MyEduSolve

Kampus Merdeka INDONESIA JAYA

HELLO WORLD, READY TO LEARNING BASIC PYTHON?

Can we modify and process data in python?

Matplotlib Group - Data Science Track B

Python Data Structure

MATPLOTLIB GROUP

1. Tobias Mikha Sulistiyo
2. Daud Ibadurahman
3. Putri Reghina Hilmi Prasati
4. Sari Yuliastuti
5. Fitri Alfaqrina

OUTLINES

- Input and output on python
- Dynamic typing in python
- Number, character, and string transformation
- Changing String Element
- String Checking
- String Formatting
- Operation String, List, Set
- Operator,Operand,and Expression
- Conditional expression

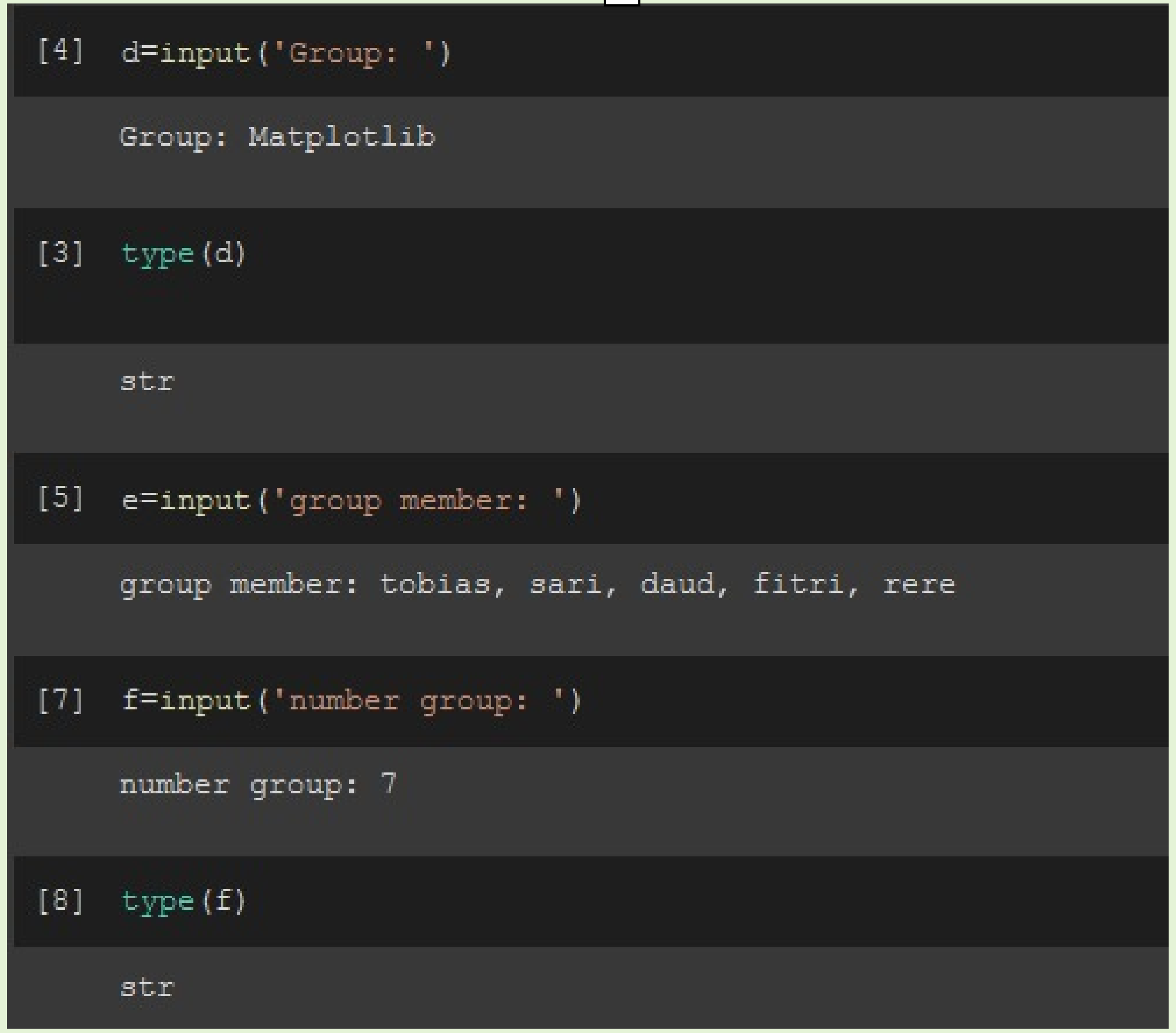
PART 1

Input and Output in the Python

Output

PART 1

The input() function allows user to input. The output of the input() result is to have a string data type



```
[4]  d=input('Group: ')

     Group: Matplotlib

[3]  type(d)

     str

[5]  e=input('group member: ')

     group member: tobias, sari, daud, fitri, rere

[7]  f=input('number group: ')

     number group: 7

[8]  type(f)

     str
```

PART 2

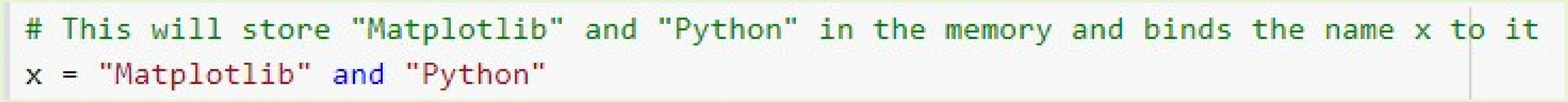
Dynamic Typing in Python

Dynamic Typing

PART 2

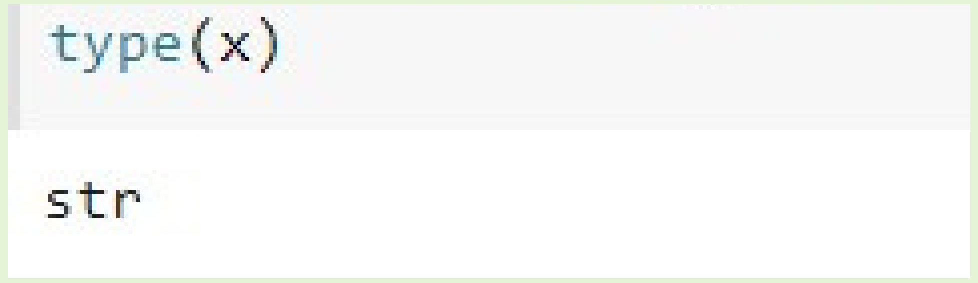
A language is dynamically typed if the type of a variable is interpreted at runtime.

Example:



```
# This will store "Matplotlib" and "Python" in the memory and binds the name x to it
x = "Matplotlib" and "Python"
```

If x never runs, so it's never type checked.
If x is evaluated, after it runs, type of x will be str (string)



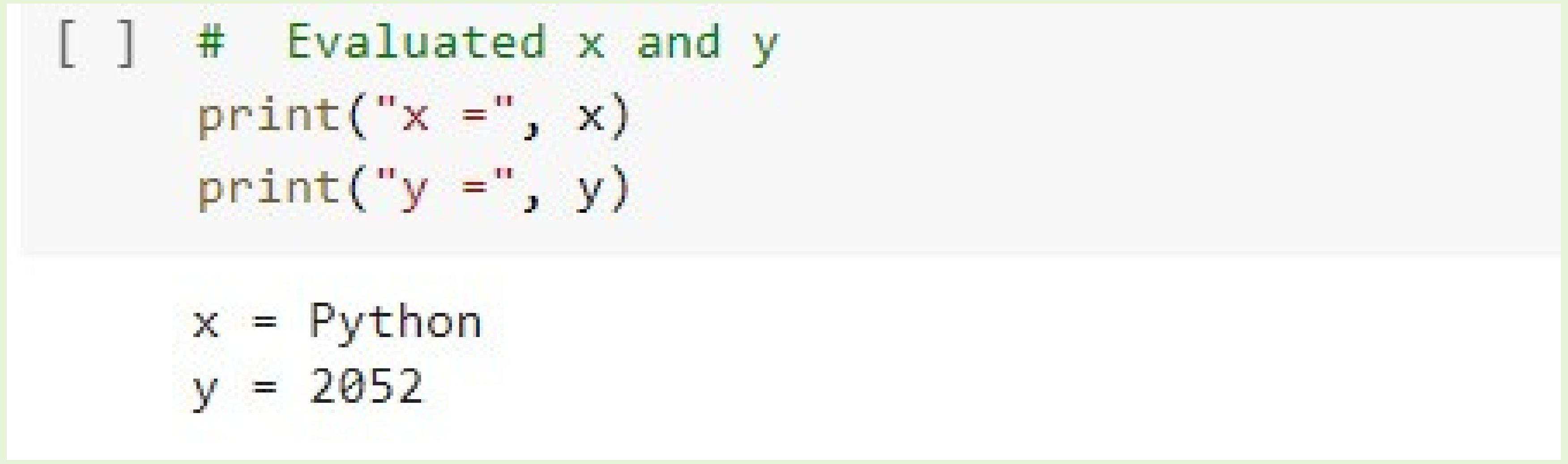
```
type(x)

str
```

Dynamic Typing

PART 2

len() is an inbuilt function in Python programming language that returns the length of the string, does not apply to numeric variables

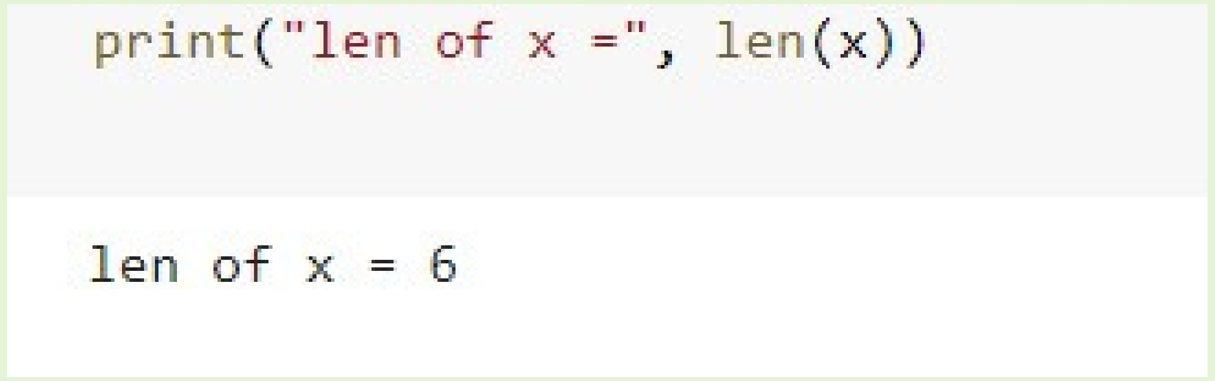
Example:



```
[ ]  #  Evaluated x and y
     print("x =", x)
     print("y =", y)
```

```
x = Python
y = 2052
```

