Lists

by shashank21j

dennysregalado 2 years ago

I just use eval function

```
n = input()
l = []
for _ in range(n):
    s = raw_input().split()
    cmd = s[0]
    args = s[1:]
    if cmd !="print":
        cmd += "("+ ",".join(args) +")"
        eval("l."+cmd)
    else:
        print l
```

241 Add Comment Permalink

Jourl 2 years ago

great :-)

0 | Add Comment Parent Permalink

mdavis 2 years ago

I also used eval but yours is so much cleaner! Thanks!

```
L = []
for _ in range(0, int(raw_input())):
    user_input = raw_input().split(' ')
    command = user_input.pop(0)
    if len(user_input) > 0:
        if 'insert' == command:
            eval("L.{0}({1}, {2})".format(command, user_input[0], user_i
nput[1]))
```

```
    else:
        eval("L.{0}({1})".format(command, user_input[0]))
elif command == 'print':
    print L
else:
    eval("L.{0}()".format(command))
```

8 | Add Comment Parent Permalink

robipolli 6 months ago

eval is evil ;)

9 | Add Comment Parent Permalink

eyshikaengineer 5 months ago

Here is mine

N = int(input()) list1=[] for i in range(N): cmd=input().split() command=cmd[0] if len(cmd)>1: index=int(cmd[1]) if len(cmd)>2: number=int(cmd[2]) if command=='insert': list1.insert(index,number) if command=='append': list1.append(index) elif command=='remove': list1.remove(index) if command=='print': print(list1) elif command=='pop': list1.pop() elif command=='reverse': list1.reverse() elif command=='sort': list1.sort()

2 | Add Comment Parent Permalink

ramanchopra_8 3 months ago

Why we have use .split command what it is used for please can you explain me with an example.

-1 | Add Comment Parent Permalink

CaymanEss 3 months ago

The Python docs have good string.split() examples.

0 | Add Comment Parent Permalink

sanyenoh 10 days ago

split basically converts the data into a list,since we have to work with list in this code.

0 | Add Comment Parent Permalink

ishanshekhar0502 3 months ago

What a piece of code , man!

0 | Add Comment Parent Permalink

waldon 2 years ago

learned a lot from your code!But I'm wondering that is "for _ in range(n):" a good habbit ?

4 | Add Comment Parent Permalink

omnius42 about a year ago

Using underscore ("_") as the variable name in the for appears to be common practice when you don't intend to use the variable in the loop. Underscore is a valid variable name, so you *could* use it in the loop, but it would make for unreadable code (what does underscore mean?) so using it designates that the variable has no meaning or is otherwise unnecessary.

58 Add Comment Parent Permalink

Max_Vignesh 11 months ago

Nice

Thanks & regards

1 | Add Comment Parent Permalink

Zurda 8 months ago

This thread taught me some valuble new things. Thanks!

0 | Add Comment Parent Permalink

mahdi420 about a year ago

Great job bro

0 | Add Comment Parent Permalink

ilango240692 6 months ago

different dimensions

0 | Add Comment Parent Permalink

pavanbawdane about a year ago

Sneaky. It took some time for me to understand it but altogether, it's an excellent piece of code.

Learnt a lot. Thumbs up!

0 | Add Comment Parent Permalink

MJKG1 about a year ago

hii bro, I am new to python language. Will you please explain the code which you used especially eval function..

-16 Add Comment Parent Permalink

varunkmr26 about a year ago

yeah even I nedd help regarding the same

-2 Add Comment Parent Permalink

masquerade0097 7 months ago

Google it dude

-9 Add Comment Parent Permalink

ishanshekhar0502 3 months ago

The eval function lets a python program run python code within itself. Let me explain you the entire code :) 1.First we take input of how many instruction user will enter. 2.We declare an array. 3.Then loop starts ,which will run n times. 4.s=input().split()..... it takes multiple input at one go.(Remember it takes input in string format, you need to convert it according to your use) What split does is explicitly specify split(' ') because split() uses any whitespace characters as delimiter as default. 5.As per the format of question the first item of 's' will be the command i.e insert ,append , print etc. 6.And starting from item 1 onwards we will get our argument eg. insert 0 1... which means item0 which is our command viz "insert" and item one onwards we get our arguments viz 0,1 . 7.Now we check whether or not the command is "Print".I f it is "print" we simply print the list. else 8.we join our command and arguments . so our final command becomes insert(0,1). (refer point 6) 9.Now, eval run l.insert(0,1) within itself what is happening in eval("l."+cmd) is: l(which is our list) we concat list with the command , and we all know the use of'.' l.command(attributes)

