MyEduSolve X Kampus Merdeka

Perform Unit Testing & Operation using Modules and Tools

Pandas Group
Data Science Track B

Ajie Prasetya
Dewa Ayu Rai I.M
M. Ramadhoni

Vivi Indah Fitriani
Firdausi Nurus Sa'idah

Pandas Group

Material Discussion

01 Argumen and Parameter
- Variables are iterable
- Variables are prefix
- Function Parameter Arrangement
- Anonymous Function (lambda())

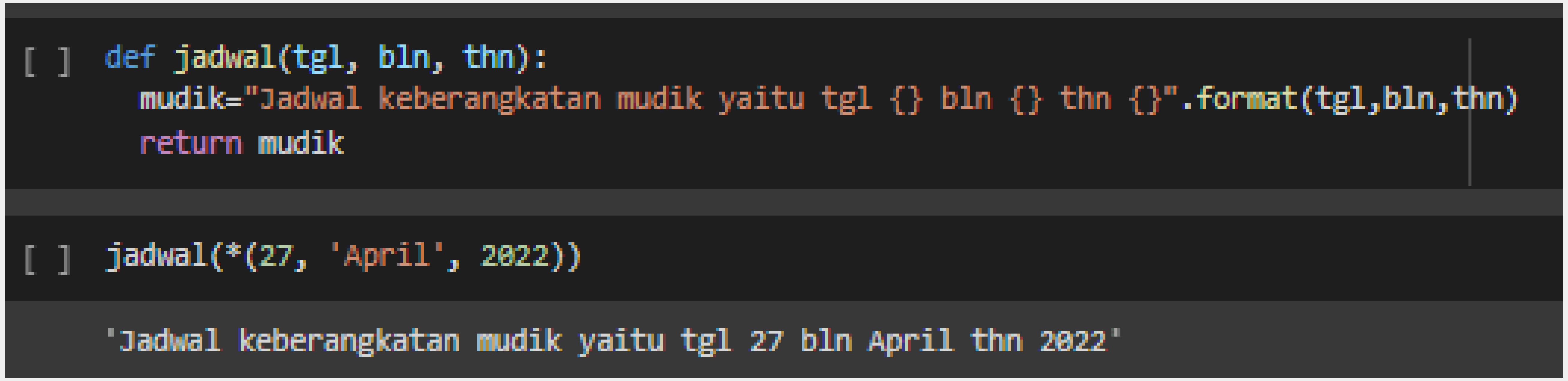
02 Method
- @classmethod
- @staticmethod

03 Open File, Read File, Close File and Delete File

04 Import io, Os Path, Pydoc, Unit Test, sys, OS, Random Library, Math and Datetime

Argumen and Parameter

Variables are iterable, must be preceded by a sign (*)

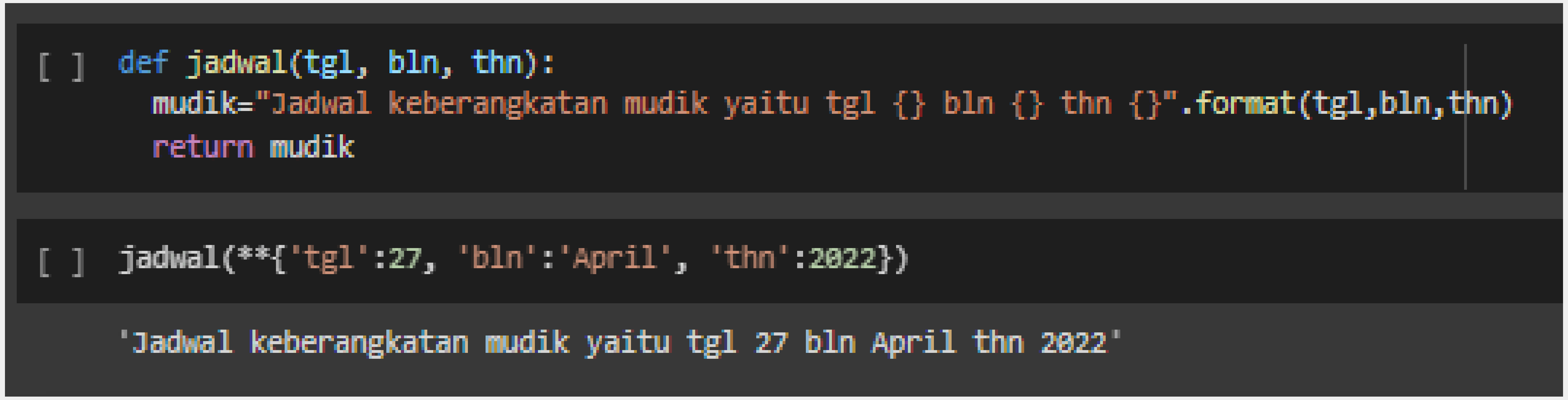


```
[ ]  def jadwal(tgl, bln, thn):
       mudik="Jadwal keberangkatan mudik yaitu tgl {} bln {} thn {}".format(tgl,bln,thn)
       return mudik
```

```
[ ]  jadwal(*(27, 'April', 2022))
```

```
'Jadwal keberangkatan mudik yaitu tgl 27 bln April thn 2022'
```

Variables are prefix, must be preceded by a sign (**)

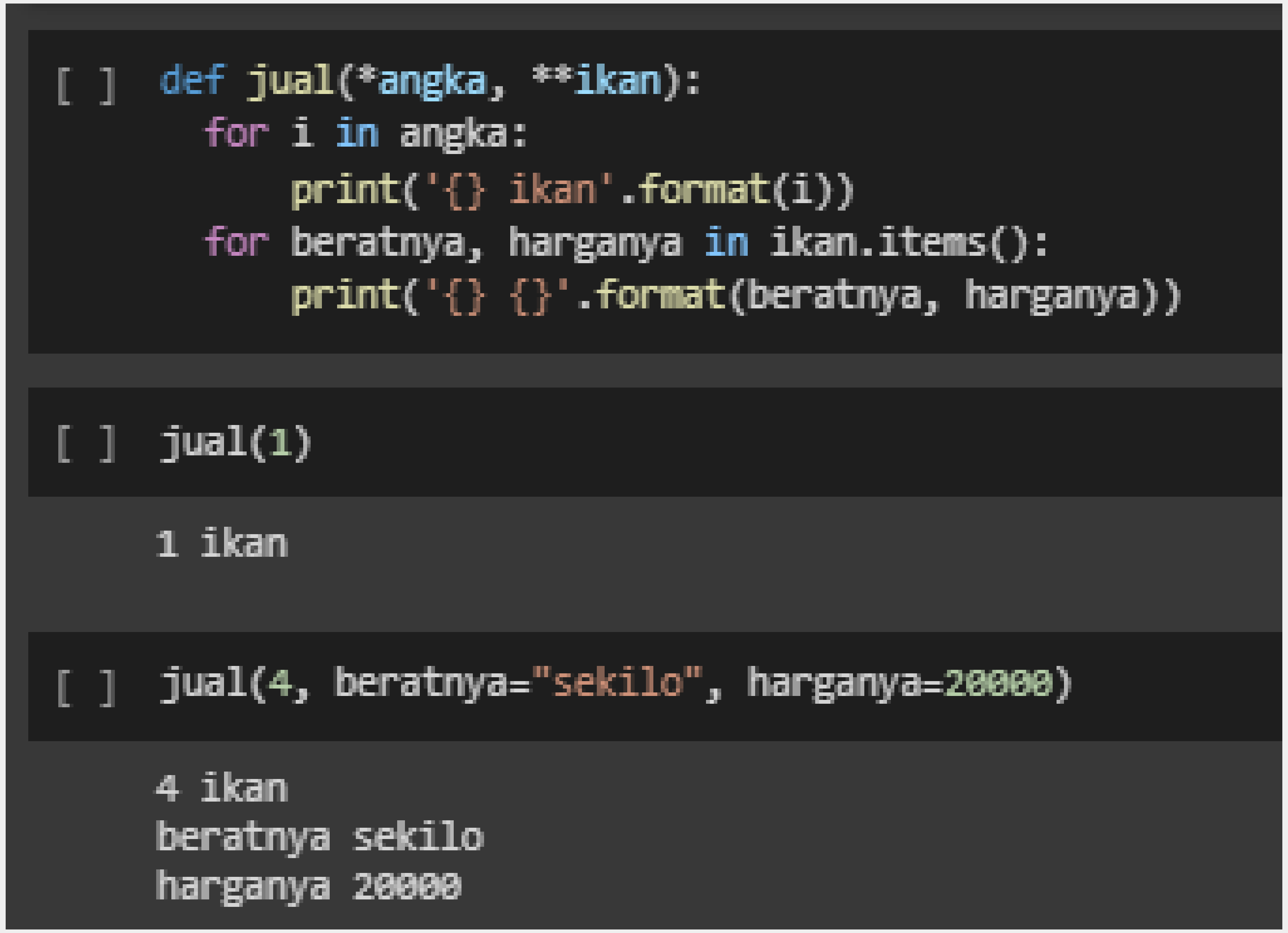


```
[ ]  def jadwal(tgl, bln, thn):
       mudik="Jadwal keberangkatan mudik yaitu tgl {} bln {} thn {}".format(tgl,bln,thn)
       return mudik
```

```
[ ]  jadwal(**{'tgl':27, 'bln':'April', 'thn':2022})
```

```
     'Jadwal keberangkatan mudik yaitu tgl 27 bln April thn 2022'
```

Function Parameter Arrangement



```
[ ]  def jual(*angka, **ikan):
       for i in angka:
           print('{} ikan'.format(i))
       for beratnya, harganya in ikan.items():
           print('{} {}'.format(beratnya, harganya))
```

```
[ ]  jual(1)
```