After it runs len of x and y will be returns the length of the string, does not apply to numeric variables



```
print("len of x =", len(x))
```

```
len of x = 6
```

PART 3

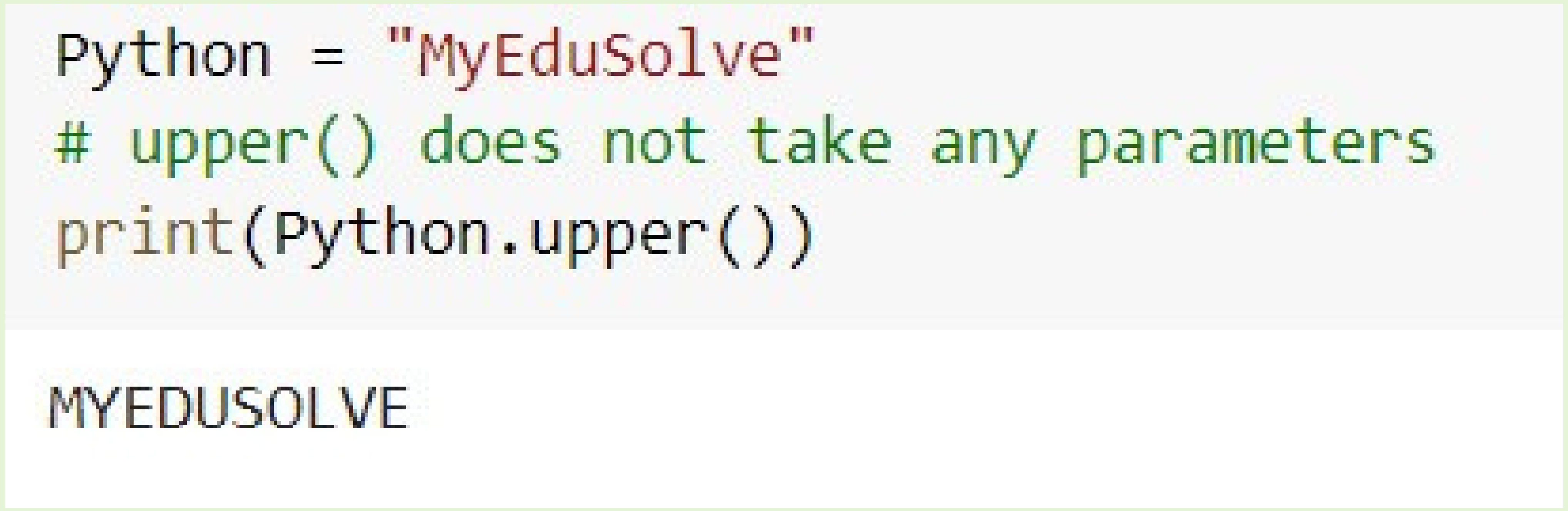
Number, character, and string transformation

Change string uppercase

PART 3

upper() Function

returns the string by converting all the characters of the string to upper case respectively



```
Python = "MyEduSolve"
# upper() does not take any parameters
print(Python.upper())
```

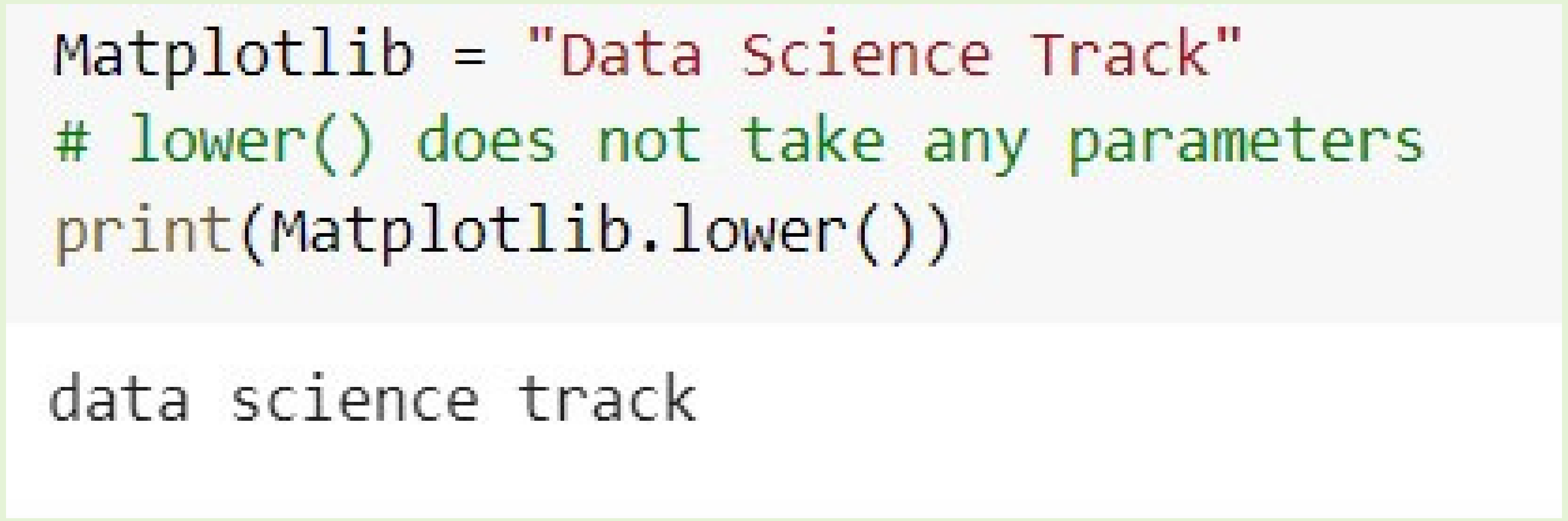
```
MYEDUSOLVE
```

Change string lowercase

PART 3

lower() Function

returns the string by converting all the characters of the string to lowercase respectively.



```
Matplotlib = "Data Science Track"
# lower() does not take any parameters
print(Matplotlib.lower())
```

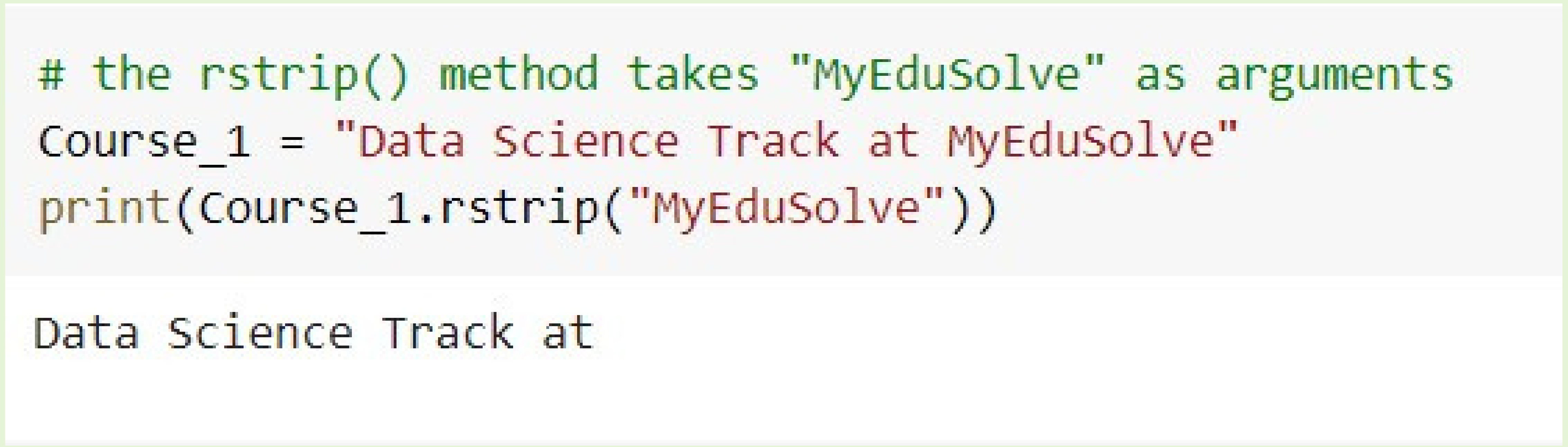
```
data science track
```

Trim a string or line

PART 3

rstrip() Function

The rstrip() method to remove only trailing whitespace and characters, only from the end of a string



```
# the rstrip() method takes "MyEduSolve" as arguments
Course_1 = "Data Science Track at MyEduSolve"
print(Course_1.rstrip("MyEduSolve"))
```

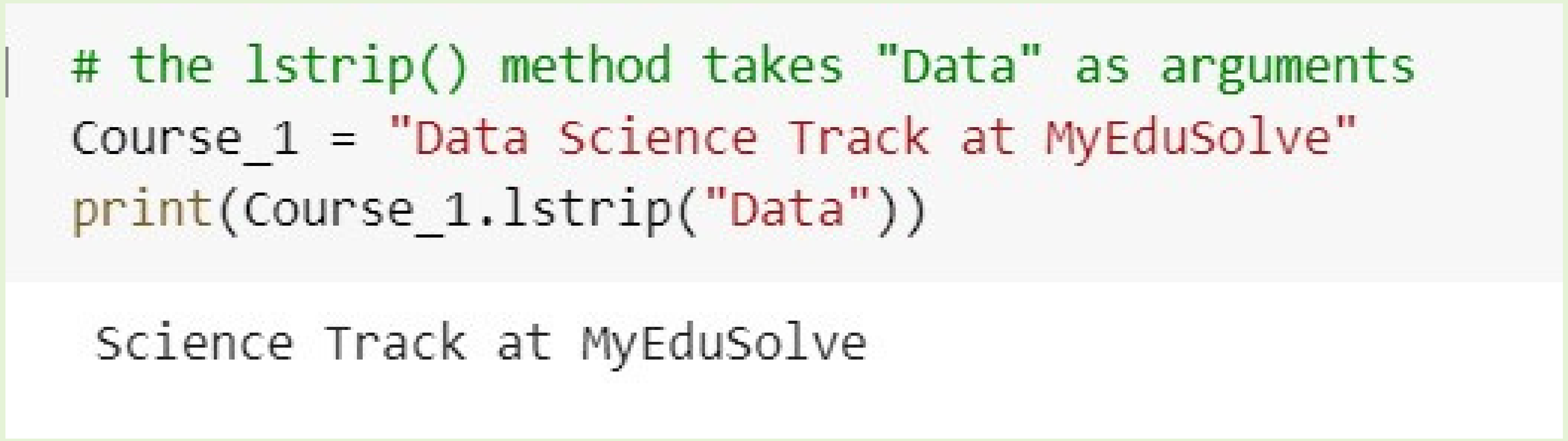
```
Data Science Track at
```

Trim a string or line

PART 3

lstrip() Function

The lstrip() method to remove only leading whitespace and characters, only from the beginning of a string.



```
# the lstrip() method takes "Data" as arguments
Course_1 = "Data Science Track at MyEduSolve"
print(Course_1.lstrip("Data"))
```