28 Add Comment Parent Permalink

kas111ph 2 months ago

thanks for explaining code ..it did a great help

0 | Add Comment Parent Permalink

khamzah22 2 months ago

If this is not sexy then what is. Piece of art. Learnt a lot

0 | Add Comment Parent Permalink

shahane_rohan96 2 months ago

Thank you very much for the explanation !!! Very much appreciate it !! :D

0 | Add Comment Parent Permalink

phani653 3 days ago

Thank you very much. Nice explanation

0 | Add Comment Parent Permalink

Dizzzy about a year ago

This is an elegant solution. Great job.

0 | Add Comment Parent Permalink

MJKG1 about a year ago

hey hii Dizzzy, Did you understand that code which you commented on?? if yes please explain me..

-28 Add Comment Parent Permalink

zigzagable about a year ago

I am new to Python, but eval() is considered bad practice and contains a security hole: http://stackoverflow.com/a/9384005

This is how I made mine and it avoids that security hole:

cmd = "l.{}({},{})".format(cmd, num[0], num[1])

exec(cmd)

16 Add Comment Parent Permalink

JackVo about a year ago

There are always another solution. If only to pass the challenges, he can use eval. It's up to him. I appreciate your code above. Great job.

-2 Add Comment Parent Permalink

PauloPerna about a year ago

Can someone explain this line?

cmd += "("+ ",".join(args) +")"

1 Add Comment Parent Permalink

jessedgb 11 months ago

I'm also trying to understand this, and am picking it appart one piece at a time. ----------Number one----------

```
cmd +=
```

here he is appending some text to the 'command' string that gets evaluated later in the code.

----------Number two----------

```
cmd += "("+ ",".join(args) +")"
```

Here the brackets on the far left and right sides have quotes around them because he is trying to construct the part where the two numbers(arguments) are used by the function. The comma also has quotes because it is needed between those two numbers.

the plus signs simply join the pieces of text together, and the .join(args) is used to insert the numbers into the argument section of the function.

my problem in understanding is that I would have assumed that the .join(args) would need to be written befor and after the comma in that line of text.

6 Add Comment Parent Permalink

psramam 9 months ago

Excellent explanation.....thank you very much

0 Add Comment Parent Permalink

benyscott 9 months ago

The reason why it's written before is that **.join()** is a function you call on a **string**. The string is what joins the arguments. The string here being a comma the result of **",".join(args)** is **2,3** (in the case that **args = [2,3]**)

3 Add Comment Parent Permalink

Ankur_Beginning 2 months ago

I am not getting this line of code fully, will you please explain again.

0 | Add Comment Parent Permalink

g_jaswanth about a month ago

join() is string method which takes either a string or List of strings as parameters ex:- join("hai") or join(["hai","bye"])

the "," before .join() is delimiter. means by using comma JOIN joins the hai and bye

so ",".join(["hai","bye"])

gives you hai,bye

same way here he is passing ",".join(args) args = s[1:] means some list

1 | Add Comment Parent Permalink

dhrub0216 about a year ago

```
cmd += "("+ ",".join(args) +")"
```

Meaning of this line

-3 Add Comment Parent Permalink

|

Ekwy_ 8 months ago

cmd += "(" + ",".join(args) + ")" the string cmd receive the following string: "(" added of ",".join(args) added of ")"

The function join links 2 strings, in this case : "," and args. It separates all the variables in args with "," and turns it in a string.

1 | Add Comment Parent Permalink

aamarques 2 months ago

It's the same of: cmd = cmd + "(" + ",".join(args) + ")"

if your input is "insert 0 5", the result will be: insert = insert(0,5)

This command joins cmd plus (plus args comma separated and)

Hope you understand!

4 | Add Comment Parent Permalink

js001 about a year ago

This answer is so much better. I'm not sure if I admire you or hate you :)

2 | Add Comment Parent Permalink

bipinoli90 about a year ago

Thanks for sharing :)

0 | Add Comment Parent Permalink

munish01211 about a year ago

can anybody explain me the meaning or purpose of s[1:]

0 | Add Comment Parent Permalink

isobel_dawn_tay1 about a year ago

Splices the list to give everything from the index 1 (the second thing in the list) to the end. In this case, that means everything except the command name.