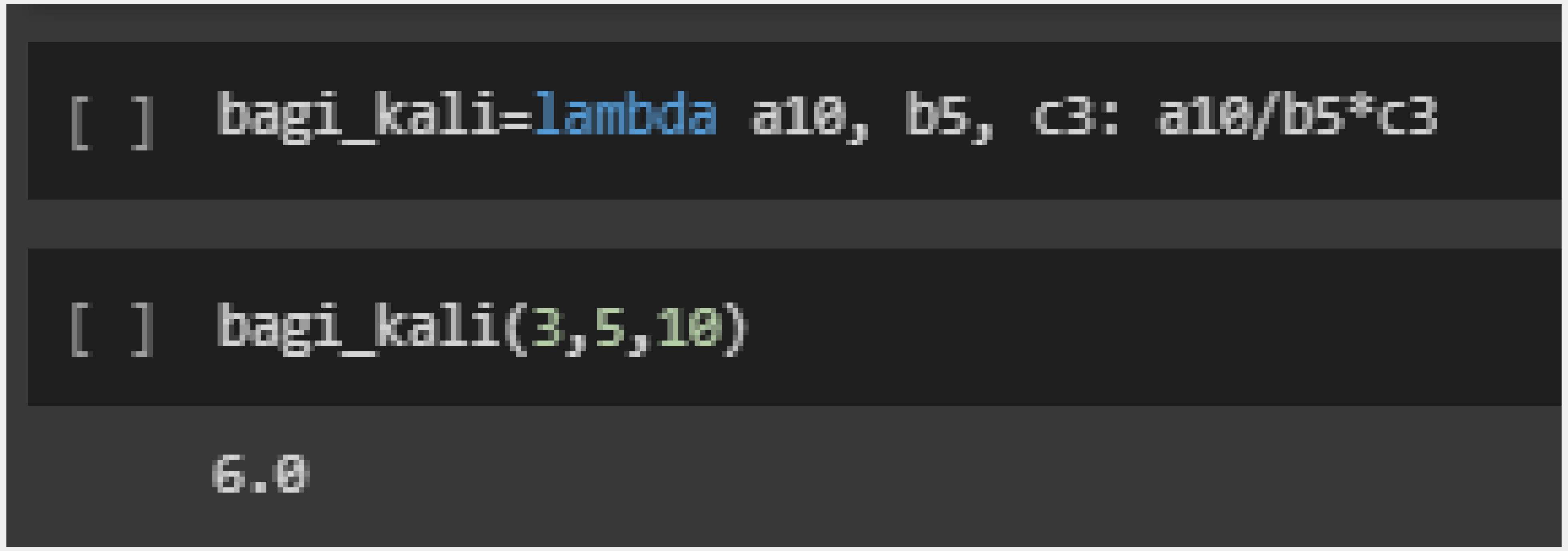
```
1 ikan
```

```
[ ]  jual(4, beratnya="sekilo", harganya=20000)
```

```
4 ikan
beratnya sekilo
harganya 20000
```

Anonymous Function (lambda())

In the regular function we use the def keyword to create it, while in this anonymous function we use the lambda keyword. Therefore, this anonymous function is also known as a lambda function. It is called an anonymous function because it is not given a name at the time it is defined.



```
[ ] bagi_kali=lambda a10, b5, c3: a10/b5*c3
```

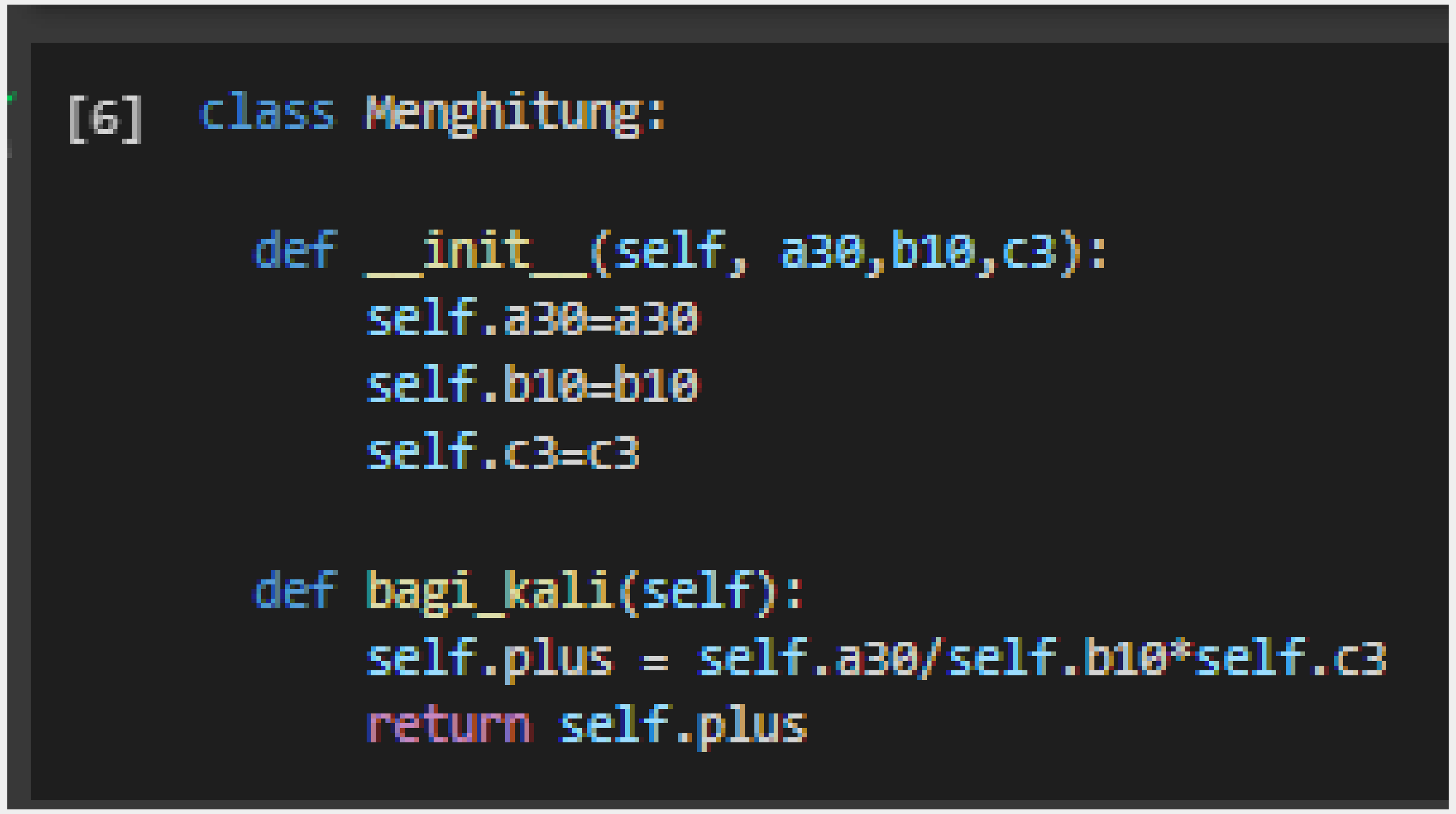
```
[ ] bagi_kali(3,5,10)
```

```
6.0
```

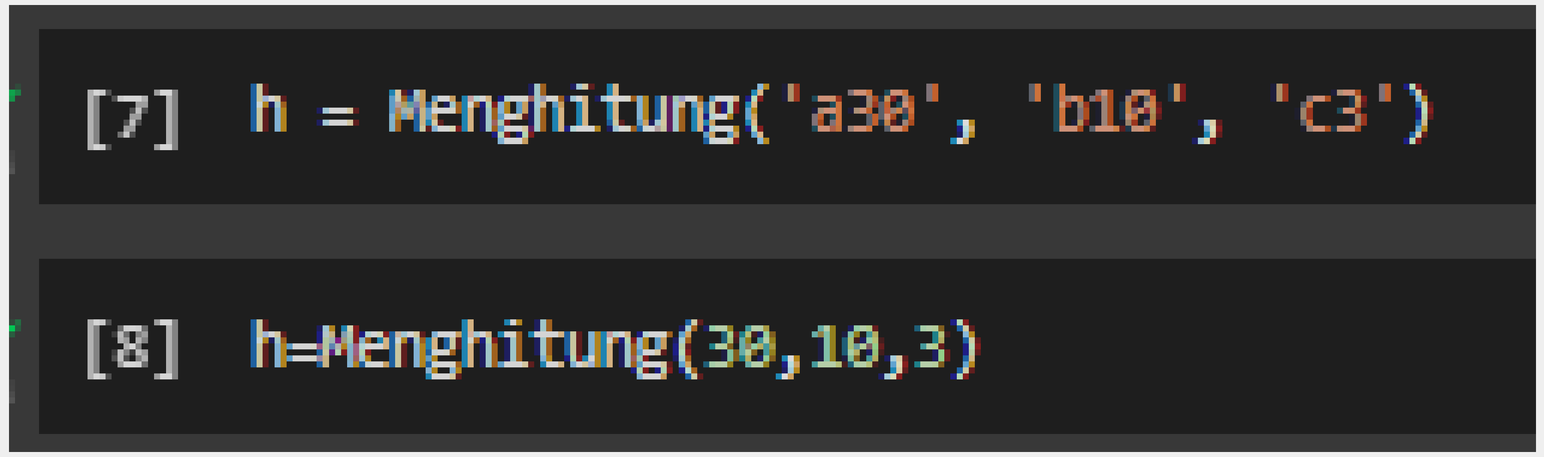
»

