

Week 3 Quiz

11 Questions

1. Why is recursion important for traversing tree like structures such as file systems?

- 3/157 **A** It reduces code complexity
- 5/157 **B** It supports the infinite expandability of tree like structures
- 1/157 **C** It *can* be more efficient than loops
- 27/157 **D** A and B only
- 121/157 **E** All of the above

2. Which of the following values will be output from the code pictured here?

- 68/157 **A** 0
- 61/157 **B** 21
- 27/157 **C** 11
- 1/157 **D** -1

```
def recursive_sum(nested_list) -> int:
    total = 0

    for obj in nested_list:
        if type(obj) == list:
            total += recursive_sum(obj)

    return total

print(recursive_sum([1,2,[3,4],[5],6]))
```

i The value for total is never changed, so the final output will be 0

3. Which of the following code snippets will fix the previous function?

- 140/154 **A**
- ```
for obj in nested_list:
 if type(obj) == list:
 total += recursive_sum(obj)
 else:
 total += obj
```
- 11/154 **B**
- ```
for obj in nested_list:
    if type(obj) == list:
        total += recursive_sum(obj)

    total += obj
```
- 3/154 **C**
- ```
for obj in nested_list:
 if type(obj) == list:
 total += recursive_sum(obj)
 else:
 total += 1
```

```
def recursive_sum(nested_list) -> int:
 total = 0

 for obj in nested_list:
 if type(obj) == list:
 total += recursive_sum(obj)

 return total

print(recursive_sum([1,2,[3,4],[5],6]))
```

**i** obj is the value we are attempting to sum, so when obj is not of type list, it must be an integer and therefore should be added to the total.

4. Take a minute to review your code for assignment 1. How many functions does your program have?

liweny1@uci.edu / tonny5@uci.edu / Garcia, Richard / sleestef@uci.edu /  
7/157 mltrinh@uci.edu / citlalr3@uci.edu / idruiz@uci.edu  
| 0

avaratip@uci.edu / lemoral1@uci.edu / amylh1@uci.edu / Quach, Bryan  
4/157 | 1

quaminh@uci.edu / victorg7@uci.edu / anurie@uci.edu / milesjc1@uci.edu /  
7/157 brianlt1@uci.edu / tvahan@uci.edu / rstiruma@uci.edu  
| 2

calvip2@uci.edu / adabrams@uci.edu / limca1@uci.edu / ruixinl3@uci.edu /  
8/157 csimbolo@uci.edu / huuji@uci.edu / shawnth@uci.edu / Pashkut, Zachary  
| 3

dpabilon@uci.edu / ocampoje@uci.edu / kevinp7@uci.edu /  
7/157 DOMINLN1@UCI.EDU / sjdesai@uci.edu / yuanyinz@uci.edu /  
bhuynhto@uci.edu  
| 4

kmurugad@uci.edu / ayelanji@uci.edu / mfyamaza@uci.edu /  
9/157 villanl2@uci.edu / darryell@uci.edu / moeezq@uci.edu / leeau@uci.edu /  
woodwarh@uci.edu / gshanna@uci.edu  
| 5

serenh3@uci.edu / ttwigg@uci.edu / dboghoss@uci.edu / davidlh1@uci.edu /  
11/157 tataylor / hortad@uci.edu / chaueq@uci.edu / mouanouc@uci.edu /  
genevied@uci.edu / brianht2@uci.edu / nichoaa1@uci.edu  
| 6

yipengl7@uci.edu / youhanz@uci.edu / RSakib@uci.edu / Shunwen /  
7/157 nmyee@uci.edu / kylet7@uci.edu / lwchu@uci.edu  
| 7

haiyix2 / jaysonn@uci.edu / Andrew Kim / andyqn1@uci.edu /  
13/157 pinc2@uci.edu / gmgould@uci.edu / rmanohar@uci.edu / mengjix1@uci.edu /  
shiliany / Kasrat@uci.edu / wafridi@uci.edu / vincenbc@uci.edu /  
sinagaaa@uci.edu  
| 8

boweny10@uci.edu / kayleeay@uci.edu / junz22@uci.edu / sjustols@uci.edu /  
6/157 pachase@uci.edu / dryi@uci.edu  
| 9

dscha1@uci.edu / khangtl3@uci.edu / cisincla@uci.edu / jhathira@uci.edu /  
6/157 akhant1@uci.edu / gnecoche  
| 10