1 | Add Comment Parent Permalink

paul_e_gradie 7 months ago

This is actually a really interesting question, and it confounded me for a little while until I looked it up. So the deal is that when you slice a list, you will always return a list, no matter what. This isn't the case with indexing a list, which is a different method and has a different set of possible return values. Check out this code:

python

x = [0, 1, 2, 3, 4, 5] # create a list

print x

> [0, 1, 2, 3, 4, 5]

Indexing

print x[3]

> 3

print x[100] # try to return an index clearly out of bounds

Traceback (most recent call last): File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2885, in run_code exec(code_obj, self.user_global_ns, self.user_ns) File "", line 1, in x[100]

IndexError: list index out of range

Slicing

print x[3:4] # a one element slice starting at element three

> [3]

print x[1:5]

> [1, 2, 3, 4] # don't forget slices end before the second argument element

Now for the illuminating cases

print x[3:100] # a much larger slice starting at element 3

> [3, 4, 5]

The returned list included everything beyond 3, starting from 3.

print x[100, 200] # an index range that is clearly out of bounds

> []

The returned list includes nothing, because there is nothing between 100 and 200.

So you see, the slice method only returns list, even if the slice is out of range of the list indices. This is due to how the code is written behind the slice method for lists, which clearly always ends with returning a list, empty or not.

6 | Add Comment Parent Permalink

netampraveen about a year ago

really helpful, thanks a lot

1 | Add Comment Parent Permalink

dohyunio 12 months ago

What's the difference between input() and raw_input()? Why did you do them in those lines respectively?

0 | Add Comment Parent Permalink

netampraveen 12 months ago

in raw_input() you give input as string while in input() as a interger

-10 Add Comment Parent Permalink

mehulj5775 12 months ago

Me new to python. Can you let me know where i went wrong?

"""Traceback (most recent call last): File "solution.py", line 4, in L = L.insert(1,10) AttributeError: 'NoneType' object has no attribute 'insert'"""

No response to STDOUT and run time error.

My Code:

L = [] L = L.insert(0,5) L = L.insert(1,10) L = L .insert(0.6) print(L) L.remove(6) L.append(9) L.append(1) L.sort() print(L) L.remove() L.reverse() print(L)

0 | Add Comment Parent Permalink

isobel_dawn_tay1 12 months ago

The insert method modifies the list in place. It doesn't return a modified version of the list.

So, on your first insert, you're modifying the list, then assigning to L the return value, which is None.

From then on, you're trying to use the insert method of None, which doesn't exist.

Just do L.insert rather than L = L.insert

0 | Add Comment Parent Permalink

mehulj5775 11 months ago

L = []

L.insert(0,5)

L.insert(1,10)

L .insert(0,6)

print(L)

L.remove(6)

L.append(9)

L.append(1)

L.sort()

print(L)

L.remove(10)

L.reverse()

print(L)

Hi, i passed the test0 case but it shows that i failed the test1, even though i am getting the expected output.

2 | Add Comment Parent Permalink

decentkaushal09 11 months ago

I tried the same method that you tried but got error. After reading these comments I got to know where I was wrong. Thanks

-6 Add Comment Parent Permalink

|

mehulj5775 11 months ago

can you let me know where i am wrong?

0 | Add Comment Parent Permalink

manasveemittal 8 months ago

you were intructed to form a loop.

0 | Add Comment Parent Permalink

SSJIV 8 months ago

You are trying to implement the example function directly. It works because the test case 1 is exactly same as the example mentioned.

However, the test case 2, gives in the raw input in form of 'string'. You need to evaluate each function and convert it correspondingly to subsequent function i.e, 'insert' which is a string becomes list1.insert(argument)

0 | Add Comment Parent Permalink

mdzeeshanali93 10 months ago

This is not a solution. you just printed all that has been asked as expected output, this code will not work for other inputs

0 | Add Comment Parent Permalink

joejoooo1 3 months ago

if **name** == **'main'**: for N in range(0,12): List = [] List.insert(0,5) List.insert(1,10) List.insert(0,6) print(List) List.remove(6) List.append(9) List.append(1) List.sort() print(List) List.remove(10) List.reverse() print(List)

0 | Add Comment Parent Permalink

aswanthg7 12 months ago

its insert(0,6) not insert(0.6)

1 | Add Comment Parent Permalink

sugalivijaychari 11 months ago

u supposed to write L.insert(0,6) but u wrote L.insert(0.6)

1 | Add Comment Parent Permalink

mehulj5775 11 months ago

oops, thank u so much

1 | Add Comment Parent Permalink

decentkaushal09 11 months ago

its still showing error in test case 1.