Method

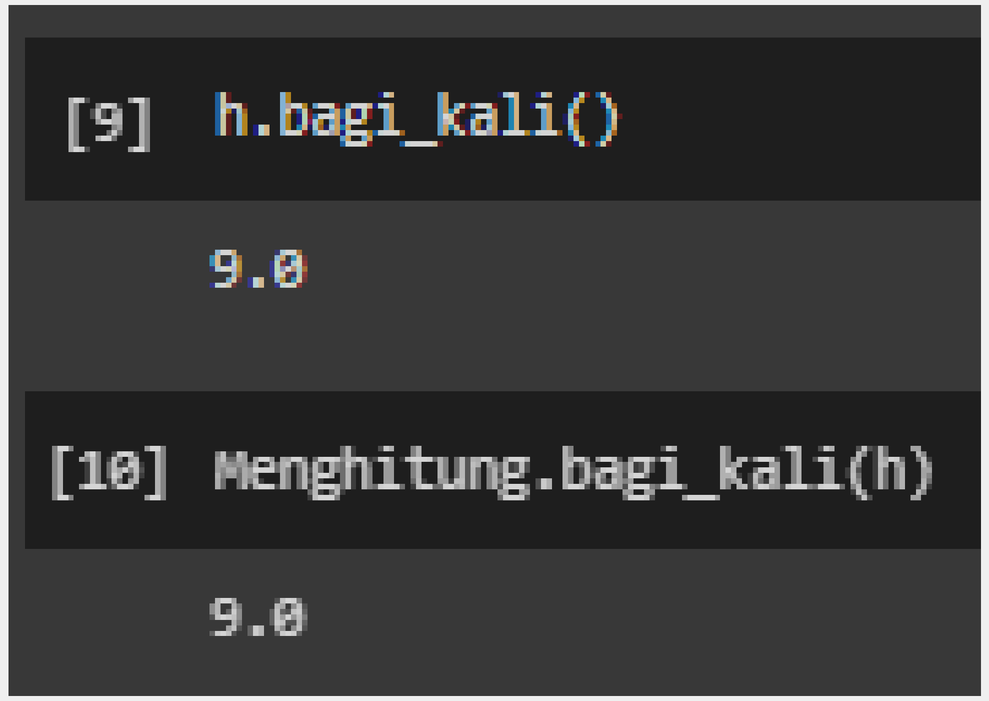
In general, a method is a special function that belongs to an object, namely the result of instantiation of a class. for a method is actually passed an object or the result of an instance of the class as its first argument



```
[6]  class Menghitung:

         def __init__(self, a30,b10,c3):
             self.a30=a30
             self.b10=b10
             self.c3=c3

         def bagi_kali(self):
             self.plus = self.a30/self.b10*self.c3
             return self.plus
```



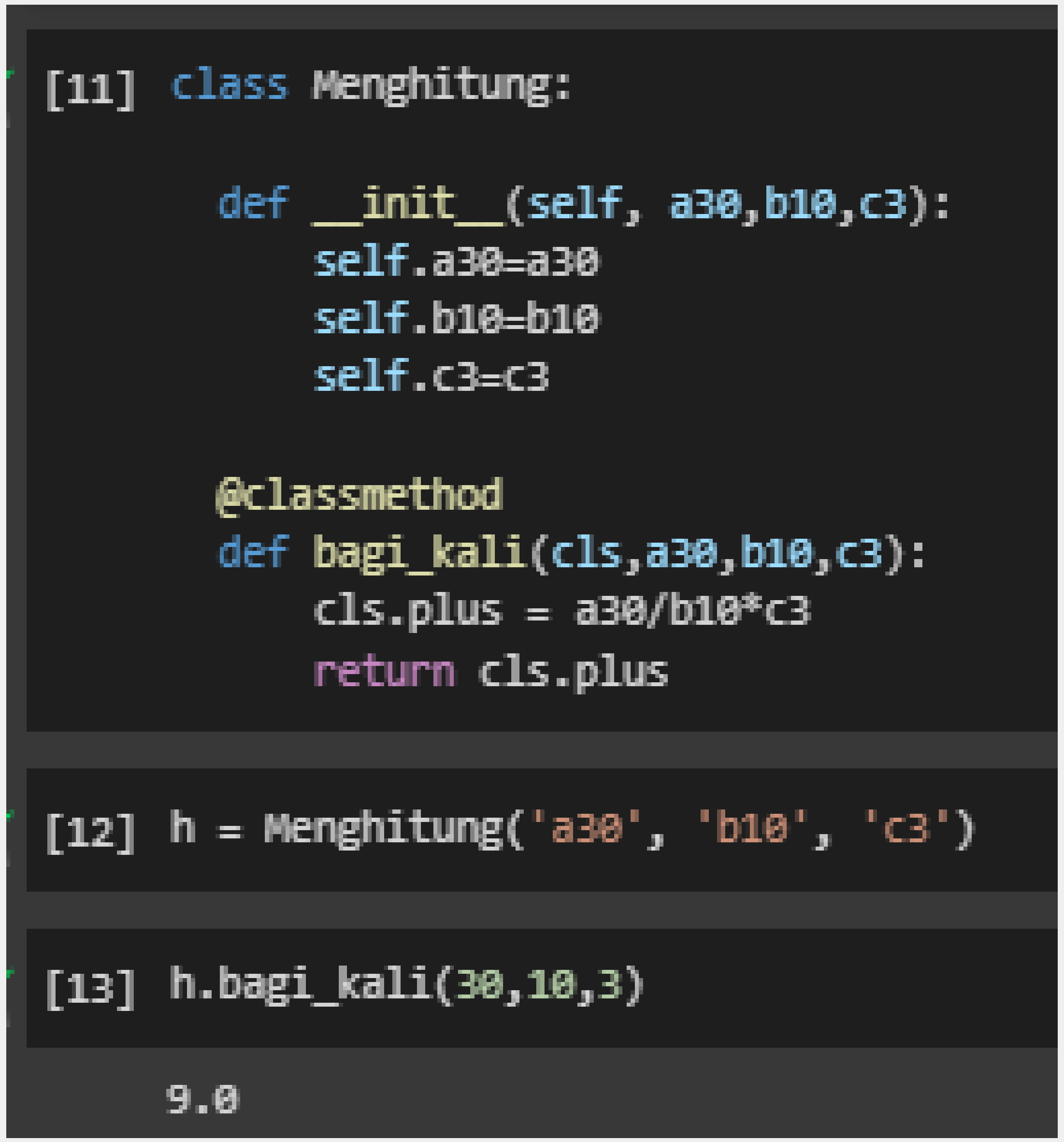
```
[7]  h = Menghitung('a30', 'b10', 'c3')

[8]  h=Menghitung(30,10,3)
```



```
[9]  h.bagi_kali()

     9.0

[10] Menghitung.bagi_kali(h)

     9.0
```

@classmethod

@classmethod means: when this method is called, we pass the class as the first argument instead of an instance of that class (as we usually do with methods). This means you can use a class and its properties inside that method instead of a specific instance



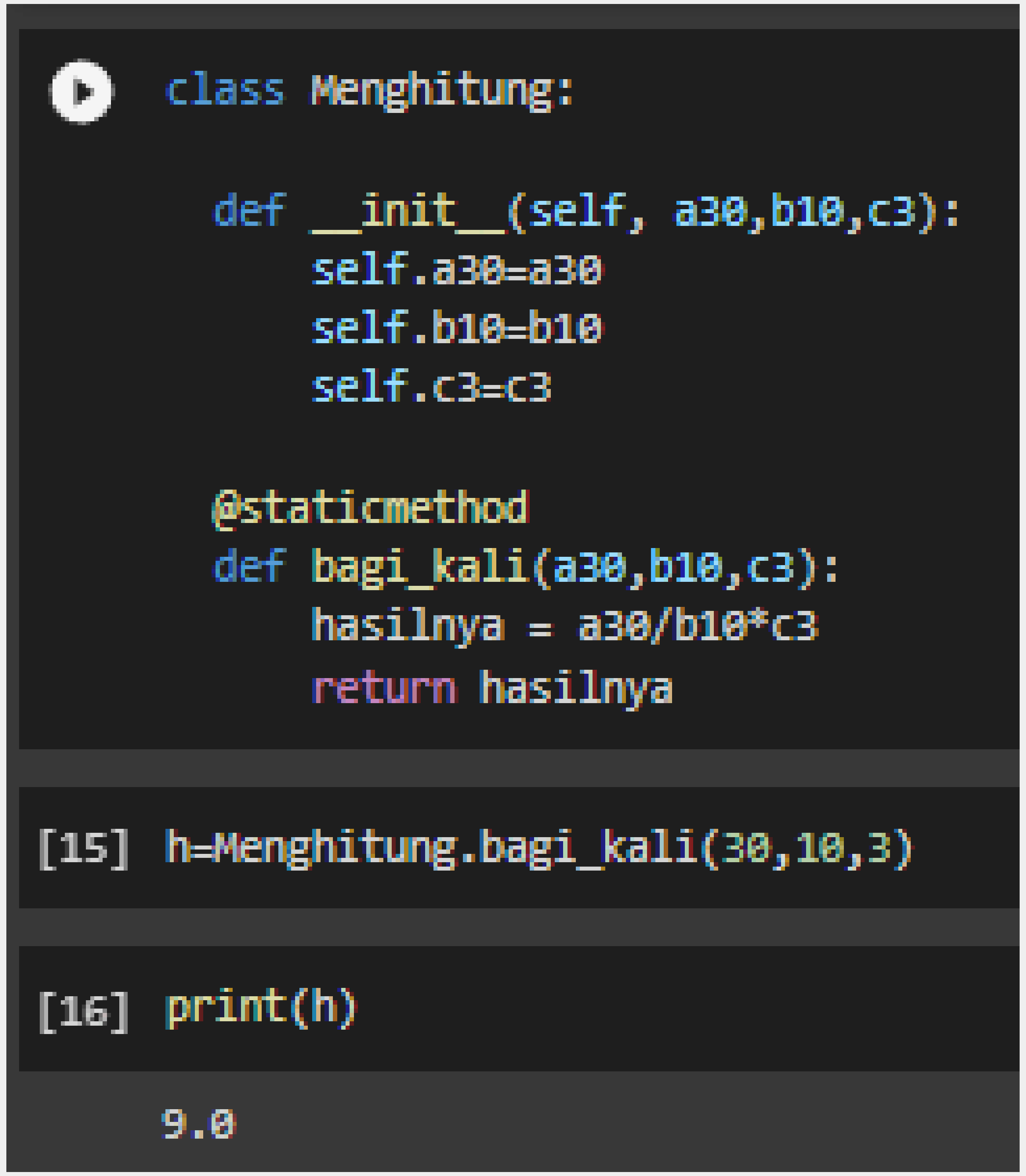
```
[11] class Menghitung:

         def __init__(self, a30,b10,c3):
             self.a30=a30
             self.b10=b10
             self.c3=c3

         @classmethod
         def bagi_kali(cls,a30,b10,c3):
             cls.plus = a30/b10*c3
             return cls.plus
```

```
[12] h = Menghitung('a30', 'b10', 'c3')
```

```
[13] h.bagi_kali(30,10,3)

     9.0
```