⊗ **kshi6@uci.edu / shihao mai / doannb@uci.edu**  
3/157 | 11

⊗ **Jameson Davis**  
1/157 | 14

⊗ **andyqt1@uci.edu / rochelln@uci.edu / jrasheed@uci.edu**  
3/157 | 15

⊗ **Ecson Hsu / ibayoumi@uci.edu**  
2/157 | 20

⊗ **Reyes, Ricardoi**  
1/157 | Currently 2.

⊗ **areshamw@uci.edu**  
1/157 | a couple so far

⊗ **mstu@uci.edu**  
1/157 | A lot around 10-20

⊗ **yinxual1@uci.edu**  
1/157 | I got 4 of them for now

⊗ **tanejad@uci.edu**  
1/157 | 10 approx.

⊗ **ktcampb1@uci.edu / dfriedho@uci.edu / vvijaywa@uci.edu**  
3/157 | 4 so far

⊗ **cabralj4@uci.edu**  
1/157 | i have like 4 functions right now

⊗ **briak10@uci.edu**  
1/157 | Currently, I have four functions.


⊗ **dsumpter@uci.edu / edgarz2@uci.edu / cravottb@uci.edu**  
3/157 | 5 so far


⊗ **Gabriella Carbonero**  
1/157 | haven't finished about 8


⊗ **davidrt@uci.edu**  
1/157 | 4 for now


⊗ **Holmstea@uci.edu**  
1/157 | 0 so far


1/157  **Anabellj@uci.edu**  
7 so far


1/157  **tzancoli@uci.edu**  
a lot of them


1/157  **djayala1@uci.edu**  
So far I have 5


1/157  **bthamilt@uci.edu**  
Well, so far I've just got functions in place for the Q and L inputs, so just two


1/157  **gegarci1@uci.edu**  
So far I have 2 but I think I need to go back and more


2/157  **ywchoi2@uci.edu** / **arwint@uci.edu**  
8 functions


1/157  **francjr3@uci.edu**  
Currently, my assignment 1 has 4 functions.


1/157  **Dongwhee Kim**  
so far 9


1/157  **atkinsr1@uci.edu**  
9 so far


1/157  **bgurkas@uci.edu**  
not done yet ;(


1/157  **nikant@uci.edu**  
uhh like 6

2/157  **hallew@uci.edu** / **cigaya@uci.edu**  
not done yet

1/157  **krogan1@uci.edu**  
not done


2/157  **Johnsoyc@uci.edu** / **walshv@uci.edu**  
8 so far


1/157  **chiual@uci.edu**  
so far it has 4


1/157  **jssu1@uci.edu**  
6 as of right now


 **bavalosh@uci.edu**


1/157 | So far 3


 **sammyp@uci.edu**  
1/157 | I am not done yet, sorry


 **davidn13@uci.edu**  
1/157 | 3 so far


 **Wu, Andre**  
1/157 | I have 2


 **nam mai**  
1/157 | idk didnt finish yet


 **zhenwenl@uci.edu**  
1/157 | not sure


 **bjrobin1@uci.edu**  
1/157 | 5 functions


 **wangm13@uci.edu**  
1/157 | 2 (so far)


 **tariqsb@uci.edu**  
1/157 | So far my code has 3 functions


 **nashv@uci.edu**  
1/157 | 8 but im only half way done


 **yiningw7@uci.edu**  
1/157 | not finished yet...

 **tfermani@uci.edu**  
1/157 | Four

 **latb@uci.edu**  
1/157 | 10 so far

 **nathandl@uci.edu**  
1/157 | not enough, i'm not done

 **pjericks@uci.edu**  
1/157 | I have 5 funtions

 **gscheafe@uci.edu**  
1/157 | Trying to find the best time to work on it, so 0

 **lyuc6@uci.edu**  
|

1/157 | Haven't do that.

⊗ **Qingxu**

1/157 | One for each of the command type, with an extra two for running the code. I'd guess 7 or eight total?