4 | Add Comment Parent Permalink

miku86coding 2 months ago

Because you are hardcoding the numbers. The function should read the input. If the numbers change, your code should return the changed numbers, not your hardcoded ones.

0 | Add Comment Parent Permalink

eltonplima 12 months ago

careful when using eval, because eval is evil.

Read the book: Dive into Python to understand.

2 | Add Comment Parent Permalink

mdavis 7 months ago

Agreed! eval is very dangerous to use in production and there is always a more pythonically correct way.

-1 | Add Comment Parent Permalink

rmtustc 12 months ago

why it doesn't compile in python 3?

0 | Add Comment Parent Permalink

eltonplima 12 months ago

In python3 the print is a function.

In python2 you use print l

In python3 you use print(l)

3 | Add Comment Parent Permalink

rmtustc 12 months ago

wow, thanks, impresive

0 | Add Comment Parent Permalink

deepenpatel19 11 months ago

Great!!!

we are await for new command and tricks to solve complicated problem in an easy way..Like This...

-1 | Add Comment Parent Permalink

e2dost 11 months ago

Excellent solution. Just one thing: it's better to use **n=int(input())** I think, rather than **n=input()**, in order to cover the error that sometimes occurs when calling the 'for' loop...

1 | Add Comment Parent Permalink

ahmadzakialsafi 11 months ago

creative

0 | Add Comment Parent Permalink

mmkc520 11 months ago

could you plese explain how this works

0 | Add Comment Parent Permalink

Max_Vignesh 11 months ago

Well this one is a good code. But I cant understand the sixth line of your code. Can u please explain the flow of this program.

Thanks & regards

0 | Add Comment Parent Permalink

Max_Vignesh 11 months ago

Well this one is a good code. But I cant understand the sixth line of your code. Can u please explain the flow of this program.

Thanks & regards

0 | Add Comment Parent Permalink

ashokkumarsudha 11 months ago

hey can u exaplain me this line step by step cmd += "("+ ",".join(args) +")"

0 | Add Comment Parent Permalink

wittrup 10 months ago

tl;dr ;) Two-liner:

```
l=[]
[getattr(l,s[0])(*map(int,s[1:])) if hasattr(l,s[0]) else print(l) for s
 in [input().split() for _ in range(int(input()))]]
```

11 | Add Comment Parent Permalink

LurkyTheHatMan 9 months ago

This is full of beautiful automagic (I adore list comprehensions when they don't become too beastly).

For a while, I wondered why there was some mystical extra line, but then I remembered that you're using a list comprehension, which spits out a list, and the interpreter prints unassigned return values :/

Anyway, you sir win one internet(s)!

2 | Add Comment Parent Permalink

akik22 3 months ago

@wittrup, Great piece of code bro :)

For newbie like me, below is the breakdown of the code.

```
l=[]
for x in range(int(input())):
        s = input().split()
        if hasattr(l,s[0]):
                getattr(l,s[0]) (*map(int,s[1:]))
        else:
                print (l)
```

2 | Add Comment Parent Permalink

deepak9046 2 months ago

Could you please explain the code?

1 | Add Comment Parent Permalink

hemantsingh1612 about a month ago

`hasattr` checks if the input operation to be performed is valid or not .

`getattr(l,s[0]) (*map(int,s[1:]))`

Here we perfom the operation on the rest of the elements in input *map is to convert the remaining input into 1,..3 format.

1 | Add Comment Parent Permalink

CrazyYoungBro 10 months ago

Could you explain what

```
cmd += "("+ ",".join(args) +")"
```

```
cmd += "("+ ",".join(args) +")"
```

means?