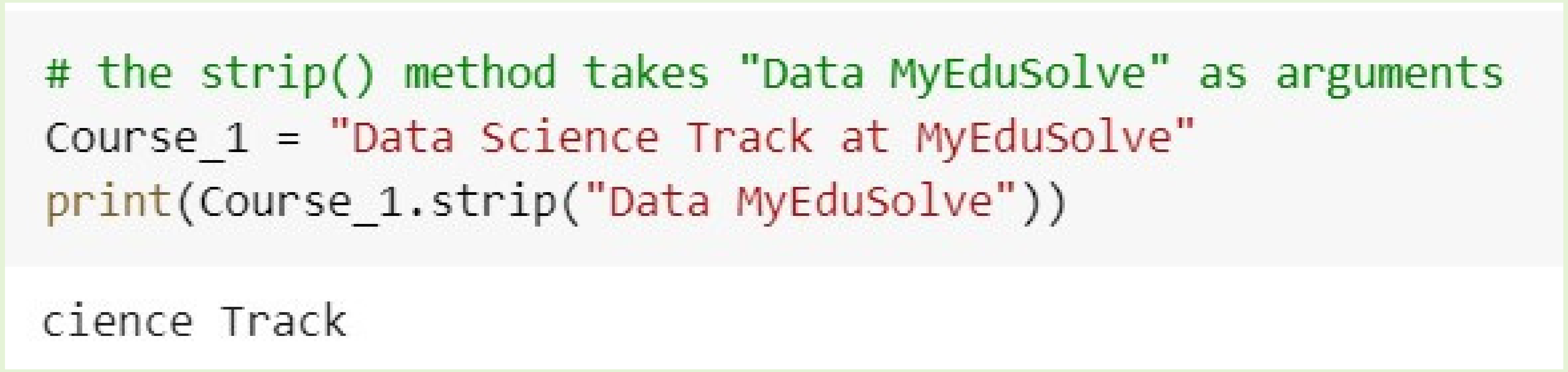
```
 Science Track at MyEduSolve
```

Trim a string or line

PART 3

strip() Function

The strip() method to removes any leading (spaces at the beginning) and trailing (spaces at the end) whitespace and characters of a string



```
# the strip() method takes "Data MyEduSolve" as arguments
Course_1 = "Data Science Track at MyEduSolve"
print(Course_1.strip("Data MyEduSolve"))
```

```
cience Track
```

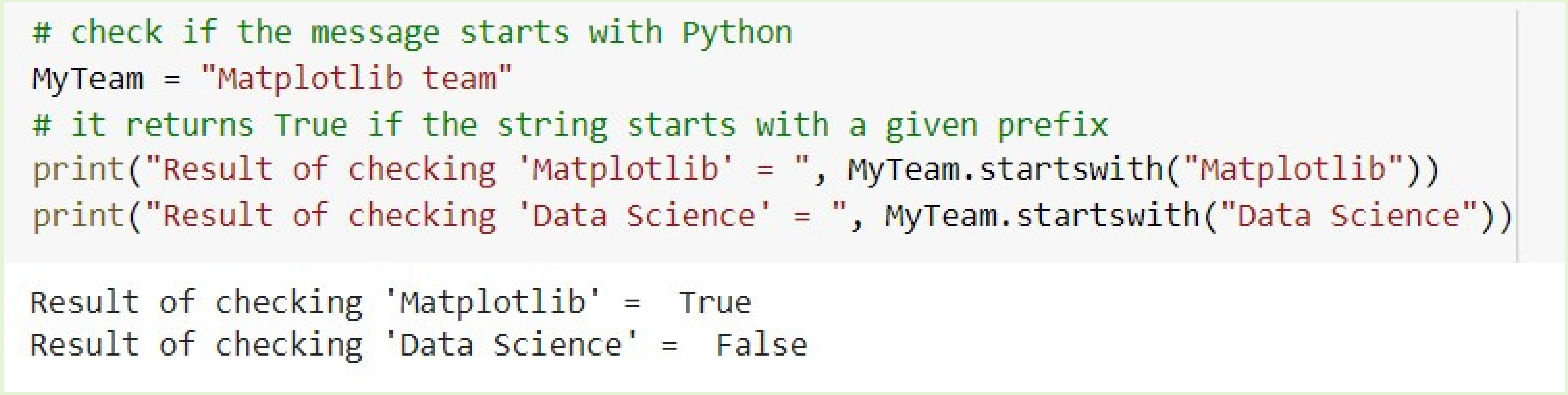
You can to remove "Data", "MyEduSolve", "S" in "Science" (removing from MyEduSolve), and "at" (removing from Data)

Python string startswith

PART 3

startswith() Function

The Python startswith() function checks if a string starts with the given a specified substring or prefix. It returns True if the string starts with a given prefix else False.



```python
# check if the message starts with Python
MyTeam = "Matplotlib team"
# it returns True if the string starts with a given prefix
print("Result of checking 'Matplotlib' = ", MyTeam.startswith("Matplotlib"))
print("Result of checking 'Data Science' = ", MyTeam.startswith("Data Science"))
```

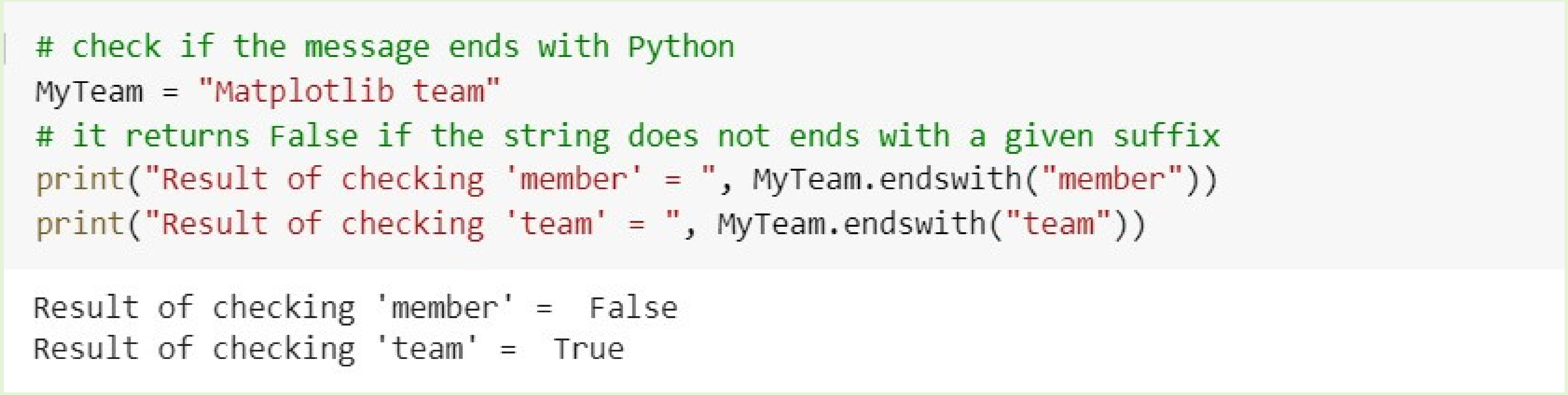
```
Result of checking 'Matplotlib' =  True
Result of checking 'Data Science' =  False
```

Python string endswith

PART 3

endswith() Function

The Python endswith() function checks if a string ends with the given specified substring or suffix. It returns True if the string ends with a given suffix, else False.



```python
# check if the message ends with Python
MyTeam = "Matplotlib team"
# it returns False if the string does not ends with a given suffix
print("Result of checking 'member' = ", MyTeam.endswith("member"))
print("Result of checking 'team' = ", MyTeam.endswith("team"))
```

```
Result of checking 'member' =  False
Result of checking 'team' =  True
```

PART 4

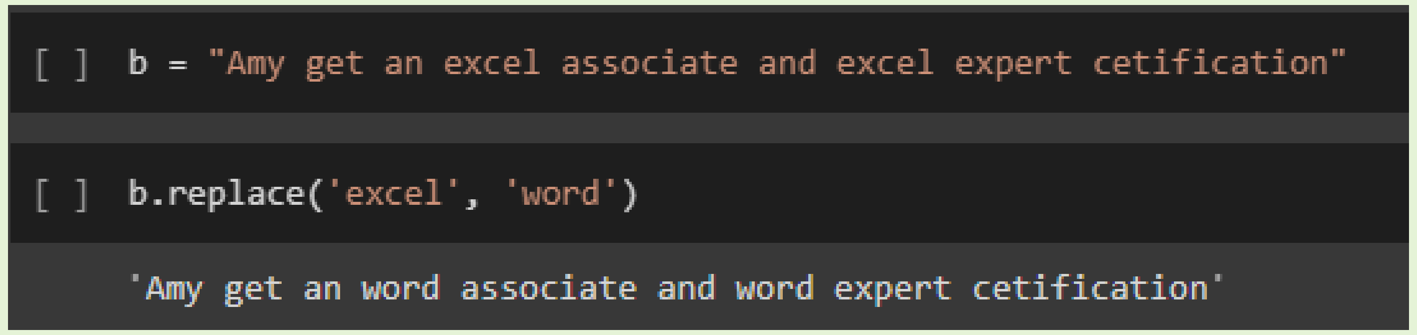
Changing String Element

Changing String

PART 4

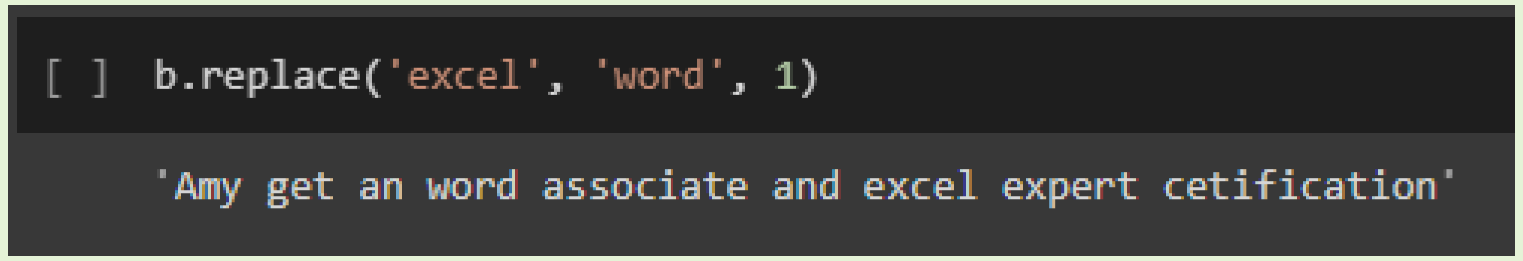
In python, you can change the string to string element if you use replace function.