⊗ **sglushki@uci.edu**

1/157 | 0, sadly. too intimidated to start...

⊗ **agee4@uci.edu**

1/157 | not enough :/

⊗ **shultman@uci.edu**

1/157 | four so far

⊗ **KVILLALB@UCI.EDU**

1/157 | 4 currently (In progress ;-;)

⊗ **zhengrw2@uci.edu**

1/157 | 5 for now

⊗ **charlix@uci.edu (Charlie Xu)**

1/157 | 2 functions and a lot of anxiety. Please, this is a cry for help

⊗ **thangnc@uci.edu**

1/157 | I have 3 so far.

⊗ **Sho Nakata**

1/157 | I actually have 0...

⊗ **aalambis@uci.edu**

1/157 | I have about 7 functions all together

⊗ **sohansen@uci.edu**

1/157 | 7 not including the main code block

## 5. Let's talk about functions...

🚫 **Kasrat@uci.edu**

1/152 | many of the code is shared amongst the if else statement indicating we can have another function

🚫 **agee4@uci.edu**

1/152 | files opened in options d, f, and n are closed without operating on them

🚫 **Ecson Hsu**

1/152 | what is trim()?

🚫 **shiliany**

1/152 | f.open(myfile) and f.close() can be put into a function to avoid repeating

🚫 **bhuynhto@uci.edu**

1/152 | redundant code

🚫 **jrchabot@uci.edu**

1/152 | can you compact the code more?

🚫 **mfyamaza@uci.edu**

1/152 | abstract code by using functions

🚫 **vincenbc@uci.edu**

1/152 | the open file procedure can be abstracted

🚫 **dsumpter@uci.edu**

1/152 |  
if option == 'e':  
 for x in y:  
 if x is True:  
 return 0  
elif option in [d, f, n]:  
 if option.trim() in [fo, bar, baz]:  
 f.open(myfile)  
 f.close()  
else:  
 return "ERROR"

🚫 **csimbolo@uci.edu**

1/152 | yes

🚫 **gmgould@uci.edu**

1/152 | d f and n all do the same thing, so check for in ('d', 'f', 'n') then check in ('foo', 'bar', 'baz') and have the f.open() f.close() happen inside that

```
if option == 'e':
 for x in y:
 if x is True:
 return 0

elif option == 'd':
 if option.trim() == 'foo':
 f.open(myfile)
 f.close()

elif option == 'f':
 if option.trim() == 'bar':
 f.open(myfile)
 f.close()

elif option == 'n':
 if option.trim() == 'baz':
 f.open(myfile)
 f.close()

elif option == 'd':
 for x in y:
 if x is True:
 return 0

else:
 return "ERROR"
```

1/152



**davidrt@uci.edu**

Theres a way to condense this

1/152



**dpabilon@uci.edu**

The code is pretty repetitive, we can try and shorten this code by creating an efficient function

1/152



**shultman@uci.edu**

the file can be opened before the conditional statements and closed after

1/152



**boweny10@uci.edu**

we can add a function to simplify the functions

1/152



**nichoaa1@uci.edu**

repetitive opening and closing files

1/152



**citlaln3@uci.edu**

For x in y and the following 2 lines of code is repetitive, so it should be made into a function. Same with "f.open(mydile)" and the following line of code.

1/152



**aalambis@uci.edu**

opening and closing the file is repetitive. so is option.trim

1/152



**gnecoche**

Many of the elif conditions do the same thing.

1/152



**jssu1@uci.edu**

opening and closing a file is repeated so a function can be created.

1/152



**cabralj4@uci.edu**

opening and closing files repeats

1/152



**nathandl@uci.edu**

use a function call instead of repeating code

1/152



**KVILLALB@UCI.EDU**

the if and last elif have the same code after the conditional statements so they can both be part of the same conditional statement. option d, f, and n have same code inside of them so it could be optimized to be made faster and less repetitive.


1/152





**dryi@uci.edu**


open up the file once in the beginning of all the code then close it at the end instead of opening it every time





1/152  **doannb@uci.edu**  
Options d, f, and n can have a function that does the .trim() and opening/closing files. This can simplify the code. Options e and d can also be abstracted into another function due to their similarities.