1 | Add Comment Parent Permalink

wittrup 10 months ago

It means that the sequence of the elements in args will be joined with the string ',' and have a "(" prefix and ")" suffix. Basically, if args are ['0','5'] the new string will be "(0,5)", and then all this appended to cmd by the cmd+= See https://www.tutorialspoint.com/python/string_join.htm

4 | Add Comment Parent Permalink

VforVivek 10 months ago

Great indeed ;)

0 | Add Comment Parent Permalink

JXSan 9 months ago

Wow, it took me some time to understand what was going on here; but fantastic code!

0 | Add Comment Parent Permalink

quincy_wofford 9 months ago

Slightly more concise take on this eval approach:

```
arg1 = input()
l = []
for _ in range(int(arg1)):
    s = input().split()
    if (s[0] != 'print'):
        eval("l.{}(".format(s[0])+','.join(s[1:])+")")
    else:
        print(l)
```

Also worth noting that unpacking a list with *l or a dictionary with **dict could be useful in similar problems.

2 | Add Comment Parent Permalink

Yusilg 9 months ago

love your code. thanks

0 | Add Comment Parent Permalink

rutwick_222 9 months ago

could you please explain your code?

1 | Add Comment Parent Permalink

quincy_wofford 9 months ago

Sure.

str.split() will take the string, str and use white space as a delimiter to turn the contents of str into a list of strings.

example: "so this becomes".split() = ['so','this','becomes']

The first value in each of the lists created within the for loop is a string that matches the method we wish to use. eval takes strings as input, and runs them as Python code.

example: eval("print("same")") == print("same")

The format code is taking the first element of the list, s[0], and substituting it for the "{}" contained in my eval string.

example: "{}{}{}".format("Hello"," ","World") becomes "Hello World"

','.join(list) takes each element of a list, and appends them into a string using ',' as the delimiter. the + operator here is just appending the front half of my eval statement to the back half of my eval statement.

example: ','.join(['this',' ','is',' ','nice']) becomes 'this, ,is, ,nice'

example: "l.insert(" + "1,2)" = "l.insert(1,2)"

All of these steps take advantage of methods that are built into Python's list data type. That's why we can do list.METHOD(arguments).

8 | Add Comment Parent Permalink

uppoorabhiram 8 months ago

Super

0 | Add Comment Parent Permalink

rushabh123453 8 months ago

Hello dennysregalado,when I saw your code I thought I am seeing the wrong code for the question that was given,but it just astonished my mind after reading your code.Initially I thought we were supposed to build our own functions for given operators

0 | Add Comment Parent Permalink

norajsinger 8 months ago

very nice code

0 | Add Comment Parent Permalink

paragguruji 7 months ago

Please see my alternative with ***getattr*** above and comment if you can compare it with ***eval*** w.r.t. efficiency.

0 | Add Comment Parent Permalink

dev_gabriel_lima 7 months ago

Good implementation!

0 | Add Comment Parent Permalink

jonobjornstad 7 months ago

very smooth

0 | Add Comment Parent Permalink

natterstefan 7 months ago

Hello,

thanks for your solution, works great. I have a question though (newbie here): Why does

```
s = raw_input().split()
```

(or another part I am missing) return line by line of the input alreay? I do not get this yet.

1 | Add Comment Parent Permalink

baldockja about a month ago

split() turns a string into a list of strings. It "splits" the string on whitespace characters, for example:

```
"a b c" => ["a", "b", "c"]
```

0 | Add Comment Parent Permalink

scythe042 7 months ago

what does s[1:] mean?.

0 | Add Comment Parent Permalink

abhinavwalia95 6 months ago

It's so clean and simple, made my day :)

0 | Add Comment Parent Permalink

omavinashkalal 5 months ago

here what is role of "_" in for loop

1 | Add Comment Parent Permalink

so_no 5 months ago

`eval` should be used for arithmetic expressions.

`exec` is the function for commands.

1 | Add Comment Parent Permalink

codebloode_123 5 months ago

great work man

0 | Add Comment Parent Permalink

vimquat 4 months ago

Thanks, the problem is confusing at first but this cleared up a lot of things and I still learned what I needed to :)

0 | Add Comment Parent Permalink

ajaysubramanian 3 months ago

nice one!

0 | Add Comment Parent Permalink

alberto_pranted1 3 months ago

I love that style

1 | Add Comment Parent Permalink

Lightyaer 3 months ago

Can anyone explain why @dennysregalado code works on python 2 and not on python 3 except the raw_input factor

0 | Add Comment Parent Permalink

alberto_pranted1 3 months ago

Certainly the print function; on python 3 you have to use the parenthesis on print function.