@staticmethod

@staticmethod means: when this method is called, we don't pass an instance of the class to it (as we usually do with methods). This means you can put a function inside a class but you can't access an instance of that class (this is useful when your method doesn't use an instance).



```
class Menghitung:

  def __init__(self, a30,b10,c3):
      self.a30=a30
      self.b10=b10
      self.c3=c3

  @staticmethod
  def bagi_kali(a30,b10,c3):
      hasilnya = a30/b10*c3
      return hasilnya
```

```
[15] h=Menghitung.bagi_kali(30,10,3)
```

```
[16] print(h)
```

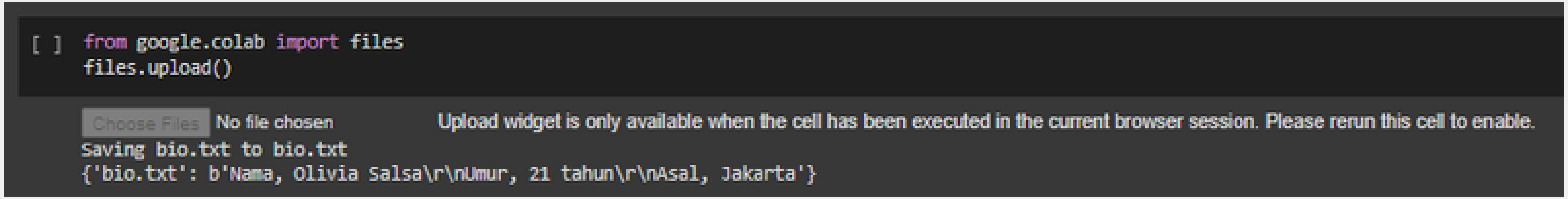
```
9.0
```

»

Open, Read, Close a File

Open File

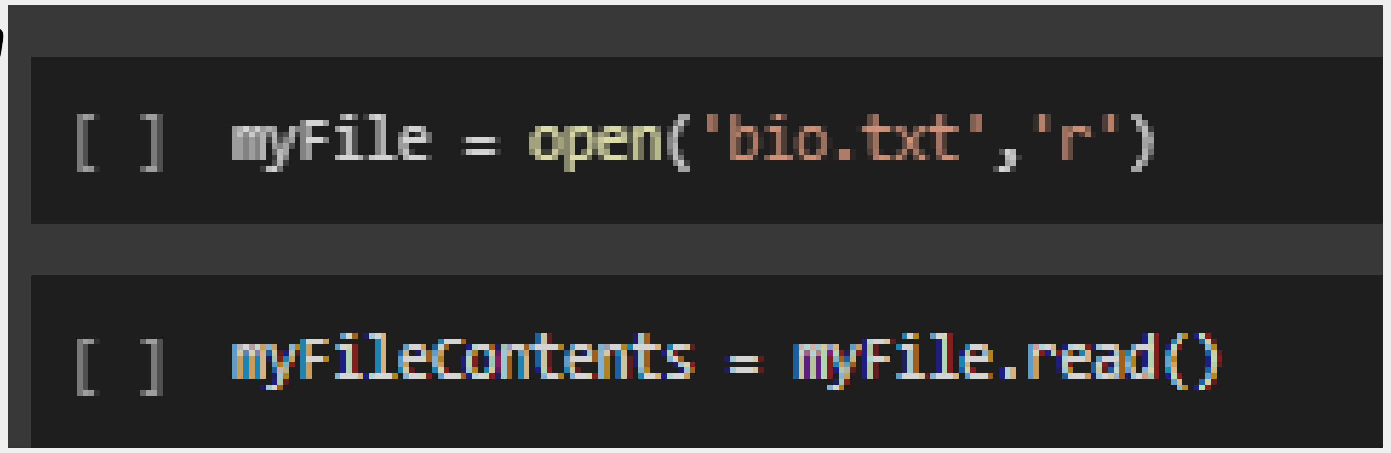
The open() function returns a file object, which has a read() method for reading the content of the file:



```
[ ]  from google.colab import files
     files.upload()
```

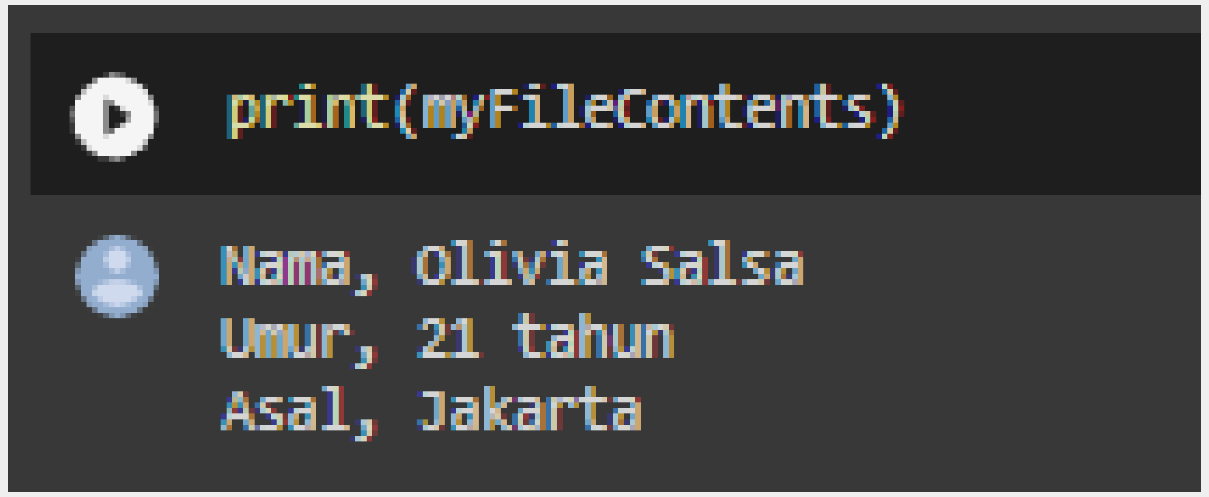
Choose Files No file chosen   Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving bio.txt to bio.txt
{'bio.txt': b'Nama, Olivia Salsa\r\nUmur, 21 tahun\r\nAsal, Jakarta'}
```



```
[ ]  myFile = open('bio.txt','r')
```

```
[ ]  myFileContents = myFile.read()
```

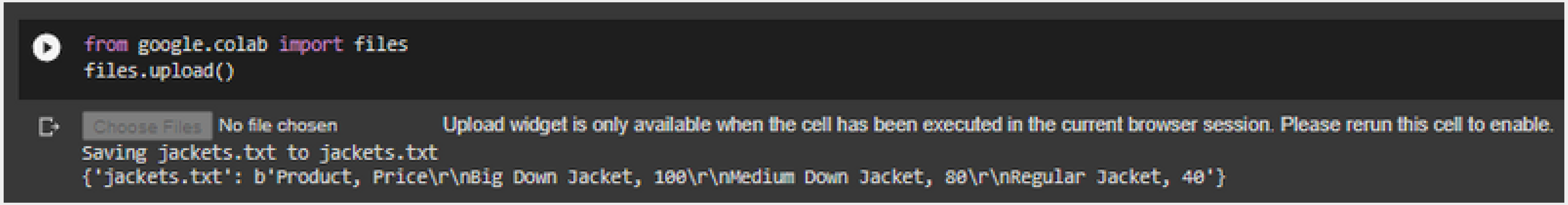


```
print(myFileContents)
```

```
Nama, Olivia Salsa
Umur, 21 tahun
Asal, Jakarta
```

Read File

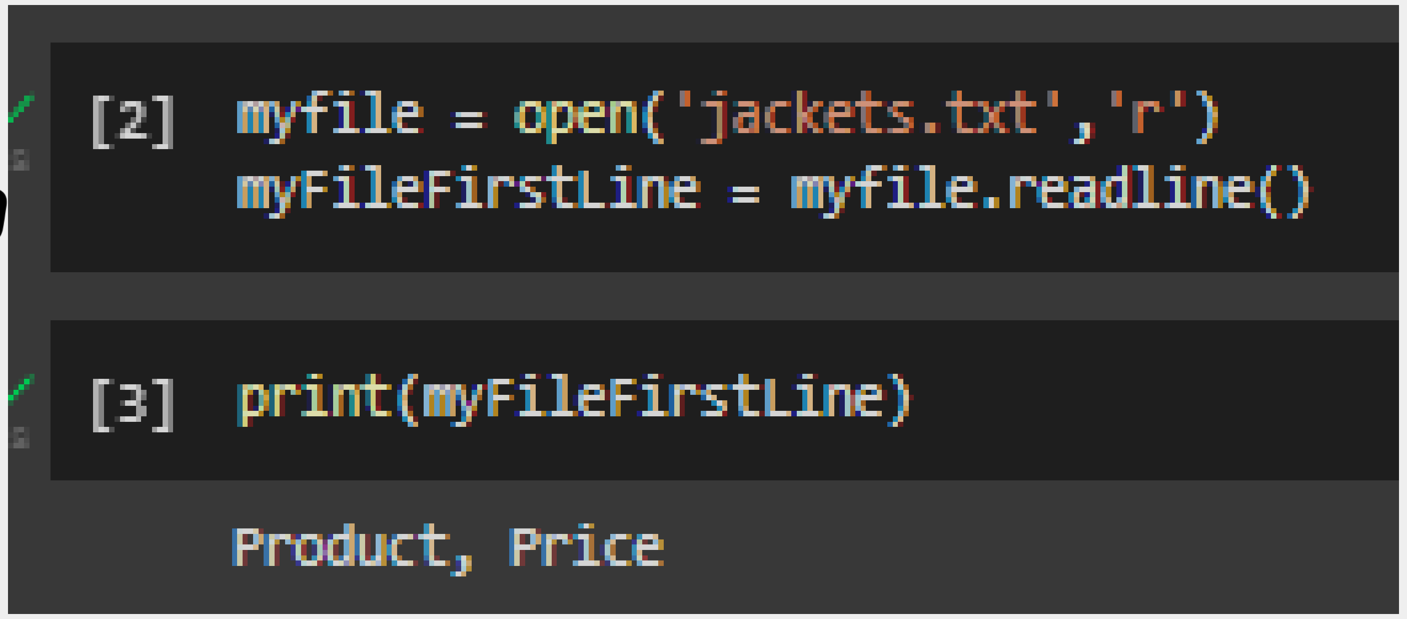
You can return one line by using the readline() method:



