

Forum:[Hack Forums](#)

Thread: [Parameters, unpacking, learnpythonthehardway](#)

hihihix12 - March 17, 2019, 10:21 pm

I'm learning python by learnpythonthehardway

But i don't understand "script". what can it do and when i delete it, code will break and has an error expected str, bytes or os.PathLike object, not list

here's my code. can you explain it for me? sorry for my bad english

Code from sys import argv

```
script, filename = argv
```

```
txt = open(filename)
```

```
print(f'Here's your filename {filename}')
```

```
print(txt.read())
```

```
print("Type your filename again: ")
```

```
file_again = input("> ")
```

```
print(file_again.read())
```

lockhead - March 18, 2019, 8:57 am

This program won't run, let's break it down:

Code script, filename = argv You can launch any program with parameters or arguments these are passed to the program and saved in argv.

The first argument argv[0] will always be the name of the program itself so in the above code the "script" variable will be the name of the

program and "filename" will be the argument you passed it when running the program. If you delete the script variable you are assigning filename to argv

which is a list of all arguments. And then later on when you try to open a file you give the open function a LIST not a string.

Code #in your terminal you would call myprogram.py and pass "my_text_file.txt" as an argument python myprogram.py my_text_file.txt So script would be "myprogram.py" and filename would be "my_text_file.txt".

The the following code, save it as a .py file and play around with it:

Code from sys import argv

```
for argument in argv:
```

```
    print(argument)
```

Next you try to open a file with the filename that you passed the program as an argument. You should not call the variable "txt" since it not the text in the file, rather a handle to the file, also you should specify in which mode the file should be opened (read, write, append)

```
Code f = open(filename, "r")
```

Next you get some input from the user and store it in file_again. Note that the input() function returns a STRING. In the next line you try to read that string and you will get an AttributeError because you cannot "read" a string, you can only read() a file.

So we need to open a file first with the input the user gave which is stored in file_again.

Also note that you never closed the file you opened, this is bad. To close files you simply call the close function on the file handler.

```

Code f = open(filename, "r")
file_content = f.read()
print(file_content)
f.close()

```

The best way to do file I/O is by using the "with" keyword, it does a lot of the file handling for you. Just be aware that the file will only be available in the following indentation block:

```

Code from sys import argv
#we never use script, and we know that the argument "filename" is the first one right after the
scriptname itself which is always the first so we can:
#script, filename = argv
#get the element at index 1 from our list of arguments, if you didnt pass an argument this will cause
an error
filename = argv[1]
with open(filename, "w") as f:
    file_content = f.read()
#note from here on we won't be able to access the file itself anymore since it is only declared in the
indentation block above, but we got the file contents stored
# if you don't use with open as ... make sure to close the file -> f.close()
print(file_contents)
file_again = input("Type your filename again: ")
with open(file_again, "w") as f:
    print(f.read())

```

hihihix12 - March 18, 2019, 10:21 am

(Mar 18, 2019 08:57 AM)lockhead Wrote: [CITE](#) This program won't run, let's break it down:

Code script, filename = argv You can launch any program with parameters or arguments these are passed to the program and saved in argv.

The first argument argv[0] will always be the name of the program itself so in the above code the "script" variable will be the name of the program and "filename" will be the argument you passed it when running the program. If you delete the script variable you are assigning filename to argv which is a list of all arguments. And then later on when you try to open a file you give the open function a LIST not a string.

Code #in your terminal you would call myprogram.py and pass "my_text_file.txt" as an argument
python myprogram.py my_text_file.txt So script would be "myprogram.py" and filename would be "my_text_file.txt".

The the following code, save it as a .py file and play around with it:

```

Code from sys import argv
for argument in argv:
    print(argument)

```

Next you try to open a file with the filename that you passed the program as an argument. You should not call the variable "txt" since it not the text in the file, rather a handle to the file, also you should specify in which mode the file should be opened (read, write, append)

```

Code f = open(filename, "r")

```

Next you get some input from the user and store it in file_again. Note that the input() function returns a STRING. In the next line you try to read that string and you will get an AttributeError because you cannot "read" a string, you can only read() a file.

So we need to open a file first with the input the user gave which is stored in file_again.

Also note that you never closed the file you opened, this is bad. To close files you simply call the close function on the file handler.

```

Code f = open(filename, "r")
file_content = f.read()
print(file_content)

```

f.close())The best way to do file I/O is by using the "with" keyword, it does a lot of the file handling for you. Just be aware that the file will only be available in the following indentation block:

```
Code from sys import argv
#we never use script, and we know that the argument "filename" is the first one right after the
scriptname itself which is always the first so we can:
#script, filename = argv
#get the element at index 1 from our list of arguments, if you didnt pass an argument this will cause
an error
filename = argv[1]
with open(filename, "w") as f:
    file_content = f.read()
#note from here on we won't be able to access the file itself anymore since it is only declared in the
indentation block above, but we got the file contents stored
# if you don't use with open as ... make sure to close the file -> f.close()
print(file_content)
file_again = input("Type your filename again: ")
with open(file_again, "w") as f:
    print(f.read())thank you very much (y) very details but in pythonthehardway exercise 13 - 15(i'm
learning exercise 15) doesn't talk about argv with "[]" the agrv[] on your code line 5, what it mean?
where should i learn python ? can you tell me the best way to learn it ? i'm the beginner thank you .
```

lockhead - March 18, 2019, 1:53 pm

(Mar 18, 2019 10:21 AM)hihihix12 Wrote: [CITE](#) thank you very much (y) very details but in pythonthehardway exercise 13 - 15(i'm learning exercise 15) doesn't talk about argv with "[]" the agrv[] on your code line 5, what it mean? where should i learn python ? can you tell me the best way to learn it ? i'm the beginner thank you . That is because the variable argv is a 'list'. To get a variables type you can call the type function on it like so:

```
Code from sys import argv
print(type(argv))
#output: <class 'list'>You can read more on lists here.
You can address each element in a list by using an index (note: computers start counting at 0):
```

```
Code #initialize a list
my_list = [1,2,3,4]
print(my_list)
#output [1,2,3,4]
my_list[0] = 55
#elemt in my_list at the index 0 equals 55
print(my_list)
#output: [55,2,3,4]
In your code they used a shortcut to unpack the list:
Code my_list = [1,2]
a,b = my_list
print(a)
print(b)
#output: 1
#output: 2
#its the same as saying
a = my_list[0]
b = my_list[1]
```

hihihi12 - *March 19, 2019, 5:59 am*

thanks you lockhead. i understand it now. thanks you