For more detailed informations about differences between python 2 and 3, you can read it:

What's New in Python 3

0 | Add Comment Parent Permalink

Ankur_Beginning 2 months ago

what is this cmd += "("+ ",".join(args) +")". I am new at python. Anyone please help me.

0 | Add Comment Parent Permalink

Hypro999 2 months ago

It's beautiful code once you understand it. Thanks for sharing it here. But you made a few mistakes regarding following PEP 8 (which you should REALLY try to follow) and this leads to a decrease in readability.

1.

you generally shouldn't use lowercase l, uppercase O, or uppercase I as variable names because they can be confused with 1 or 0 hence decreasing readability.

2.

that cmd + line can be cleanly written as:

```
cmd += "(" + ",".join(args) + ")"
```

1. In python 3 print is a function so you should type: print(l) in the last line

But the idea is beautiful and VERY clean.

0 | Add Comment Parent Permalink

sairamkamalay 2 months ago

Brilliant

0 | Add Comment Parent Permalink

kumarswapnil9900 2 months ago

Could you please explain the above code as i am not able to see any discription,please i am new to this and got stuck with this.

0 | Add Comment Parent Permalink

pawel_wujkiewicz 2 months ago

Just one 'if', nice :)

0 | Add Comment Parent Permalink

kumarswapnil9900 2 months ago

Thank you for replying ... I tried with if condition thought didn't get excat output, could you please explain this with an example ..

Thanks in advance

0 | Add Comment Parent Permalink

praneethkalimis1 about a month ago

can you explain this?

0 | Add Comment Parent Permalink

divya_gracie 20 days ago

simpler

```
if __name__ == '__main__':
    N = int(raw_input())
    my_list=[]
    for x in range(N):
        args= raw_input().strip().split()
        args[1:]=map(int,args[1:])
        try:
            getattr(my_list,args[0])(*args[1:])
        except AttributeError:
            print my_list
```

0 | Add Comment Parent Permalink

taha_karachi 12 days ago

can anyone explain me, why at this line

```
cmd += "("+ ",".join(args) +")"
```

"," is use before .join ?

and why i couldn't be done with

```
cmd += "("+ .join(args) +")"
```

because in the trailing part, there is no "," between (args) and +

0 | Add Comment Parent Permalink

kaydeesullivan 9 days ago

Any chance you can explain this line to me: cmd += "("+ ",".join(args) +")"

I understand that you're concatenating the cmd variable to run the command entered, but don't understand why the comma is where it is (","). I would have thought it should go after .join(args) but that doesn't work.

0 | Add Comment Parent Permalink

taha_karachi 8 days ago

comma works as a separator for values of args

0 | Add Comment Parent Permalink

jadams76 about a year ago

These questions are poorly worded and not clear at all.

46 Add Comment Permalink

xorith 2 years ago

This worked first try. Thought I would share because so many seemed to be having issues. I've been in the habit of always using `raw_input().strip()` or `input().strip()` (Python 3). It saves you many headaches trying to figure out non-printable characters.

I didn't bother with a map statement. Probably could have wrapped it in a `try` block, but it still passes. :)

```
T = int(raw_input().strip())

L = []
for t in range(T):
    args = raw_input().strip().split(" ")
    if args[0] == "append":
        L.append(int(args[1]))
    elif args[0] == "insert":
        L.insert(int(args[1]), int(args[2]))
    elif args[0] == "remove":
        L.remove(int(args[1]))
    elif args[0] == "pop":
        L.pop()
    elif args[0] == "sort":
        L.sort()
    elif args[0] == "reverse":
        L.reverse()
    elif args[0] == "print":
        print L
```

31 Add Comment Permalink

xorith 2 years ago

And for the sake of wanting to learn... I decided to look up how to replicate the ability to call a function via a name stored in a variable. In PHP, it would be `$func(var);` but I found in Python, it is `getattr(Object, Function)(Vars)`

Example of using this method below. Again, this is assuming the user isn't going to enter in an invalid command!

```
T = int(raw_input().strip())
L = []
for t in range(T):
    args = raw_input().strip().split(" ")
    if args[0] == "print":
```

```
    print L
elif len(args) == 3:
    getattr(L, args[0])(int(args[1]), int(args[2]))
elif len(args) == 2:
    getattr(L, args[0])(int(args[1]))
else:
    getattr(L, args[0])()
```