```
from google.colab import files
files.upload()
```

Choose Files No file chosen   Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

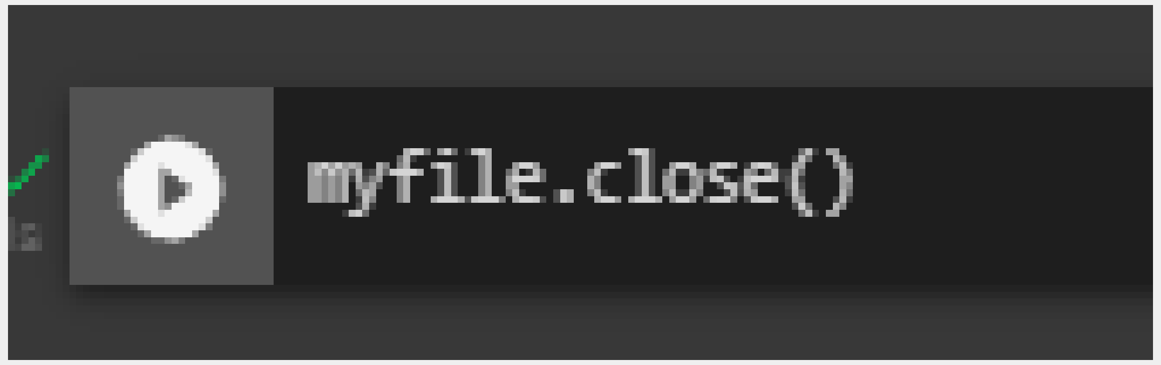
```
Saving jackets.txt to jackets.txt
{'jackets.txt': b'Product, Price\r\nBig Down Jacket, 100\r\nMedium Down Jacket, 80\r\nRegular Jacket, 40'}
```



```
[2]  myfile = open('jackets.txt','r')
     myFileFirstLine = myfile.readline()

[3]  print(myFileFirstLine)

     Product, Price
```

Close File

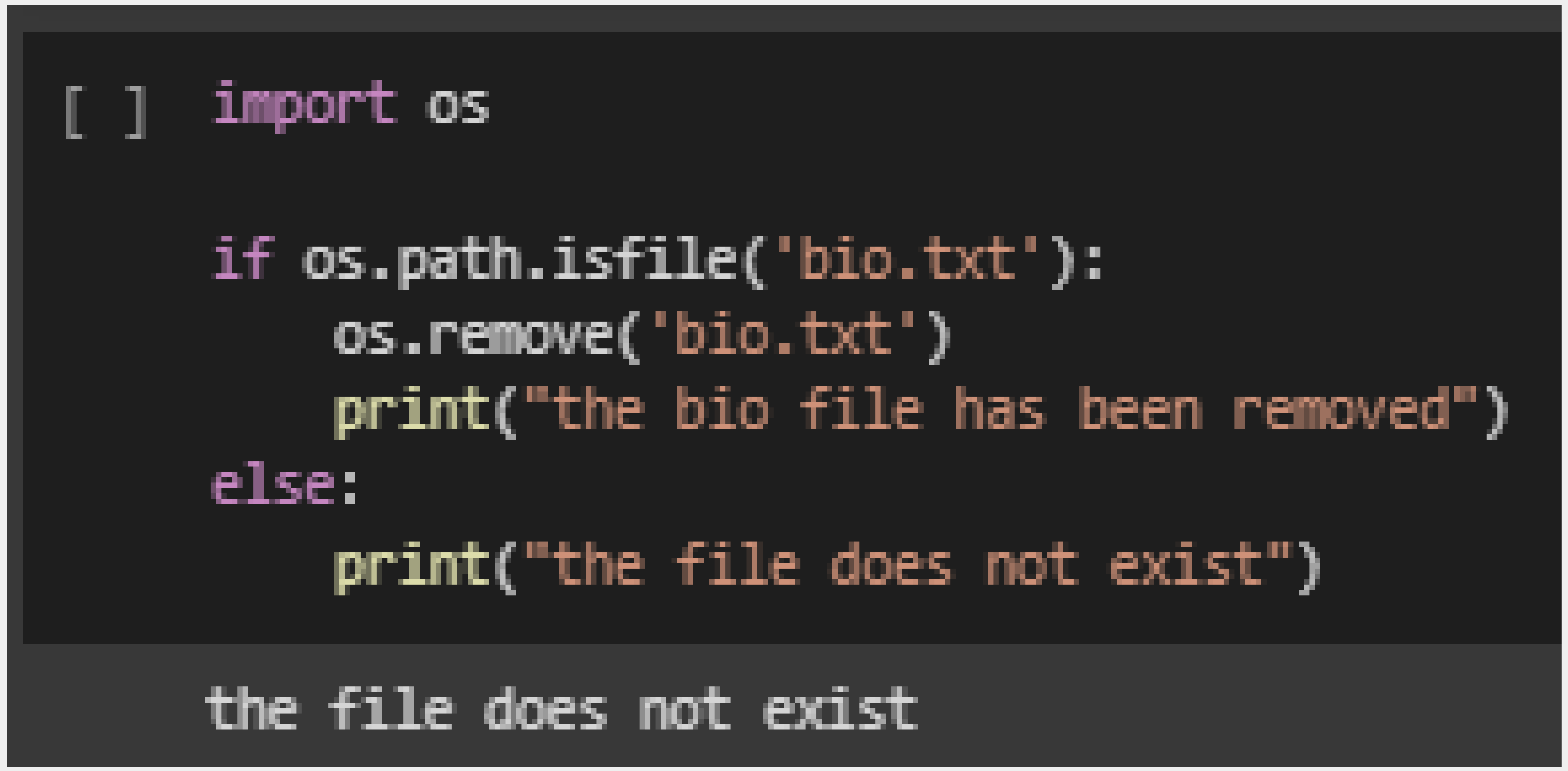
Python file method close() closes the opened file. A closed file cannot be read or written any more. Any operation, which requires that the file be opened will raise a ValueError after the file has been closed



```
myfile.close()
```

Delete File

To delete a file, you must import the OS module, and run its os.remove() function:



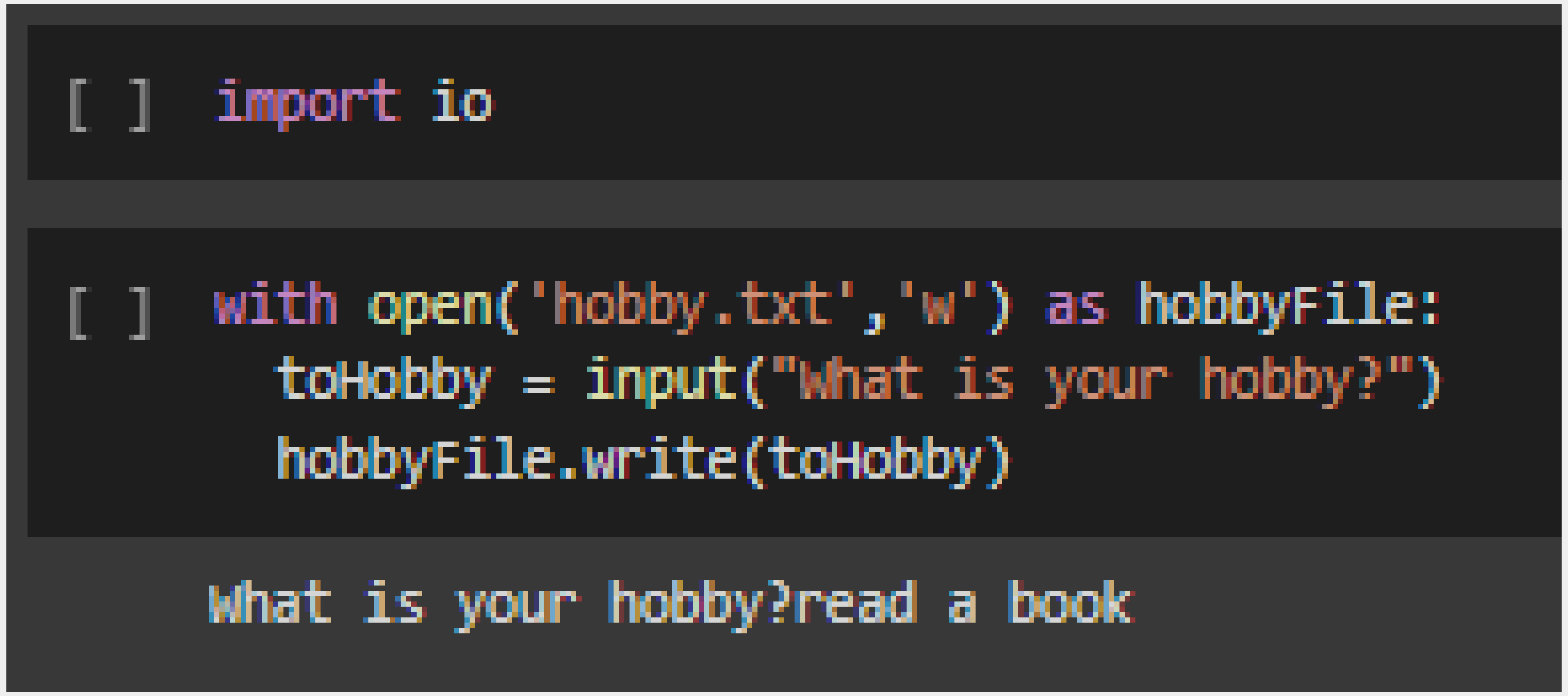
```
[ ]  import os

     if os.path.isfile('bio.txt'):
         os.remove('bio.txt')
         print("the bio file has been removed")
     else:
         print("the file does not exist")
```