in default, replace function will replace all string that you command it. but, if you add number at the end, it will replace the n number that you want.



```
[ ]  b = "Amy get an excel associate and excel expert cetification"

[ ]  b.replace('excel', 'word')

     'Amy get an word associate and word expert cetification'
```

by default it would replace all 'excel' to 'word', but if you add number, it select the string you want to replace



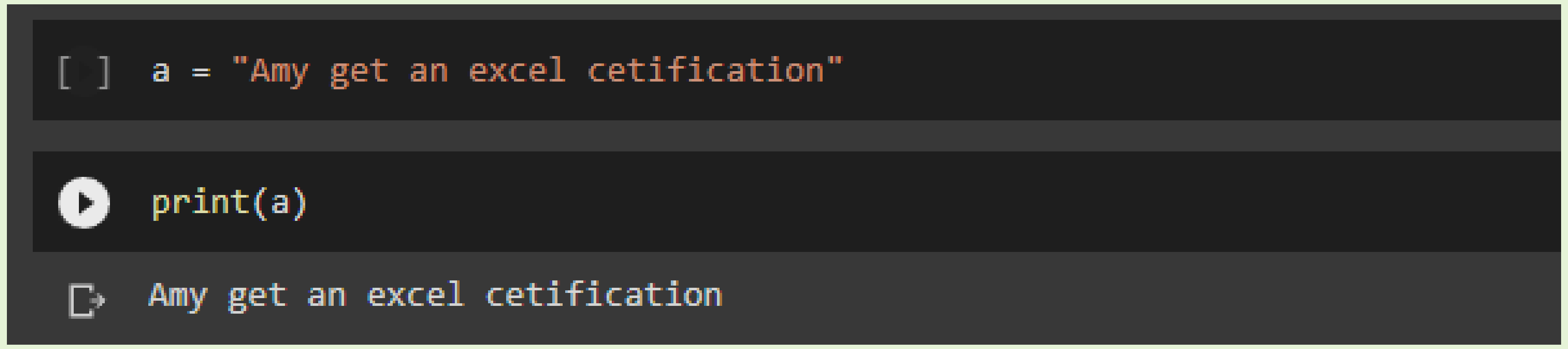
```
[ ]  b.replace('excel', 'word', 1)

     'Amy get an word associate and excel expert cetification'
```

Changing String

PART 4

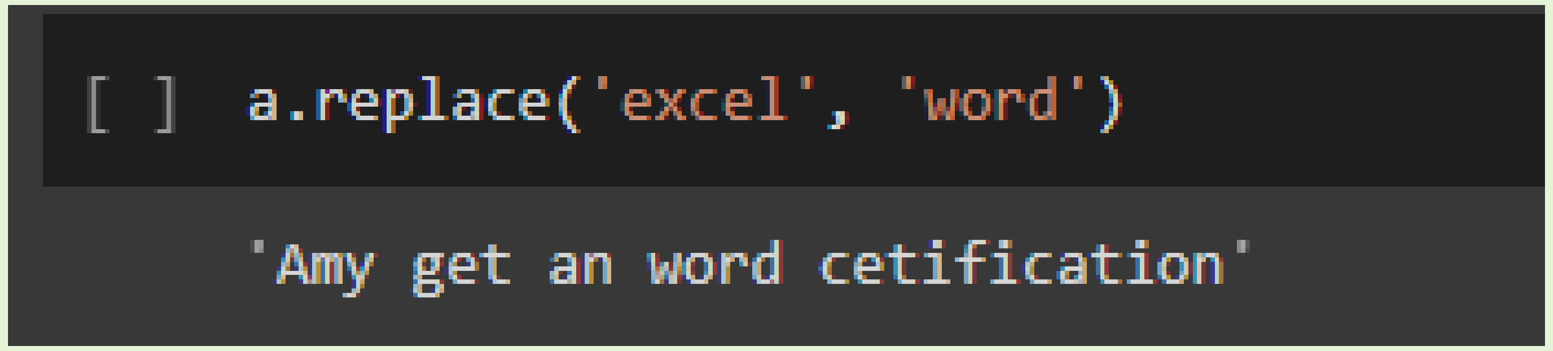
In python, you can change the string to string element if you use replace function.

Example:



```
[ ] a = "Amy get an excel cetification"

print(a)

Amy get an excel cetification
```

if you want to replace the 'excel' to 'word', you can use replace function.



```
[ ] a.replace('excel', 'word')

'Amy get an word cetification'
```

PART 5

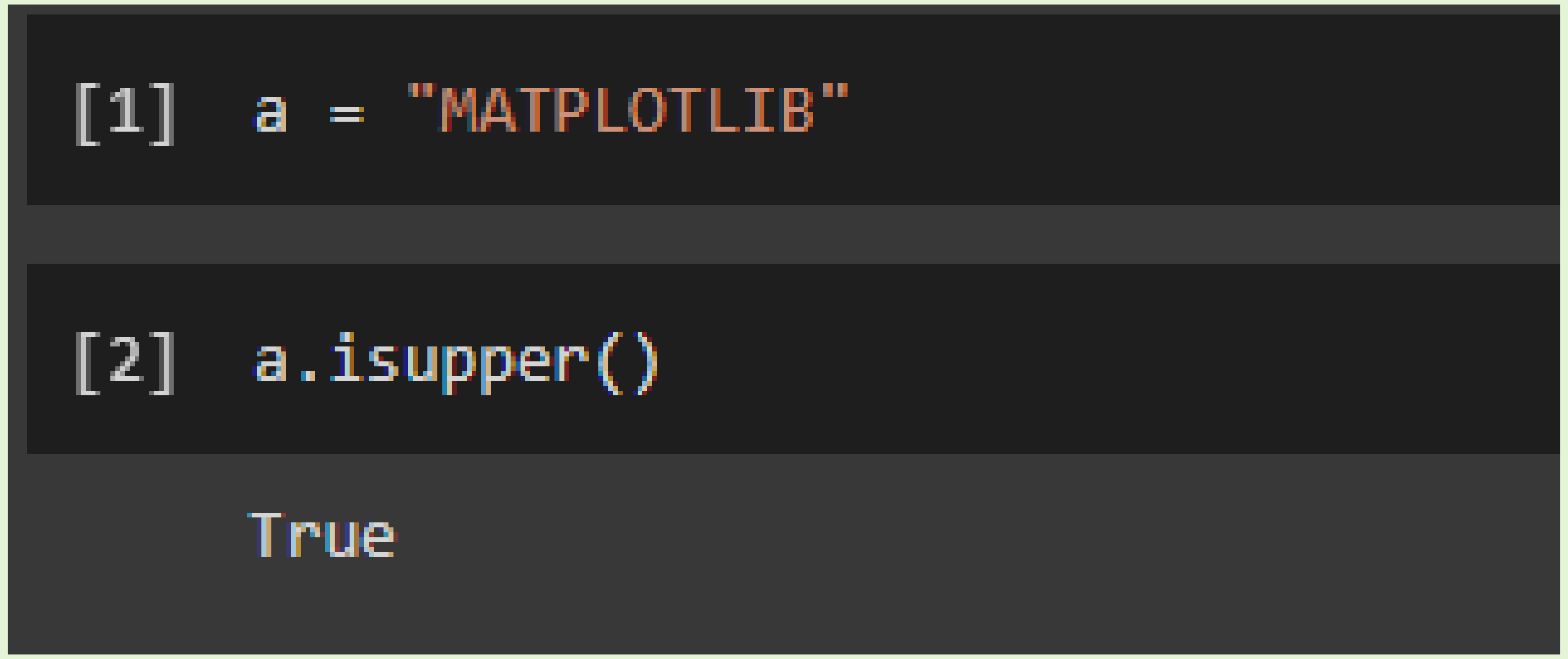
String Checking

String Checking

PART 5

isupper() Function

if you want to check your string is on capital letter, use isupper function. this function will give you boolean results (TRUE or FALSE)



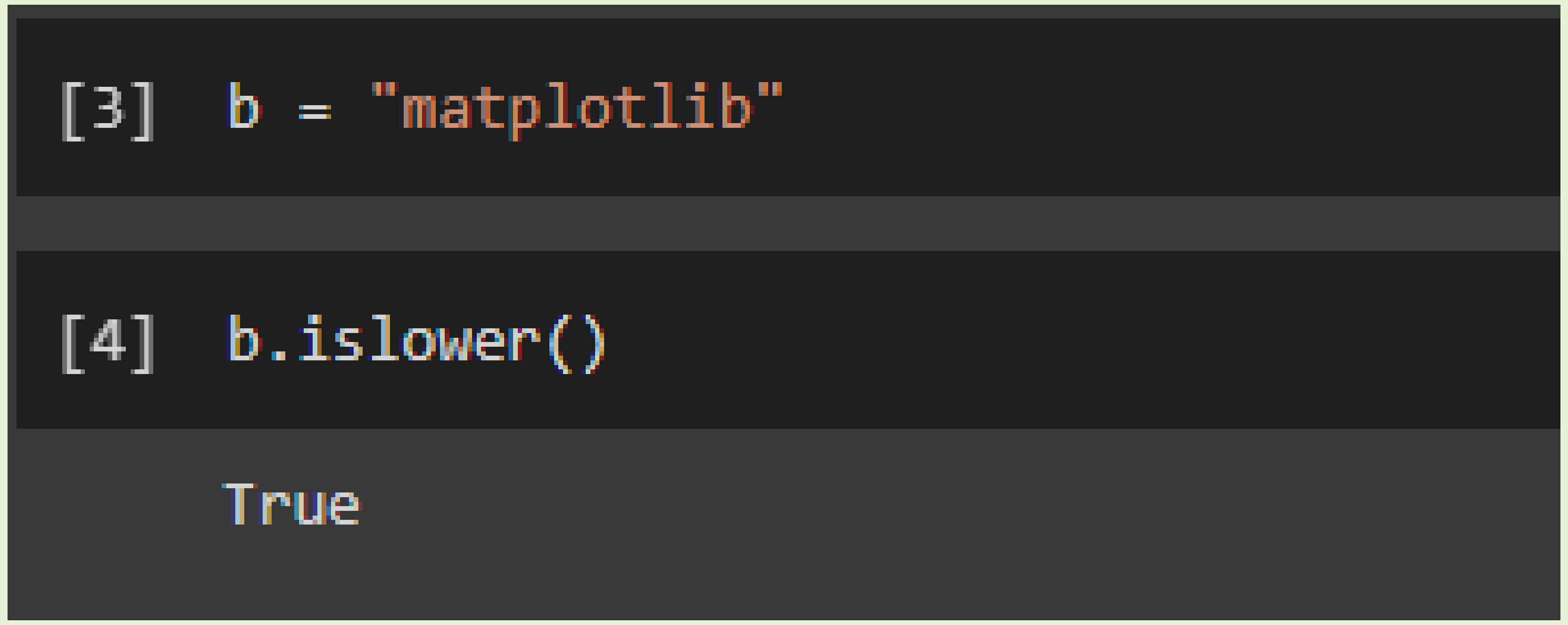
```
[1]  a = "MATPLOTLIB"

[2]  a.isupper()

     True
```

