COMP10001

Foundations of Computing

Semester 1, 2021

Tutorial 7

Andrew Naughton

andrew.naughton@unimelb.edu.au

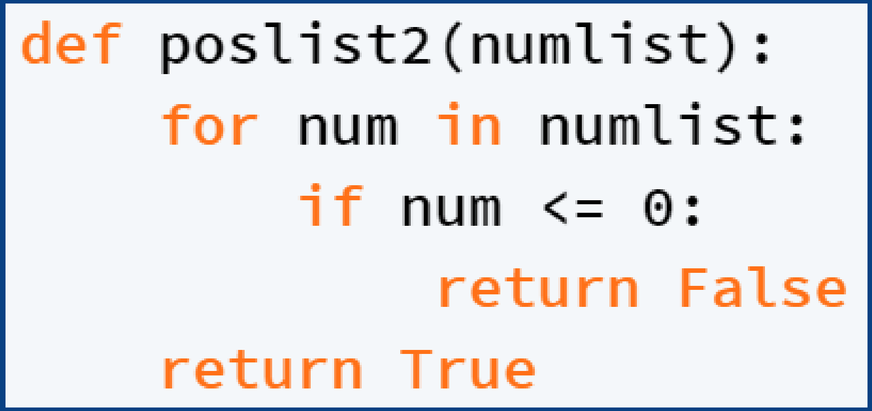
THE UNIVERSITY OF
MELBOURNE

Outline

- Returning Early
- Short circuiting
- Helper functions
- Debugging strategies
- Exercises

Returning Early

- ❖ Also known as “Lazy Evaluation”
- ❖ Where during a long-running computation the answer is returned as soon as it is known
- ❖ Example:
  - ❖ Find whether or not a list contains all positive numbers



```python
def poslist2(numlist):
    for num in numlist:
        if num <= 0:
            return False
    return True
```

Returning Early

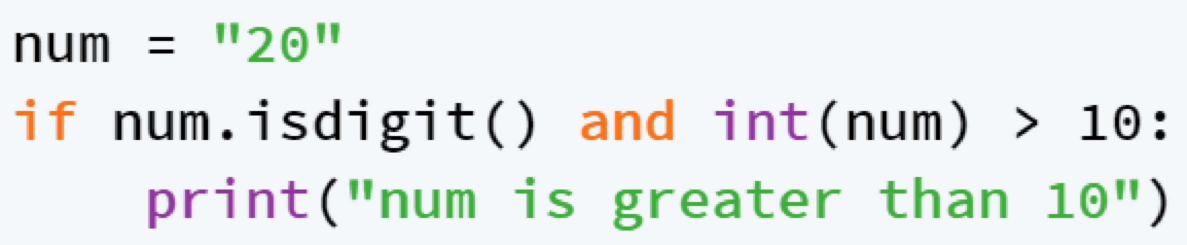
- Avoids unnecessary computation. E.g.,
  - `numlist = [-1,1,1,…,1]`
- Increases efficiency of our program

- This principle is also used in “Short Circuiting” in Boolean tests

THE UNIVERSITY OF
MELBOURNE

Short circuiting

- Typically used in if statements
- Consider this ->

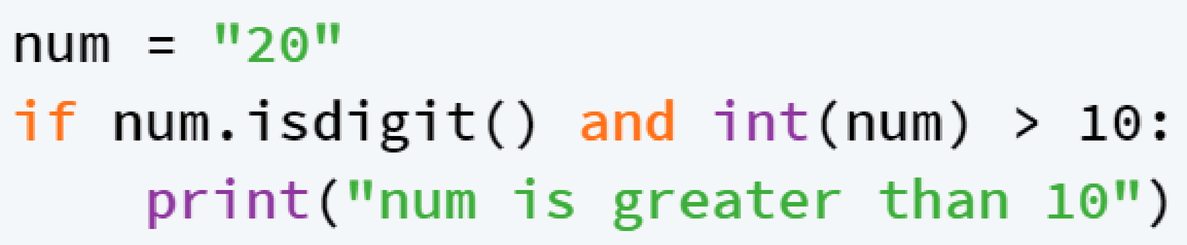


```python
num = "20"
if num.isdigit() and int(num) > 10:
    print("num is greater than 10")
```

- Python will test if num can be converted to a digit via the first test (.isdigit()) before it attempts the conversion (int(num))
- For and: Will not proceed to the second test if the first test is False
- For or: Will not proceed to the second test if the first test is True

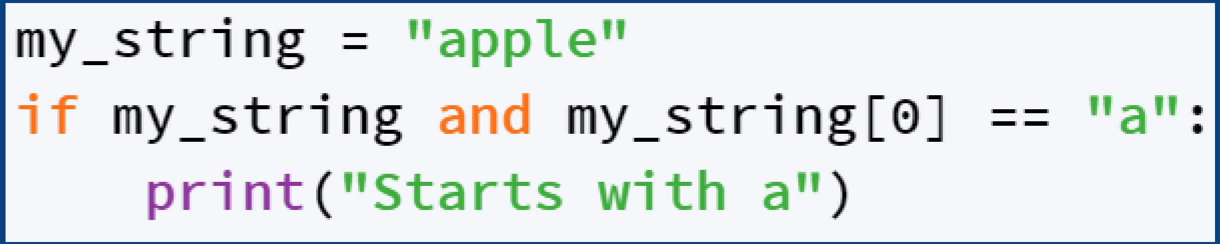
Short circuiting

- This means our code is safer against potential errors. Suppose:
  - num = “twenty”
  - If not for the first test (short circuiting), we would get a ValueError



```
num = "20"
if num.isdigit() and int(num) > 10:
    print("num is greater than 10")
```

- Another typical use case:



```
my_string = "apple"
if my_string and my_string[0] == "a":
    print("Starts with a")
```

Helper functions

- A function that performs part of the computation of another function
- Often make programs more readable because we reduce the complexity (and length) of a function
- E.g. finding the top 5 most frequent words in a paragraph.
- By placing this computation in a helper function, it can be re-used whenever we need to that computation again

THE UNIVERSITY OF
MELBOURNE

Debugging strategies

- Two distinct ways to debug a program:
  - 1. Running test cases
  - 2. Using diagnostic print() statements

Test cases

- These allow us to compare the actual output with the expected output
- Three categories:
  - Normal data (inputs that should be accepted)
  - Boundary/Extreme data (inputs on upper and lower boundaries of what should be accepted)
  - Error data (inputs that should be rejected)

THE UNIVERSITY OF MELBOURNE

Diagnostic print() statements

- To insert print() statements in parts of our code to check the value of variables during execution
- Where values are unexpected or code is not running in the way we intended, we have found where the error is and can write new code to fix the problem

THE UNIVERSITY OF
MELBOURNE

Exercises