```
     the file does not exist
```

Import io

Python io module allows us to manage the file-related input and output operations.

To import the io module, we can do the following:



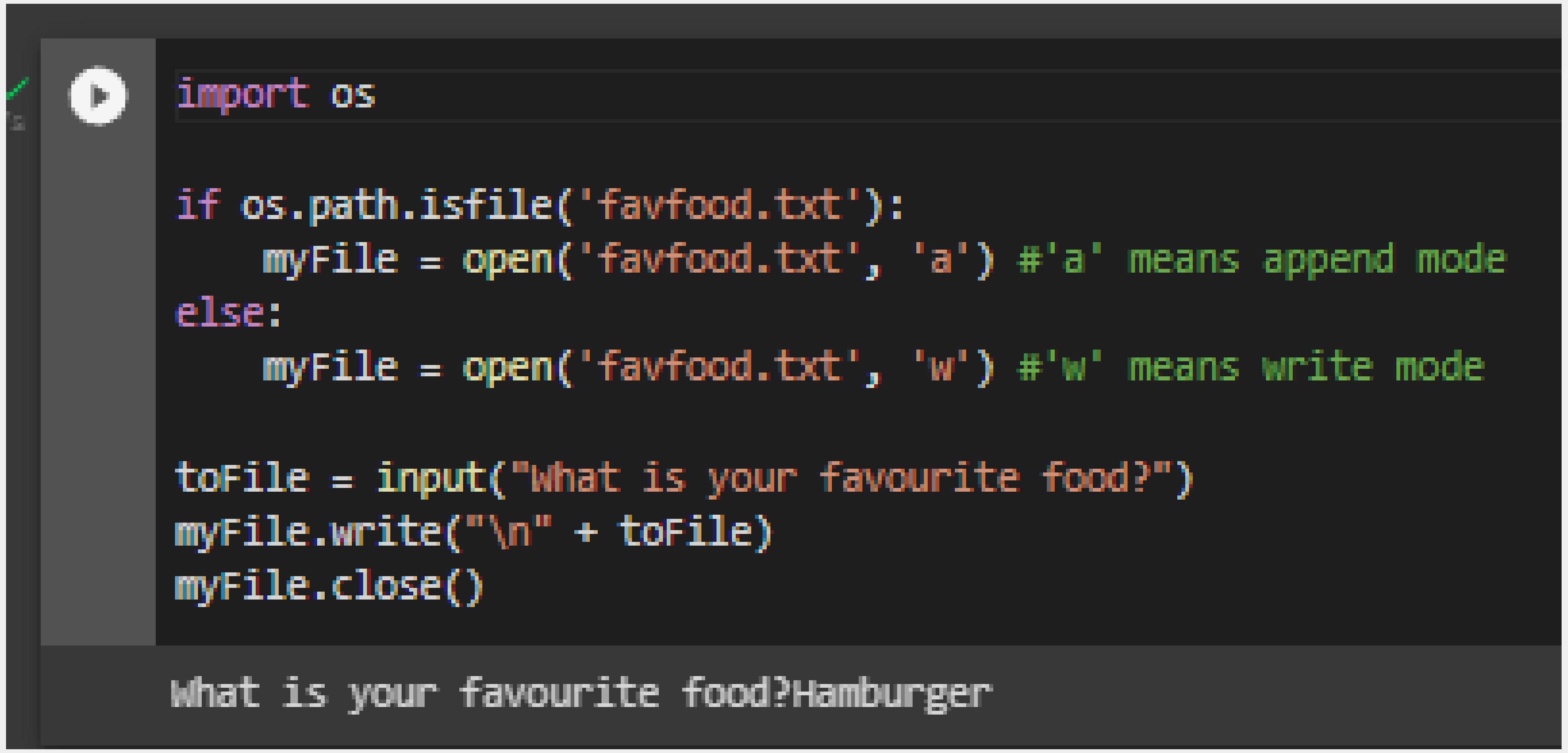
```
[ ]  import io
```

```
[ ]  with open('hobby.txt','w') as hobbyFile:
       toHobby = input("What is your hobby?")
       hobbyFile.write(toHobby)
```

```
What is your hobby?read a book
```

Os Path

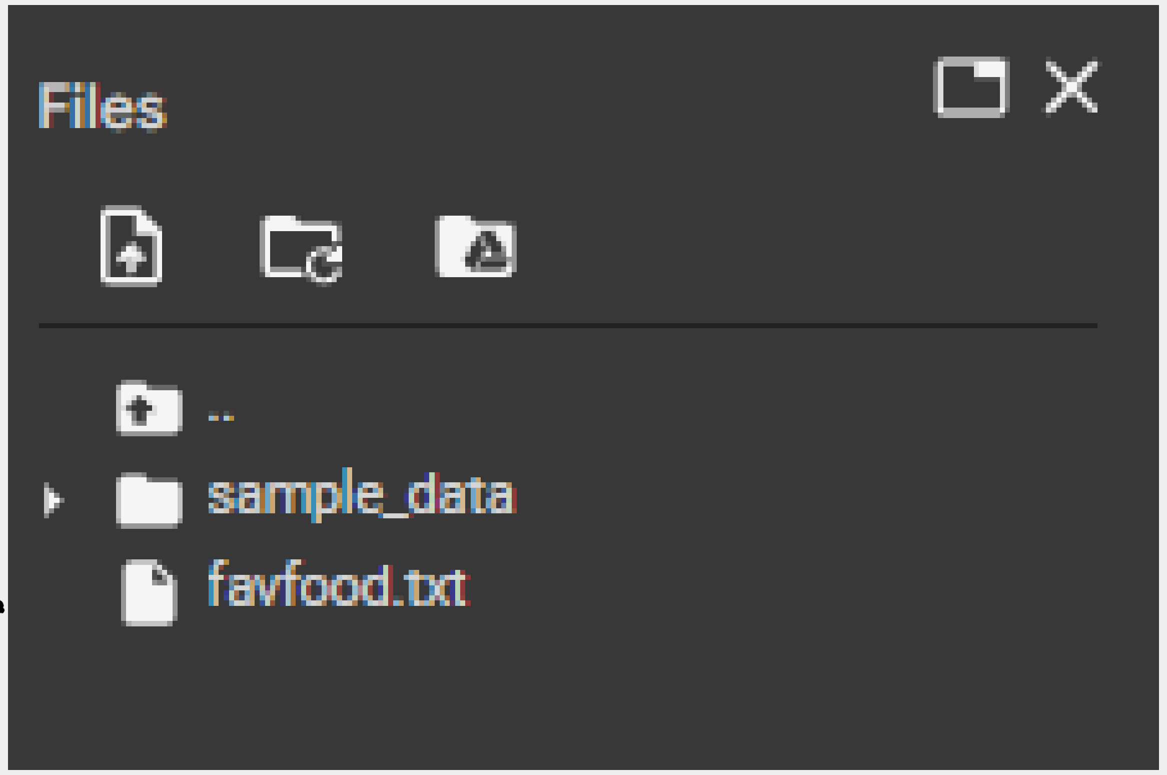
- The os.path module contains functions that deal with long filenames (pathnames) in various ways.
- To use this module, import the os module, and access this module as os.path

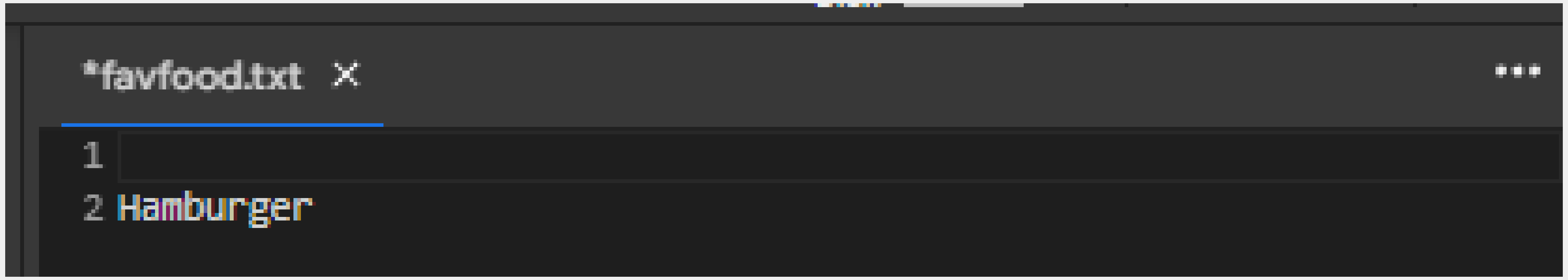


```
import os

if os.path.isfile('favfood.txt'):
    myFile = open('favfood.txt', 'a') #'a' means append mode
else:
    myFile = open('favfood.txt', 'w') #'w' means write mode

toFile = input("What is your favourite food?")
myFile.write("\n" + toFile)
myFile.close()
```

```
What is your favourite food?Hamburger
```

Output Os Path

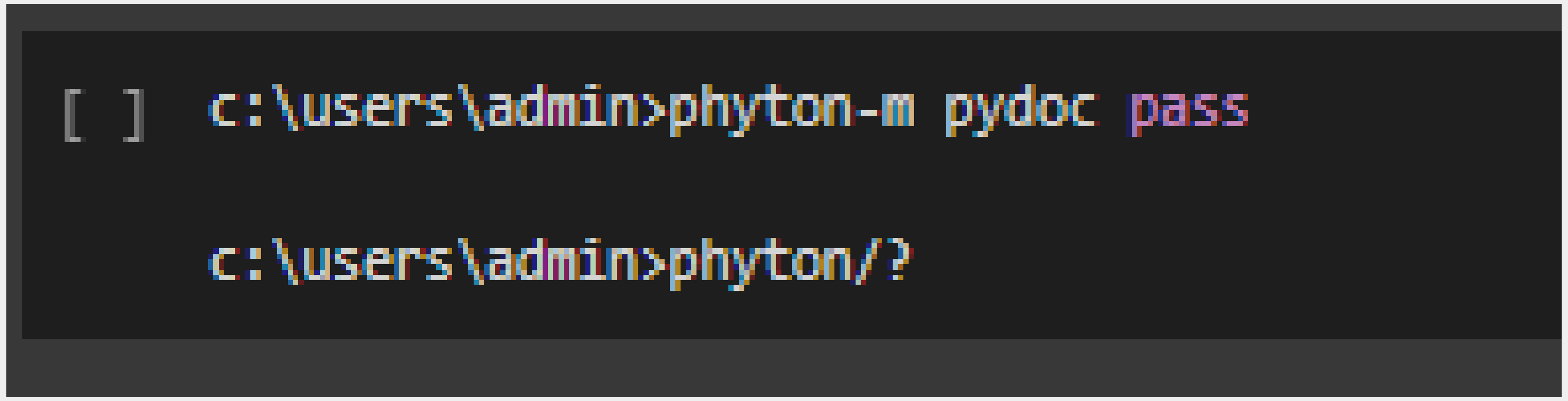