86 Add Comment Parent Permalink

neo1691 2 years ago

Thanks a lot. I always wondered why getattr is used. This made a lot of sense!!

edit: typo

0 | Add Comment Parent Permalink

nvillanueva 2 years ago

I went on a similar route, and to avoid the if/elif's, you can use wildcards. Also, you can check out what methods the list has, using dir() and then, callable()

Example:

```
arr = list()
tests = input()
for test in range(int(tests)):
    test = input()
    instruction = test.split(" ")
    if instruction[0] in dir(arr) and callable(getattr(arr, instruction
[0])):
        args = instruction[1:]
        args = list(map(int, args))  # str -> int
        f = getattr(arr, instruction[0])  # get the callable method
        f(*args)  # and send it *whatever* argument you have
    else:
        if instruction[0] == "print":
            print(arr)
```

6 | Add Comment Parent Permalink

berteun 2 years ago

This can be done a bit shorter by just catching an exception (you could also catch TypeError):

```
L = []
for _ in range(int(input())):
    command, *args = input().split()
    try:
        getattr(L, command)(*(int(x) for x in args))
    except AttributeError:
        print(L)
```

34 Add Comment Parent Permalink

weizhao0 2 years ago

Why not use if statement to check if the command is 'print'? I think it's clearer.

0 | Add Comment Parent Permalink

berteun 2 years ago

You could do that as well, e.g. (which might actually be a bit better).

```
if command == 'print':
    print(L)
else:
    getattr(L, command)(*(int(x) for x in args))
```

However, I'd argue strongly the way nvillaneuva structures the code by first doing some investigation if the call is going to work, is not so idiomatic. See, e.g. the python documentation. You should just do it and catch only does exceptions you can fully handle at that point and let the rest propagate.

This is my main point.

7 | Add Comment Parent Permalink

nvillanueva 2 years ago

Huh, I went for safety there, and that's why I checked for callable(). I'll give it another look as soon as I get home. Also, I liked your `except AttributeError` better than the `if`. I tried to make it short, more than readable/idiomatic (considering this is not meant to be production-ready code)
Thanks!

0 | Add Comment Parent Permalink

conquer_you about a year ago

cool

0 | Add Comment Parent Permalink

sarathid 2 years ago

This is great..

0 | Add Comment Parent Permalink

soren_olegnowicz 2 years ago

Better to ask forgiveness than permission

```
import this
```

0 | Add Comment Parent Permalink

Shivan111 2 years ago

berteun can you explain your getattr statement i just read about how the getattr works but im not able to understand your statement

0 | Add Comment Parent Permalink

ericali 2 years ago

awesome

0 | Add Comment Parent Permalink

YaMogu about a year ago

It looks so cool and pythonic to me. What's the shortest way to do it in Python2?

0 | Add Comment Parent Permalink

rschmidt_z about a year ago

this is the most undesrstandable and reliable piece of code to solve this problem (for me)

0 | Add Comment Parent Permalink

maximshen about a year ago

So getattr is like java reflection. Good to know, thanks.

0 | Add Comment Parent Permalink

adityakamath 2 years ago

^ Makes sense! Thanks a lot, using all those if/elifs was stupid lol

0 | Add Comment Parent Permalink

yadavsunil00001 2 years ago

even after solving question, i always look for discussion, learn many things as a beginner in python.

2 | Add Comment Parent Permalink

Shivan111 2 years ago

can you explain your getattr statement i need to know how it works here

1 | Add Comment Parent Permalink

douhaolee 2 years ago

Thanks, this is great.

0 | Add Comment Parent Permalink

buddhaseeking 2 years ago

The above code (by xorith) is not working for extend() function which accepts a single list as a parameter.

0 | Add Comment Parent Permalink

metusiast about a year ago

Ouput is ok without, strip() function. Why didyou use it?

0 | Add Comment Parent Permalink

cursiv3 about a year ago

This code helped me SO much in understanding args, thank you!

0 | Add Comment Parent Permalink

Dipenshah90 about a year ago

Beautiful!