String Checking

PART 5

islower() Function

if you want to check your string is on small letter, use islower function. this function also give you boolean results (TRUE or FALSE)



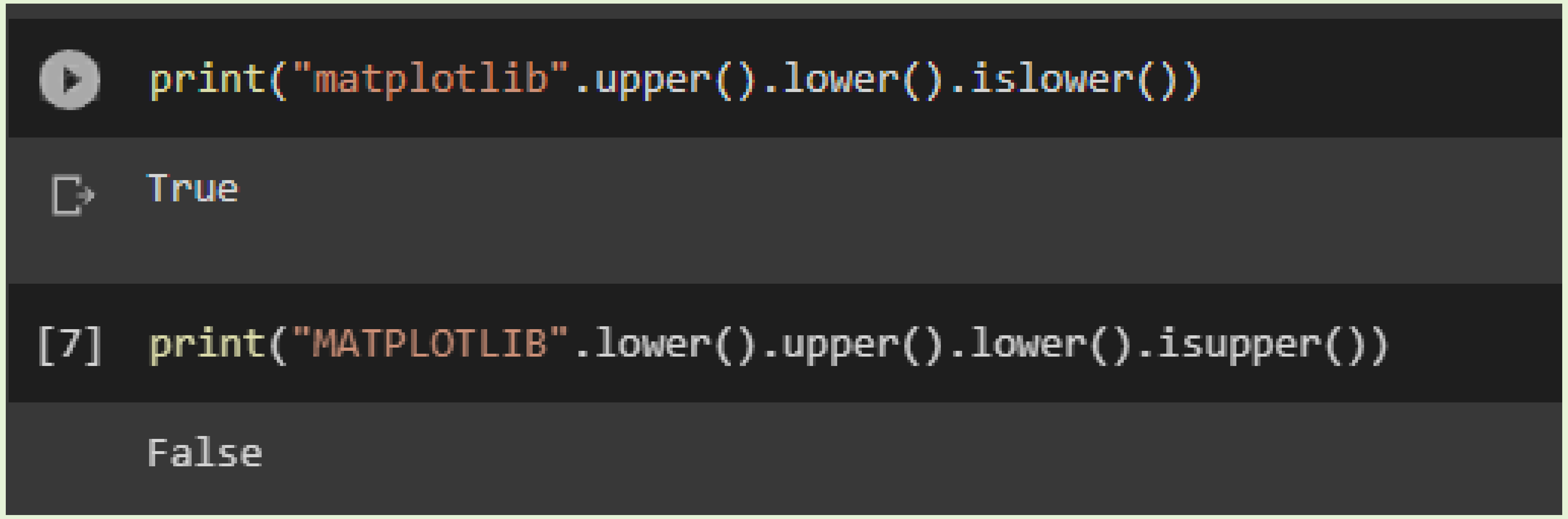
```
[3]  b = "matplotlib"

[4]  b.islower()

     True
```

String Checking

PART 5

isupper() and islower() check type string

isupper and islower function also can checking string although the string type was replaced upper and lower



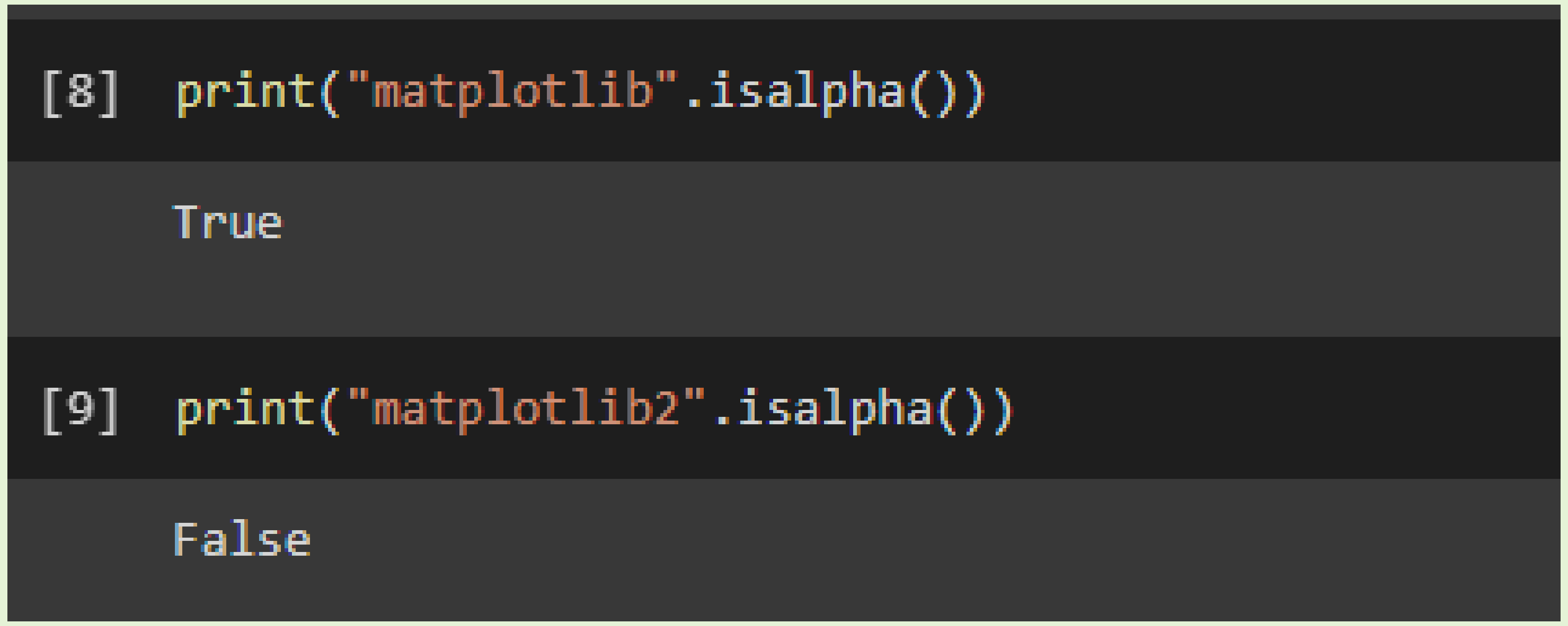
```
print("matplotlib".upper().lower().islower())
True

[7] print("MATPLOTLIB".lower().upper().lower().isupper())
False
```

String Checking

PART 5

isalpha() Function

if you want to check your string are using alphabet or not, use isalpha function. this function also give you boolean results (TRUE or FALSE). if your string has some numbers, it will turn FALSE.



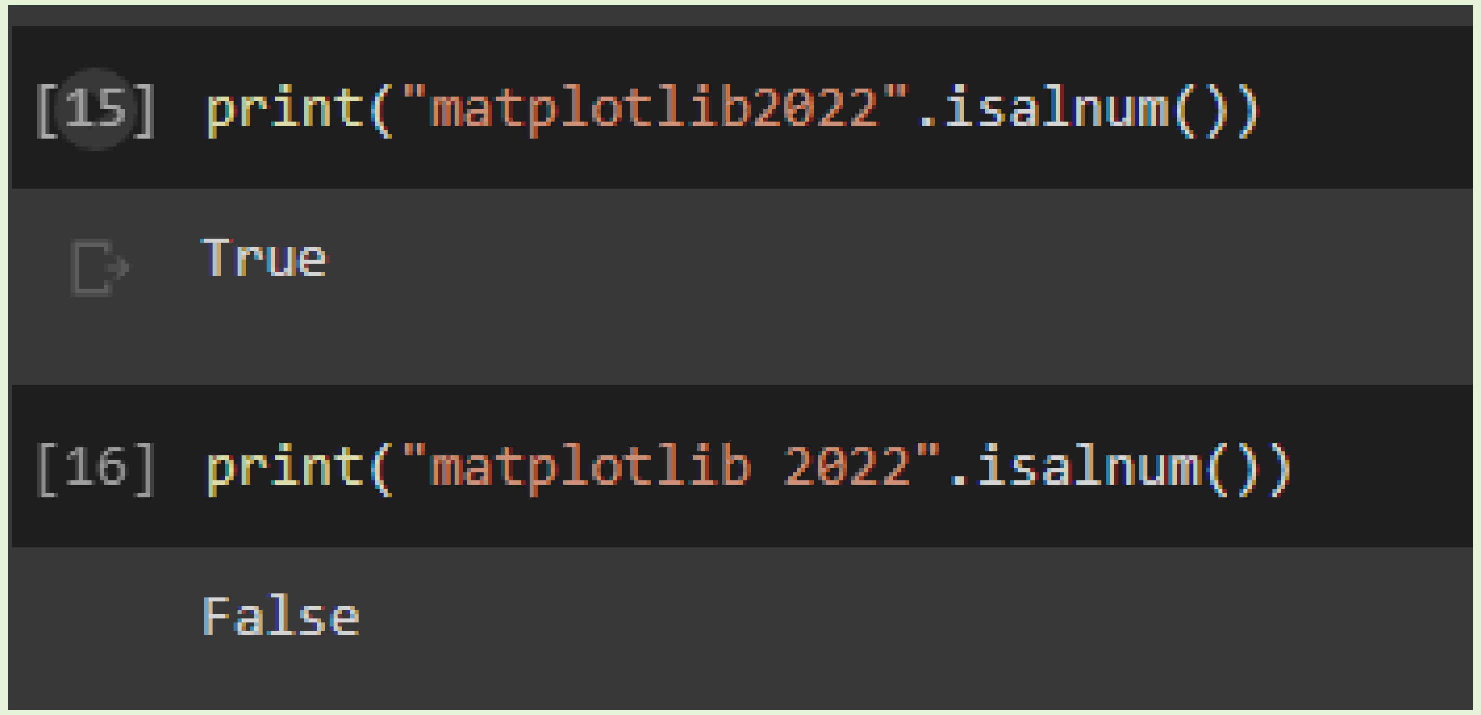
```
[8]  print("matplotlib".isalpha())

     True

[9]  print("matplotlib2".isalpha())

     False
```