1/152  **Andrew Kim**  
It is very repetitive. It just opens and closes files. You can make function for that


1/152  **victorg7@uci.edu**  
open and close can be made into one function


1/152  **thangnc@uci.edu**  
They open and then immediately close a file without doing anything to it for option == "e", "d", and "n".


1/152  **akhant1@uci.edu**  
The last elif statement repeats the same process as the first if statement. and the option 'd', 'f', and 'n' repeat the same processes.

1/152  **brianht2@uci.edu**  
They all do the same thing, which can be turned into a function


1/152  **lwchu@uci.edu**  
you can open and close files with a new function


1/152  **Shunwen**  
It's too repetitive

1/152  **lyuc6@uci.edu**  
d,f,n just open and immediately close the file. Nothing has been done.

1/152  **edgarz2@uci.edu**  
The open file and close file is being called in every function.

3/152  **halley@uci.edu / tariqsb@uci.edu / andyqn1@uci.edu**  
repetitive

1/152  **Anabellj@uci.edu**  
d,f,n are doing the same "operations"

1/152  **cravottb@uci.edu**  
option 'd' does the exact same thing as option 'e'

 **chaueq@uci.edu**  
|

1/152 | f.open(myfile) and f.close are repeated 3 times

🚫 davidlh1@uci.edu

1/152 | If blocks for 'd', 'f', and 'n' are practically the same; the only difference is their if condition

🚫 khangtl3@uci.edu

1/152 | doing the same thing. we can make a function to shorten it

🚫 briak10@uci.edu

1/152 | This seems like a way to keep accepting input after the first input

🚫 tanejad@uci.edu

1/152 | 'x is True' can just be 'if x'

🚫 sleestef@uci.edu

1/152 | open and close so much opening and closing... oh the quarrels of life

🚫 jaysonn@uci.edu

1/152 | noticed that the x in y is repeated and opening and closing the file

🚫 kmurugad@uci.edu

1/152 | Can be done more efficiently very repetitive.

🚫 shihao mai

1/152 | It put out the condition and it created the new file for d, f, n option. Furthermore, it will return "ERROR" for the things are not these options

🚫 darryell@uci.edu

1/152 | the trim function is called in each branch

🚫 gegarci1@uci.edu

1/152 | They repeat for x in y and f.open f.close()

🚫 mouanouc@uci.edu

1/152 | options d, f, and n are repetitive

🚫 vvijaywa@uci.edu / Quach, Bryan

2/152 | repetition

🚫 Johnsoyc@uci.edu

1/152 | repetitive so the code can be shortened

🚫 nashv@uci.edu

1/152 | instead of if \_ == \_\_ you could write a function with a

- | parameter for those two blanks
- 1/152 | **sohansen@uci.edu**  
the if and last elif are the same. the middle 3 elifs could use the same function but with a parameter for the strings 'foo', 'bar', and 'baz'
- 1/152 | **sinagaaa@uci.edu**  
It does the same thing
- 1/152 | **ruixinl3@uci.edu**  
too repetitive of some of the functions
- 1/152 | **cisincla@uci.edu**  
d f n and all doing the same thing?
- 1/152 | **brianlt1@uci.edu**  
What are we supposed to be typing here?
- 1/152 | **cigaya@uci.edu**  
Two cases of repetition with 3 instances of if option.trim() and 2 instances of for x in y that both do the same actions
- 1/152 | **shawnth@uci.edu**  
f.open(myfile) and f.close() is used throughout the code multiple times. The coder could use a function to shorten the code by a few lines.
- 1/152 | **zhenwenl@uci.edu**  
the open and close file is repetitive
- 1/152 | **krogan1@uci.edu**  
opening and closing a file
- 1/152 | **adabrams@uci.edu**  
Opening the file each time is repetitve
- 1/152 | **milesjc1@uci.edu**  
'd' & 'e' extraneous
- 1/152 | **Qingxu**  
Would put more obvious variable names (what's x and y?) or at least some comments
- 1/152 | **lemoral1@uci.edu**  
File being open and close is repetitive
- 1/152 | **leeau@uci.edu**  
Repetitive action could be simplified to 2 functions, the first

and last are the same, but the second through fourth are handled similarly and could still be handled in a single function.



