = 60):
        self.battery_size = battery_size
def get_range(self):
    if self.battery_size == 70:
        range = 240
    elif self.battery_size == 85:
        range = 270

    message = "This car can go approx " + str(range)
    message += "Fully charge"
    print(message)
```

My message will not execute, because none of my conditions are fulfill therefore receiving " UnboundLocalError: local variable 'range' referenced before assignment"

```
def get_range(self):
    if self.battery_size <= 70:
        range = 240
    elif self.battery_size >= 85:
        range = 270
```

answered Jan 19 at 1:23

Greg Vazquez

1 ● 1

I was facing same issue in my exercise. Although not related, yet might give some reference. I didn't get any error once I placed addition_result = 0 inside function. Hope it helps! Apologize if this answer is not in context.

```
user_input = input("Enter multiple values separated by comma > ")

def add_numbers(num_list):
    addition_result = 0
    for i in num_list:
        addition_result = addition_result + i
    print(addition_result)

add_numbers(user_input)
```

answered May 11 '15 at 4:26

Lali

1 ● 1