String Checking

PART 5

isalnum() Function

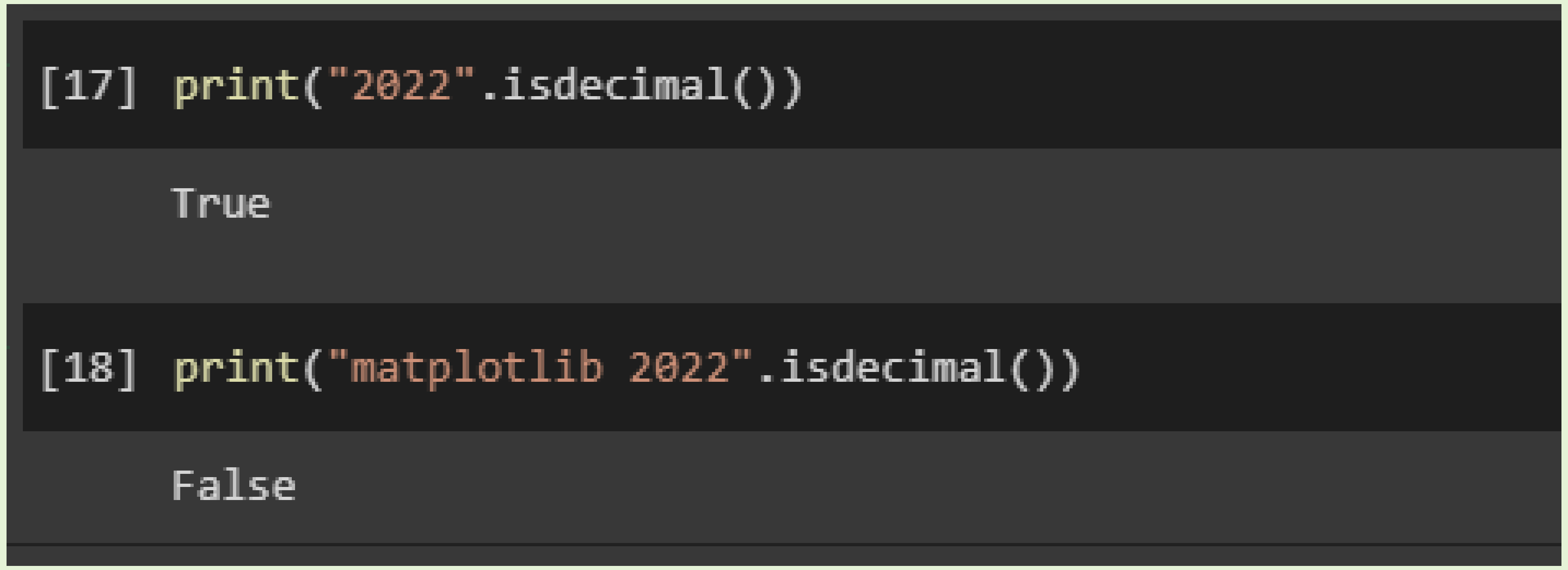
if you want to check your string are using alphanumeric or not, use isalnum function. this function also give you boolean results (TRUE or FALSE). if your string has just alphabet or numeric in one sentence, it will turn FALSE.



```
[15] print("matplotlib2022".isalnum())

     True

[16] print("matplotlib 2022".isalnum())

     False
```

isdecimal() Function

if you want to check your string are decimal (0-9) or not, use isdecimal function. this function also give you boolean results (TRUE or FALSE). if your string has some alphabet or anothher type of string besides decimal (0-9), it will turn FALSE.



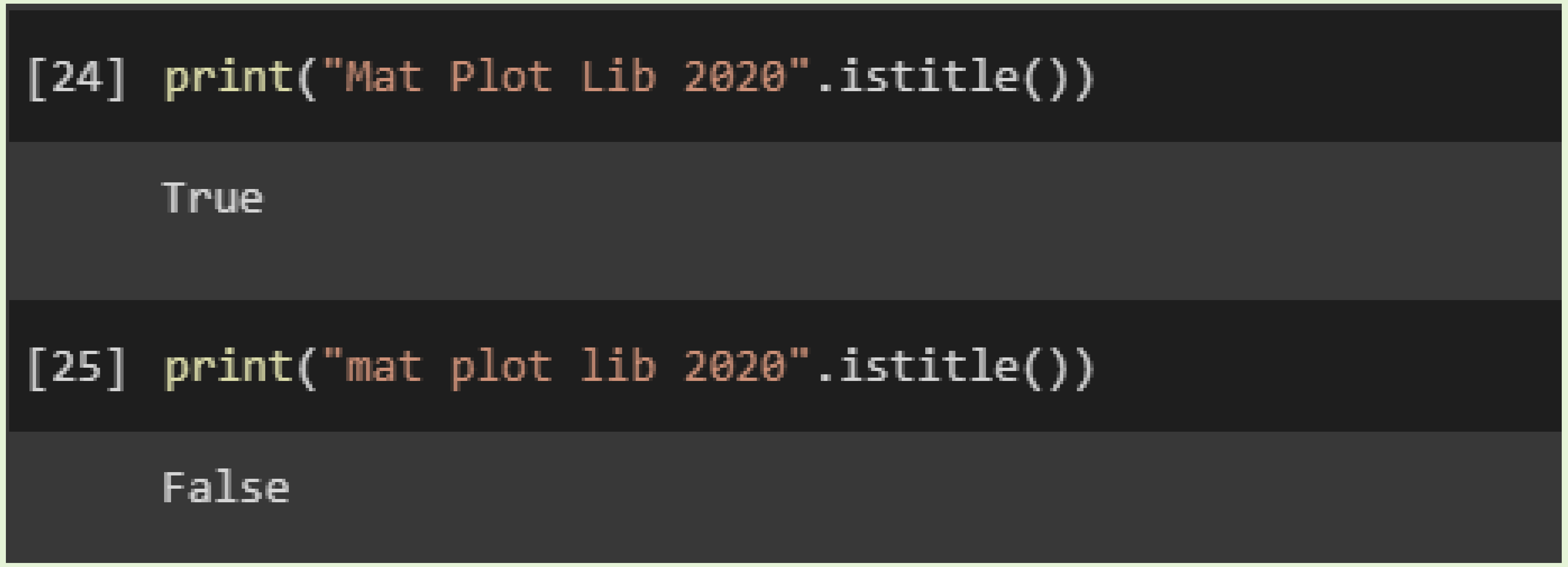
```
[17] print("2022".isdecimal())

     True

[18] print("matplotlib 2022".isdecimal())

     False
```

String Checking

PART 5

istitle() Function

if you want to check your string are start with upper case or not, use istitle function. this function also give you boolean results (TRUE or FALSE).



```
[24] print("Mat Plot Lib 2020".istitle())

     True

[25] print("mat plot lib 2020".istitle())

     False
```

STRING ALIGNMENT

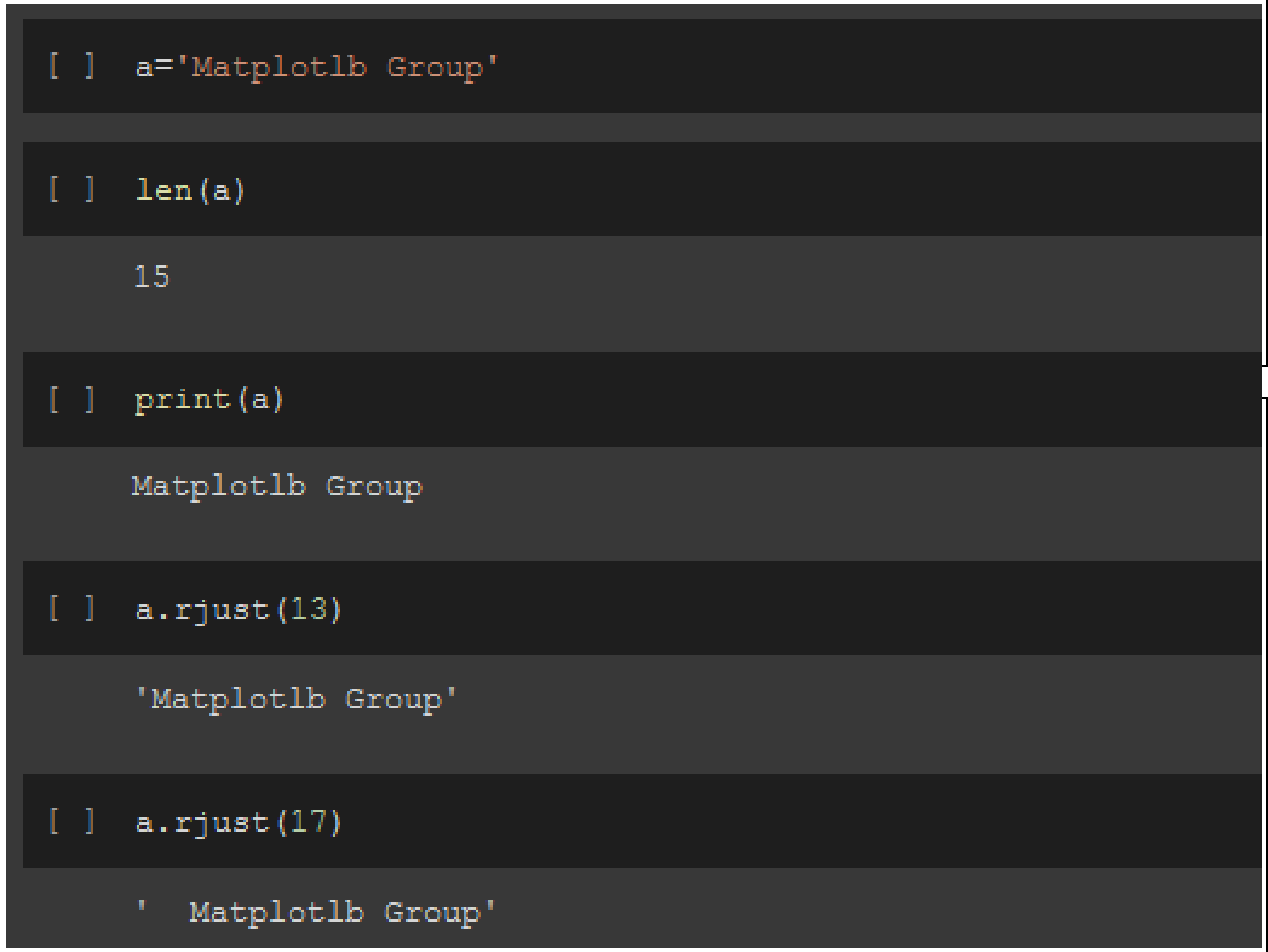
RIGHT ALIGMENT

The rjust() method will right align the string, using a specified character (space is default) as the fill character.
string.