```

Hamburger
```

Pydoc

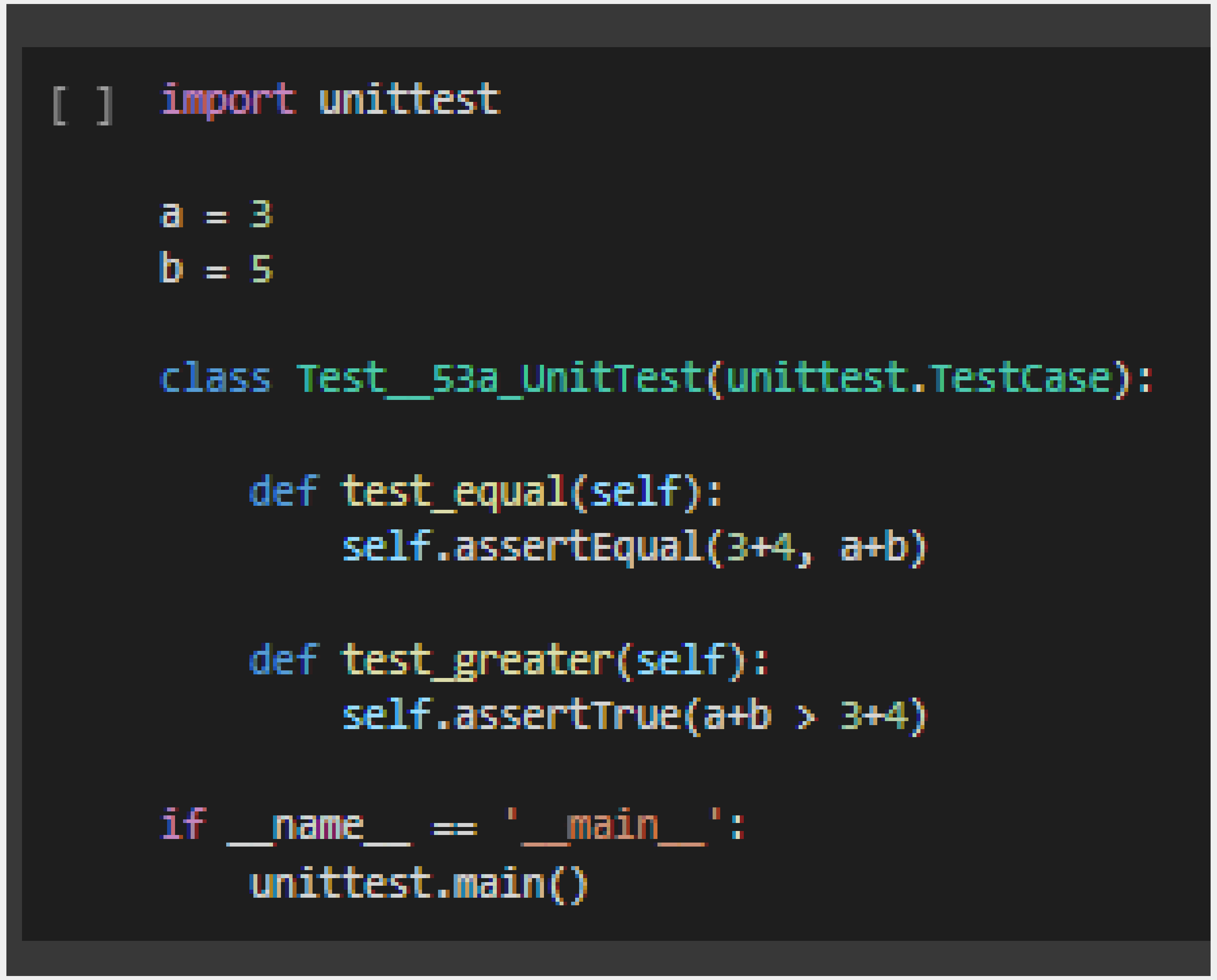
- In Python, the pydoc module is a document generator.
- It can be executed using the pydoc command (or, equivalently, python -m pydoc), followed by the name of the module or script file



```
[ ]  c:\users\admin>phyton-m pydoc pass

     c:\users\admin>phyton/?
```

Unit Test

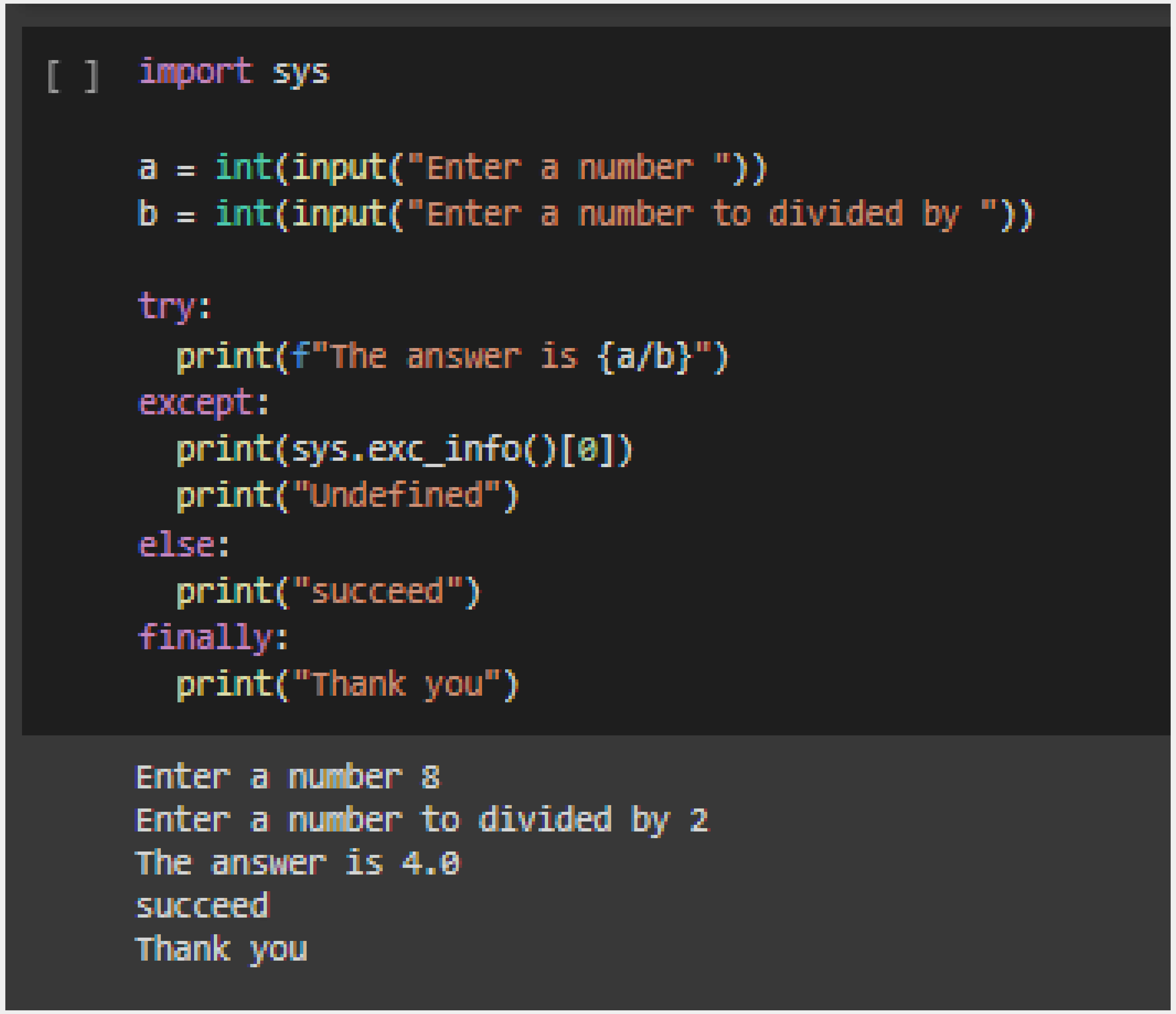
Unittest is a software testing process that ensures every unit/function of the program is tested. **Unittests can be run in vscode and are usually .py files.**



```
[ ] import unittest

    a = 3
    b = 5

    class Test__53a_UnitTest(unittest.TestCase):

        def test_equal(self):
            self.assertEqual(3+4, a+b)

        def test_greater(self):
            self.assertTrue(a+b > 3+4)

    if __name__ == '__main__':
        unittest.main()
```

sys

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

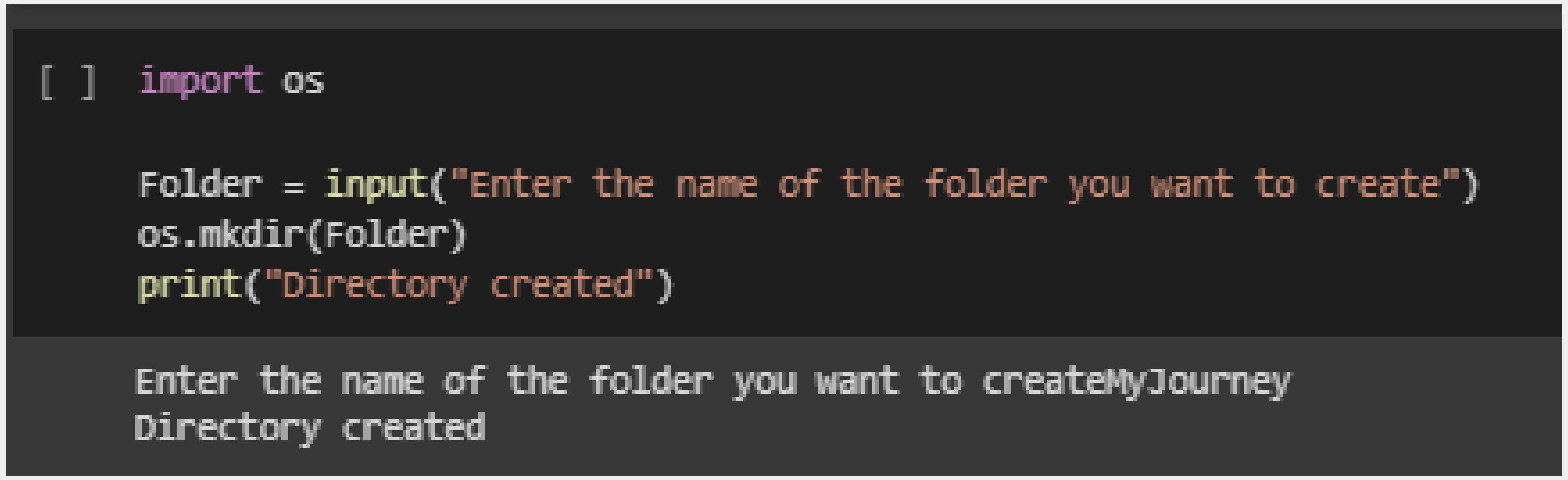