0 | Add Comment Parent Permalink

saitosean about a year ago

```
n = int(raw_input())
l = []
for i in range(n):
    command = raw_input().split()
    if command[0] == "print":
        print l
    elif len(command) == 1:
        getattr(l, command[0])()
    else:
        # Cast the params to int
        getattr(l, command[0])(*map(int, command[1:]))
```

Similar, but I use the * operator to unpack the arguments if there are any.

2 | Add Comment Parent Permalink

jefjob15 10 months ago

Cool. I knew there had to be a way to unpack a number of arguments that may vary in length. Knew about *, but couldn't get the syntax right, like this:

```
# I tried this and got SyntaxError
(int(a) for a in *command[1:])

# And I tried this and got TypeError
int(*command[1:])
```

If I had used *map in front of that second attempt (and not *command), I suppose it would've worked out.

0 | Add Comment Parent Permalink

jasek_t 12 months ago

Thanks! More pythonic way of your solution (I think). N = int(input()) L = [] for _ in range(N): line = input().strip().split() cmd = line[0] args = [int(x) for x in line[1:]] if cmd != 'print': getattr(L, cmd)(*args) else: print(L)

0 | Add Comment Parent Permalink

yogoth 12 months ago

Heh, it's clever, but the original one you posted is way more readable.

1 | Add Comment Parent Permalink

josa1 about a year ago

```
T = int(raw_input())

    L = []
    for t in range(T):
        args = raw_input().split(" ")
        if args[0] == "append":
            L.append(int(args[1]))
        elif args[0] == "insert":
            L.insert(int(args[1]), int(args[2]))
        elif args[0] == "remove":
            L.remove(int(args[1]))
```

```
elif args[0] == "pop":
    L.pop()
elif args[0] == "sort":
    L.sort()
elif args[0] == "reverse":
    L.reverse()
elif args[0] == "print":
    print L
```

Also works.

6 | Add Comment Parent Permalink

dvened 10 months ago

Your formatting is wrong, there are unnecessary indents on every row except for the first one

0 | Add Comment Parent Permalink

NouranSamy96 about a year ago

why do i get an error in the elif statement saying invalid syntax although it's the same line as you ??!!

0 | Add Comment Parent Permalink

varunkmr26 about a year ago

may be you need to check the whitespaces.

0 | Add Comment Parent Permalink

munish01211 about a year ago

now this code helped me to understand this problem. Thank you!

0 | Add Comment Parent Permalink

Max_Vignesh 11 months ago

Thank you but what is the difference between the .strip() and .split() code. Can you please explain this code.

Thanks & Regards

1 | Add Comment Parent Permalink

JXSan 9 months ago

Thank you so much for your clear code! My biggest mistake was not stripping / splitting the input!

- Jonathan

0 | Add Comment Parent Permalink

srriyer about a year ago

Do not use eval -- as a general rule. You almost never have a good reason to. Use `getattr(..)`.

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
l = []
```

```
for _ in range(int(raw_input())):
    line = raw_input().strip()
    if line == "print":
        print l
        continue
    parts = line.split()
    getattr(l, parts[0])(*(map(int, parts[1:])))
```

17 | Add Comment Permalink

tedmiston 9 months ago

I wanted to avoid a long `if-else` block and `eval()`, hence using a dict of lambdas.

I think this is pretty simple and readable.

```
n = int(input())
operations = [input().strip() for _ in range(n)]

list_ = []

commands = {
    'insert': lambda idx, ele: list_.insert(int(idx), int(ele)),
    'print': lambda: print(list_),
    'remove': lambda ele: list_.remove(int(ele)),
    'append': lambda ele: list_.append(int(ele)),
    'sort': lambda: list_.sort(),
    'pop': lambda: list_.pop(),
    'reverse': lambda: list_.reverse(),
}

for operation in operations:
    name, args = [i.strip() for i in (operation + ' ').split(' ', maxspl
it=1)]
    command = commands.get(name)
    command(*args.split())
```

8 | Add Comment Permalink

jdeflaux 8 months ago

That's a neat solution!

0 | Add Comment Parent Permalink

robipolli 6 months ago

don't need to lambda everything

0 | Add Comment Parent Permalink

yatalsingh611 about a month ago

Great

0 | Add Comment Parent Permalink

kaushik_ora 19 days ago

can u please explain this part:

for operation in operations: name, args = [i.strip() for i in (operation + ' ').split(' ', maxsplit=1)] command = commands.get(name) command(*args.split())

0 | Add Comment Parent Permalink

Load more conversations