Formatting:
rjust(length, character)

Note:

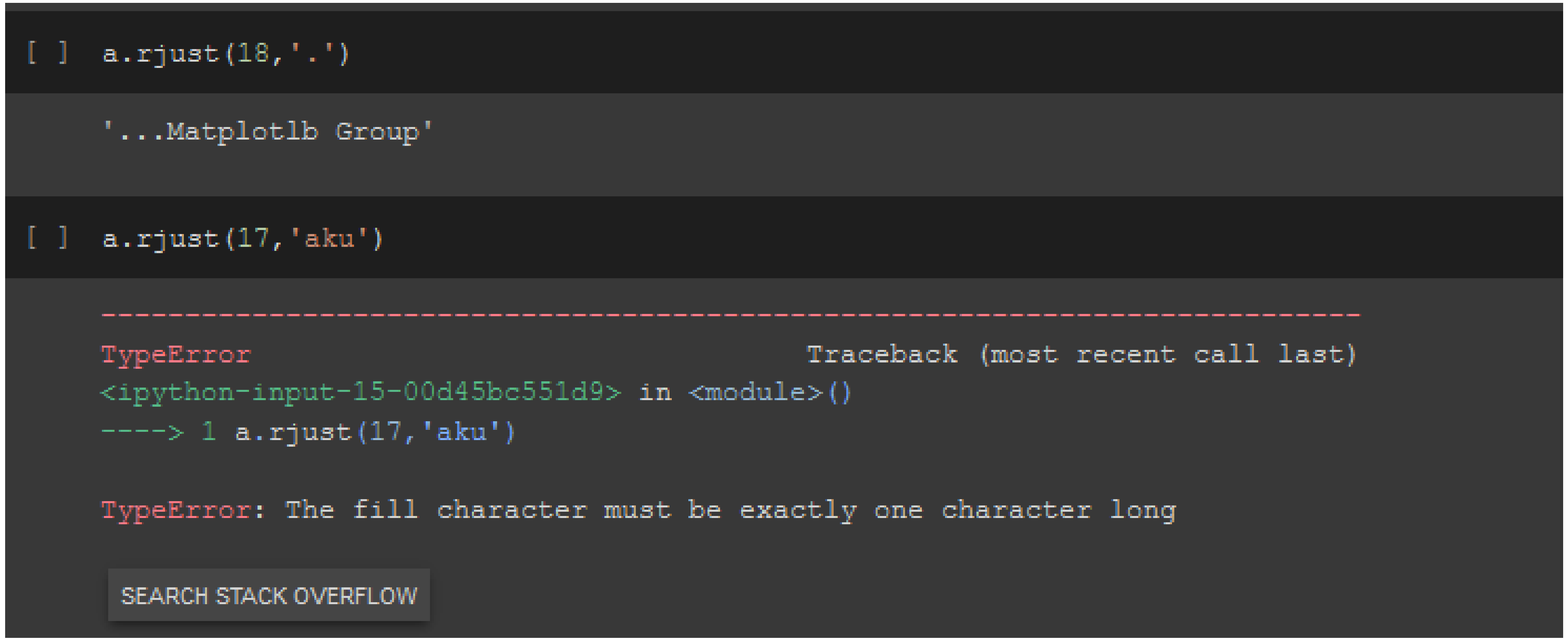
For the length of depending on the number of characters in the sentence. If the length is less than the number of characters in the word, then the alignment does not change



```
[ ] a='Matplotlb Group'

[ ] len(a)

    15

[ ] print(a)

    Matplotlb Group

[ ] a.rjust(13)

    'Matplotlb Group'

[ ] a.rjust(17)

    '  Matplotlb Group'
```

RIGHT ALIGMENT

For any alignment can be replaced by other string characters. If the character is replaced by a word it will be an error



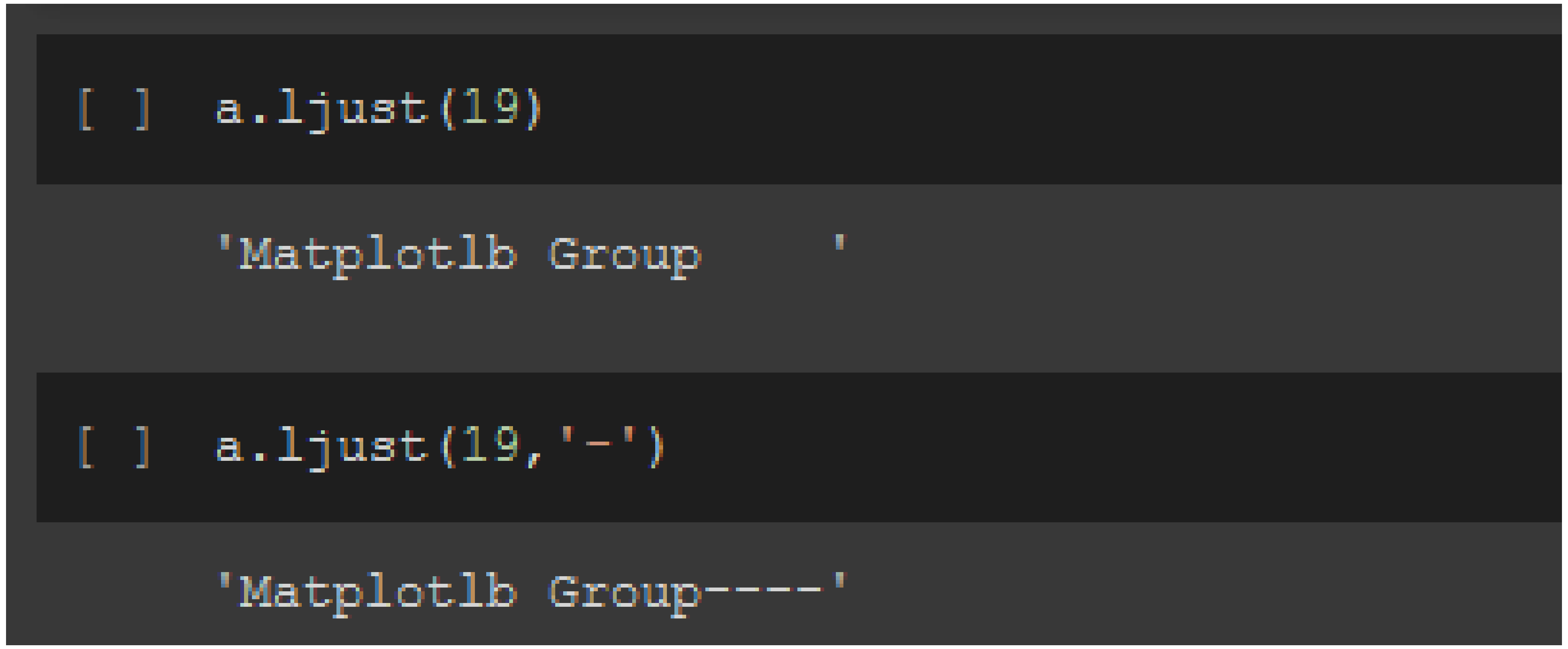
```
[ ]  a.rjust(18,'.')

     '...Matplotlb Group'

[ ]  a.rjust(17,'aku')

     ---------------------------------------------------------------------------
     TypeError                                 Traceback (most recent call last)
     <ipython-input-15-00d45bc551d9> in <module>()
     ----> 1 a.rjust(17,'aku')

     TypeError: The fill character must be exactly one character long
```

LEFT ALIGMENT

TThe ljust() method will left align the string, using a specified character (space is default) as the fill character.
string.

Formatting:
ljust(length, character)

For any alignment can be replaced by other string characters. If the character is replaced by a word it will be an error



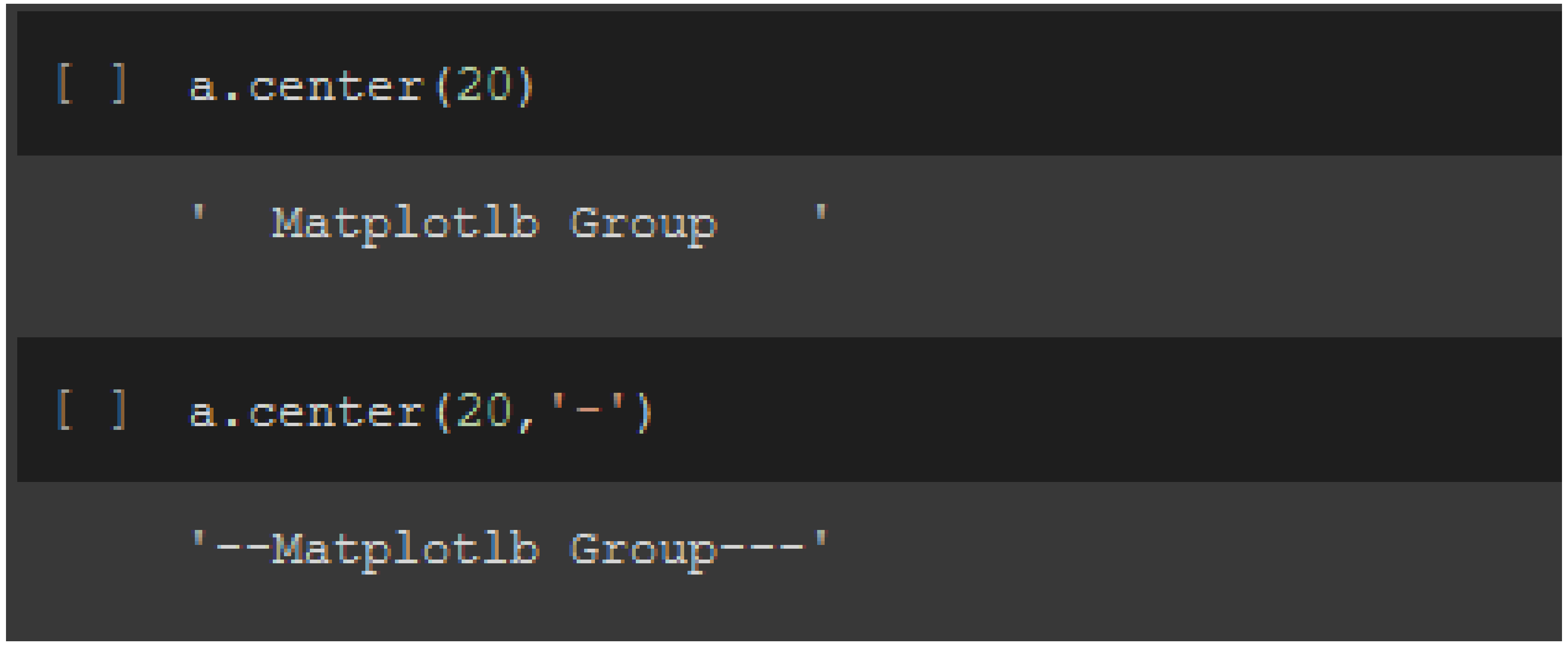
```
[ ]  a.ljust(19)

     'Matplotlb Group    '

[ ]  a.ljust(19,'-')

     'Matplotlb Group----'
```

CENTER ALIGMENT

The center() method will center align the string, using a specified character (space is default) as the fill character.