**pinc2@uci.edu**

1/152 | repetitive and open and close the file



**huuji@uci.edu**

1/152 | different options do relatively the same thing



**areshamw@uci.edu**

1/152 | opening and closing the file is repetitive and happens every time



**genevied@uci.edu**

1/152 | e and d are the same, d f n and last one are relatively the same



**ocampoje@uci.edu**

1/152 | Repetitive stemants, can optimize code to lessen the amount the code



**Jameson Davis**

1/152 | they open and close stuff for no reason



**yipengl7@uci.edu**

1/152 | two 'd' for the elif



**dfriedho@uci.edu**

1/152 | the first and last blocks are the same and all the body blocks are the same



**sjustols@uci.edu**

1/152 | combine e and the last elif statement into one if statement and make d f and n into a function



**Garcia, Richard**

1/152 | You can make a function that trims an option and opens a file



**mstu@uci.edu**

1/152 | A lot of repetition and doing nothing to the file!



**RSakib@uci.edu**

1/152 | You could make two functions out of the open close file and the return 0 for x executions.



**pachase@uci.edu**

1/152 | Repetitive: (1) for x in y:...  
(2) opening and closing a file

- 1/152 | **rochelln@uci.edu** | repetitive code with opening and closing a file
- 1/152 | **sglushki@uci.edu** | Two elif's for option d? compress everything under one conditional.
- 1/152 | **serenh3@uci.edu** | repetitive code for option e and d. also repetitive for options d, f, n
- 1/152 | **gscheafe@uci.edu** | The first and last elifs have the exact same purpose, since both return 0 from the same conditions
- 1/152 | **bjrobin1@uci.edu** | just opening files and closing them
- 1/152 | **kayleeay@uci.edu** | stuff inside of the 1st, 2nd, and 3rd, elif are the same and the stuff in if and last elif are the same so could just write functions and call them in those places
- 1/152 | **kshi6@uci.edu** | repetitive 'if' statements
- 1/152 | **rstiruma@uci.edu** | can make everything after each branch a different function.
- 1/152 | **jhathira@uci.edu** | very repetitive, open and closes file alot
- 1/152 | **kylet7@uci.edu** | opening and closing repeats
- 1/152 | **villanl2@uci.edu** | can't you just do 'if x:'
- 1/152 | **ibayoumi@uci.edu** | options e and d are the same
- 1/152 | **mengjix1@uci.edu** | we could define a function for repetitive code " f.open(myfile) f.close."
- 1/152 | **yinxual1@uci.edu** | there are 3 functions with the same function, which is unnecessary

1/152 | **ywchoi2@uci.edu** opens the file then closes the file right away without reading it

1/152 | **ktcampb1@uci.edu** option d f and n can be simplified?

1/152 | **walshv@uci.edu** the open and close lines are repetitive

1/152 | **avaratip@uci.edu** open and close files are repetitive

1/152 | **chiual@uci.edu** can use 'and'

1/152 | **bavalosh@uci.edu** It is repetitive

3/152 | **ayelanji@uci.edu / Wu, Andre / calvip2@uci.edu** very repetitive

2/152 | **zhengrw2@uci.edu / nmyee@uci.edu** repetitive statements

1/152 | **nikant@uci.edu** too much opening and closing files without any reason

1/152 | **tvahan@uci.edu** opening and closing the file in each statement is repetitive

1/152 | **Holmstea@uci.edu** opening and closing files are repetitive, and they don't so anything to the file before closing it

1/152 | **rmanohar@uci.edu** opening and closing is repetitive

1/152 | **ttwigg@uci.edu** the for loops are the same and the if option.trim() conditions are the same

1/152 | **liweny1@uci.edu** Open and close

1/152 | **wafridi@uci.edu** Repetition of open and close

**Dongwhee Kim**

1/152 | repetitive

 moeezq@uci.edu

1/152 | Opening and closing files is repetitive

 quaminh@uci.edu

1/152 | open and close file is repeating

 mltrinh@uci.edu

1/152 | option == d twice

 bthamilt@uci.edu

1/152 | f.open() and f.close() repeat every time

 tonyy5@uci.edu

1/152 | elif statements d, f, and n are all similar. It is really repetitive.