```
[ ] import sys

    a = int(input("Enter a number "))
    b = int(input("Enter a number to divided by "))

    try:
      print(f"The answer is {a/b}")
    except:
      print(sys.exc_info()[0])
      print("Undefined")
    else:
      print("succeed")
    finally:
      print("Thank you")
```

```
Enter a number 8
Enter a number to divided by 2
The answer is 4.0
succeed
Thank you
```

os

This module provides functions for interacting with the operating system

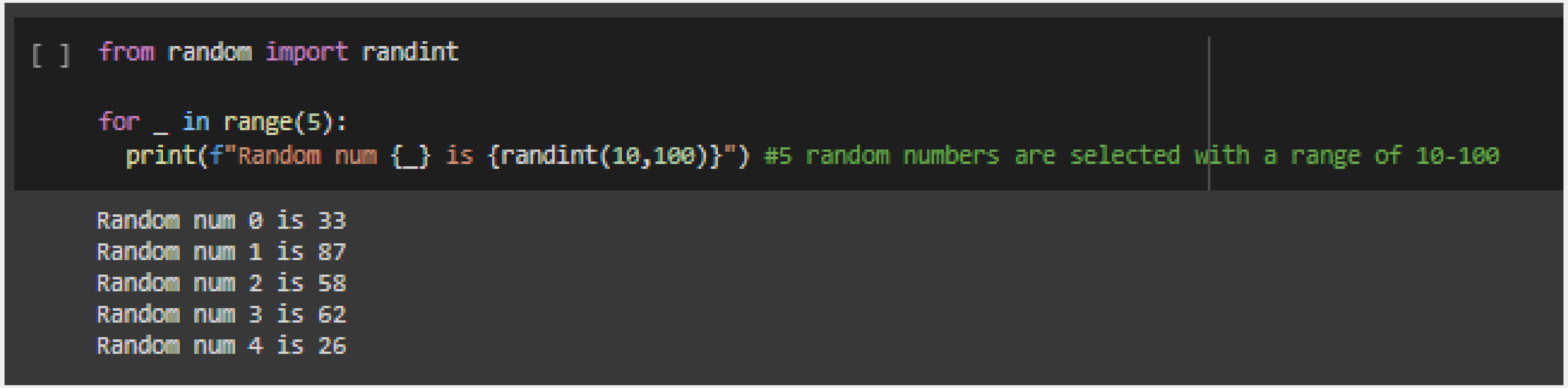


```
[ ]  import os

     Folder = input("Enter the name of the folder you want to create")
     os.mkdir(Folder)
     print("Directory created")
```

```
Enter the name of the folder you want to createMyJourney
Directory created
```

Random Library

This module is an in-built module of Python which is used to generate random numbers.

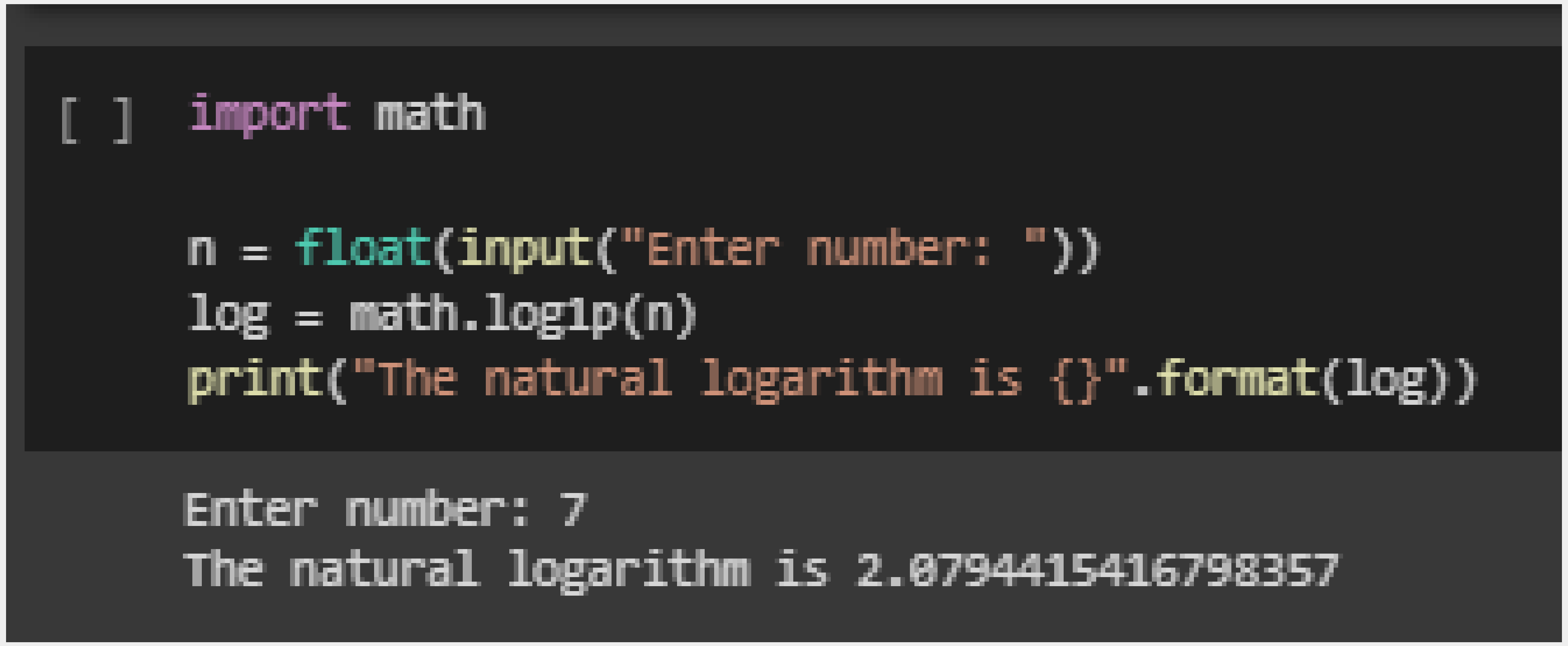


```
[ ]  from random import randint

     for _ in range(5):
       print(f"Random num {_} is {randint(10,100)}") #5 random numbers are selected with a range of 10-100
```

```
     Random num 0 is 33
     Random num 1 is 87
     Random num 2 is 58
     Random num 3 is 62
     Random num 4 is 26
```

Math

This module provides access to the mathematical functions was defined.

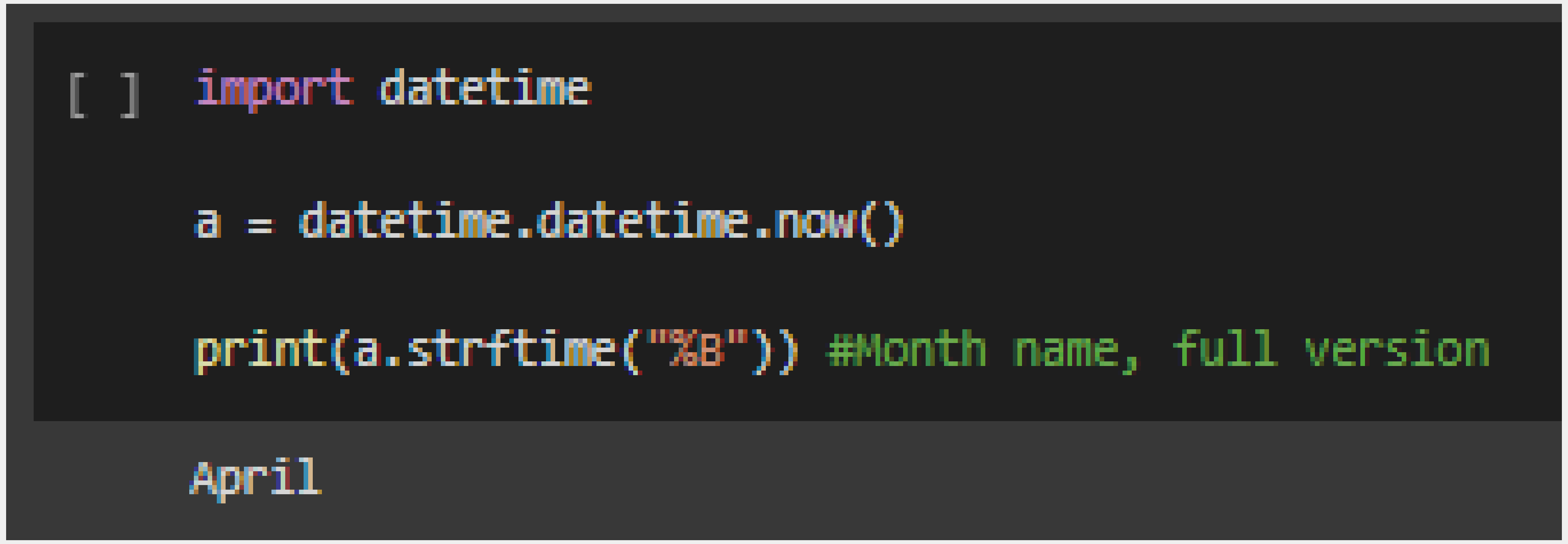


```
[ ]  import math

     n = float(input("Enter number: "))
     log = math.log1p(n)
     print("The natural logarithm is {}".format(log))
```

```
Enter number: 7
The natural logarithm is 2.0794415416798357
```

Datetime

The datetime module has many methods to return information about the date object.



```
import datetime

a = datetime.datetime.now()

print(a.strftime("%B")) #Month name, full version
```

```
April
```

Thank You