 Reyes, Ricardoi

1/152 | D, F, and N options are the same whne they can be within a single function?

 kevinp7@uci.edu

1/152 | open and closing w/o doing anything

 anurie@uci.edu

1/152 | the closing file can just be put at the end

 pjericks@uci.edu

1/152 | Python's version of a switch statement. 3 of the statements basiclly do the same thing.

 jrasheed@uci.edu

1/152 | the files are opened and closed with nothing actually being executed

 limca1@uci.edu

1/152 | The open and closing of the file can be abstracted into a function

 idruiz@uci.edu

1/152 | seems repetitive

 latb@uci.edu

1/152 | we can combine the if and last elif. if option == 'e' or option == 'd':...

 davidn13@uci.edu

1/152 | Repetitive opening and closing of the file

1/152 | **francjr3@uci.edu**  
Opening and closing seems excessive

1/152 | **sjdesai@uci.edu**  
a lot of opening and closing

1/152 | **gshanna@uci.edu**  
function is unorganized and repetitive

1/152 | **charlix@uci.edu (Charlie Xu)**  
Why would you open the file and then close it immediately after. Actually, sounds like what I would do

1/152 | **DOMINLN1@UCI.EDU**  
some statements repeated can be turned into functions?

1/152 | **sammyp@uci.edu**  
The code can be shortened using functions especially for opening and closing the file.

1/152 | **hortad@uci.edu**  
functions seem unorganized and repetitive

1/152 | **yiningw7@uci.edu**  
repetitive ones

1/152 | **dboghoss@uci.edu**  
Two of the if statements are similar, if the option is either e or d then the same code runs so it is better to combine them with an or statement

1/152 | **yuanyinz@uci.edu**  
open and close

1/152 | **junz22@uci.edu**  
open and close file is repetitive


1/152 | **tzancoli@uci.edu**  
first if statement and last elif statement do the exact same thing so should be combined.


1/152 | **tataylor**  
pretty repetitive


1/152 | **bgurkas@uci.edu**  
repetitive opening and closing, what's trim


1/152 | **amylh1@uci.edu**  
open and closing files are repetitive





 **Gabriella Carbonero**  
1/152 | all elifs do the same thing


 **dscha1@uci.edu**  
1/152 | One can combine with or statements to shorten the code.


 **andyqt1@uci.edu**  
1/152 | they open a file and immediately close without doing any reading or writing


 **tfermani@uci.edu**  
1/152 | have a differenct function for opening and closing a file.

 **haiyix2**  
1/152 | only conditions there

 **arwint@uci.edu**  
1/152 | don't need two elifs for 'd' to be honest

 **nam mai**  
1/152 | lots of trimming in the functions

 **Sho Nakata**  
1/152 | I mean looks good to me.

 If you are taking the late quiz, please watch the recorded lecture for discussion about this question!

6. Are you using the IDLE debugger or any Python debugger to analyze your code when it doesn't work as expected?

96/156  Yes

61/156  No

7. Which of the following Python statements can be used to test the output of a function in your code?

14/155  raise

3/155  pass

138/155  assert

0/155  del

8. What convention do programming languages like Python use to scope the code that we write?

- 4/159 ☐ A Access
- 27/159 ☐ B Modules
- 18/159 ☐ C Scope
- 110/159 ☒ D Namespaces

9. Which of the following is **NOT** a type of namespace supported by Python?

- 0/159 ☐ A Local
- 98/159 ☒ B Included
- 53/159 ☐ C Enclosed
- 7/159 ☐ D Built-In
- 1/159 ☐ E Global

10. Python will convert all variable and function names to *private* access when their names are prepended with an underscore like so:

```
def _myprivatefunction():
 pass
```

- 78/159 ☐ T True
- 81/159 ☒ F False

**i** The underscore is a naming *convention*, private attributes are not supported by the Python interpreter.

11. Which of the following Python statements allows us to link modules together?

- 145/160 ☒ A import
- 2/160 ☐ B export
- 1/160 ☐ C scope
- 3/160 ☐ D def
- 6/160 ☐ E global
- 3/160 ☐ F assignment