Formatting:
center(length, character)

For any alignment can be replaced by other string characters. If the character is replaced by a word it will be an error



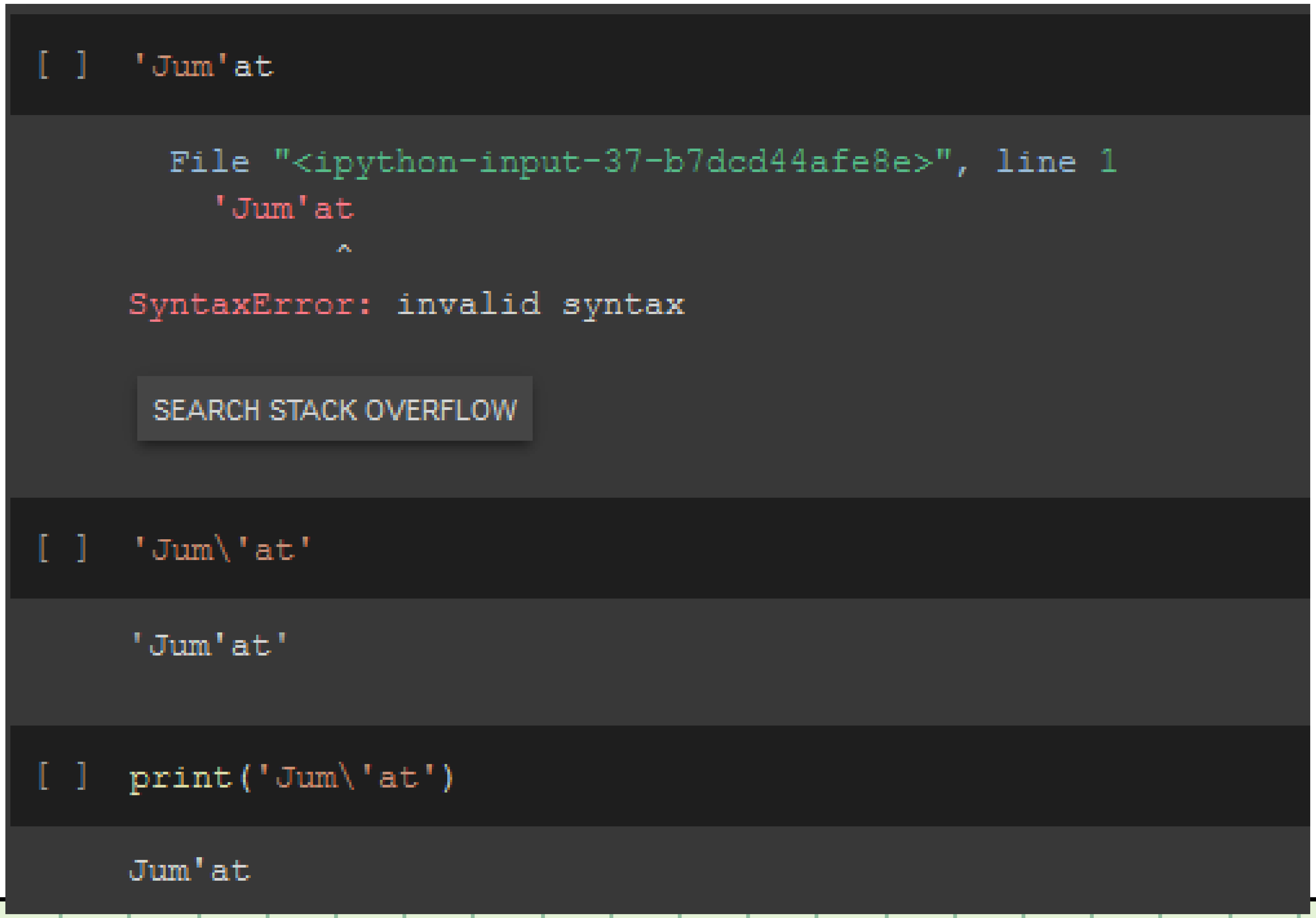
```
[ ]  a.center(20)

     '  Matplotlib Group   '

[ ]  a.center(20,'-')

     '--Matplotlib Group---'
```

STRING FORMATTING

STRING APOSTROPHE

USING \

- if in python you want to use apostrophe you must add (\ ) so that apostrophe can be read

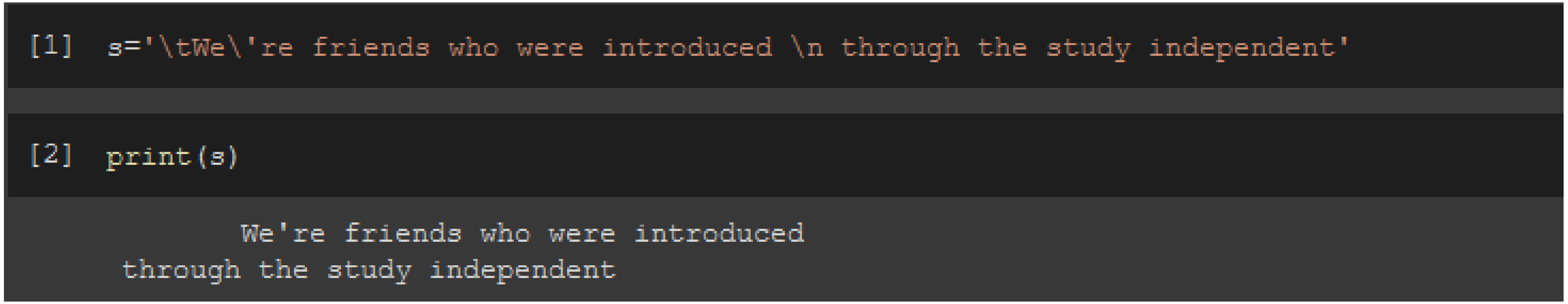


```
[ ]  'Jum'at

       File "<ipython-input-37-b7dcd44afe8e>", line 1
         'Jum'at
              ^
     SyntaxError: invalid syntax
```

```
[ ]  'Jum\'at'

     'Jum'at'

[ ]  print('Jum\'at')

     Jum'at
```

USING \

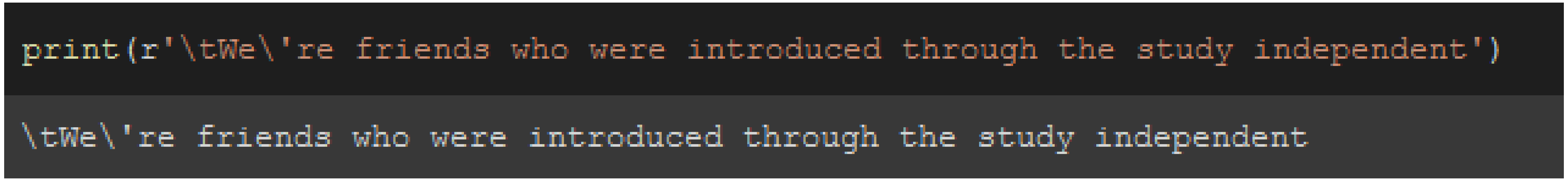
- \ can also provide tabs when \ is used with the letter t. when used with the letter n it will define as enter.



```
[1]  s='\tWe\'re friends who were introduced \n through the study independent'

[2]  print(s)

             We're friends who were introduced
      through the study independent
```

USING \

- If you add the letter r at the beginning of the sentence, it will reverse the value in the initial format



```
print(r'\tWe\'re friends who were introduced through the study independent')
```

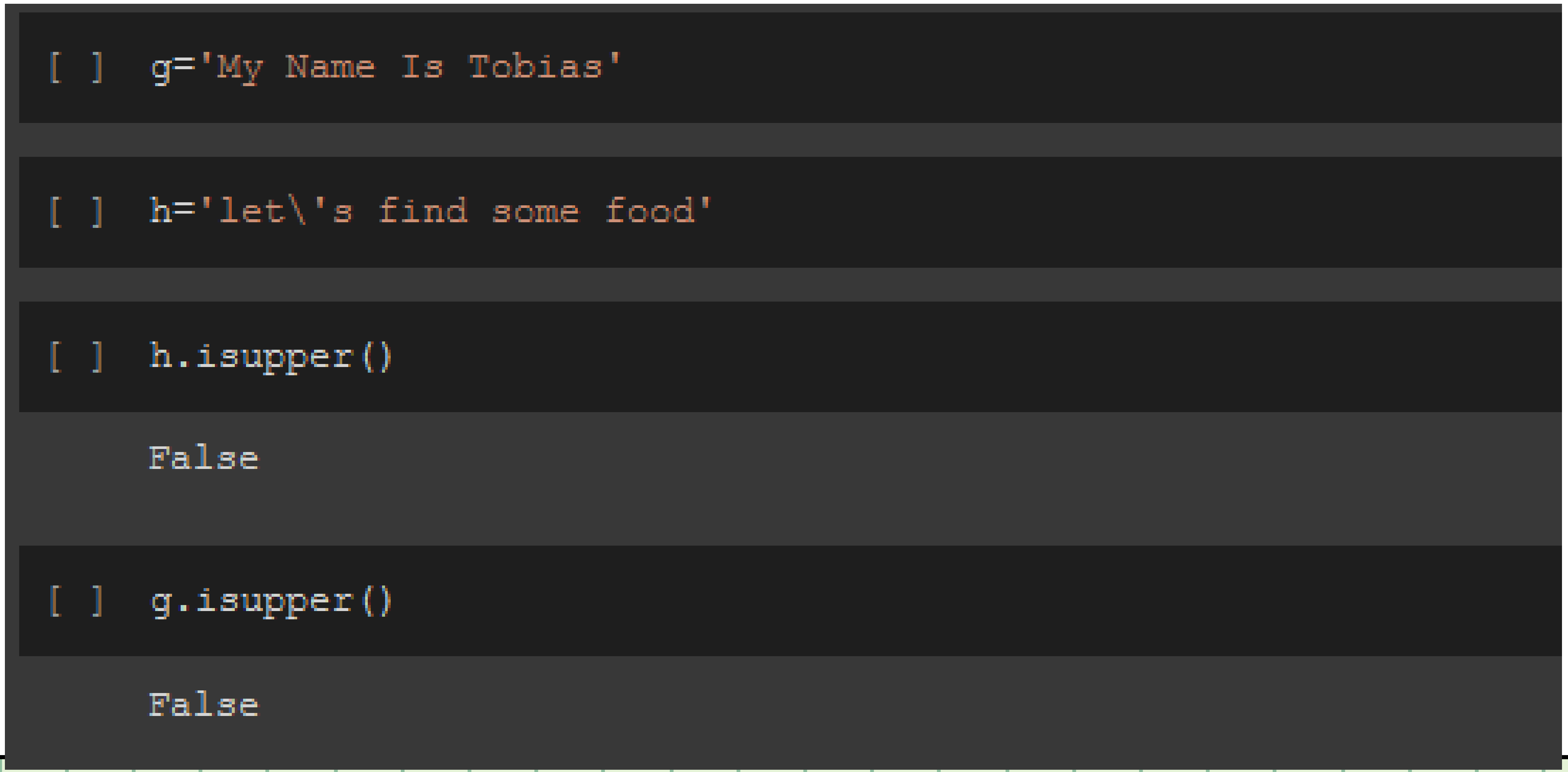
```
\tWe\'re friends who were introduced through the study independent
```

STRING FORMATTING

UPPER AND LOWER

UPPER AND LOWER CHECK

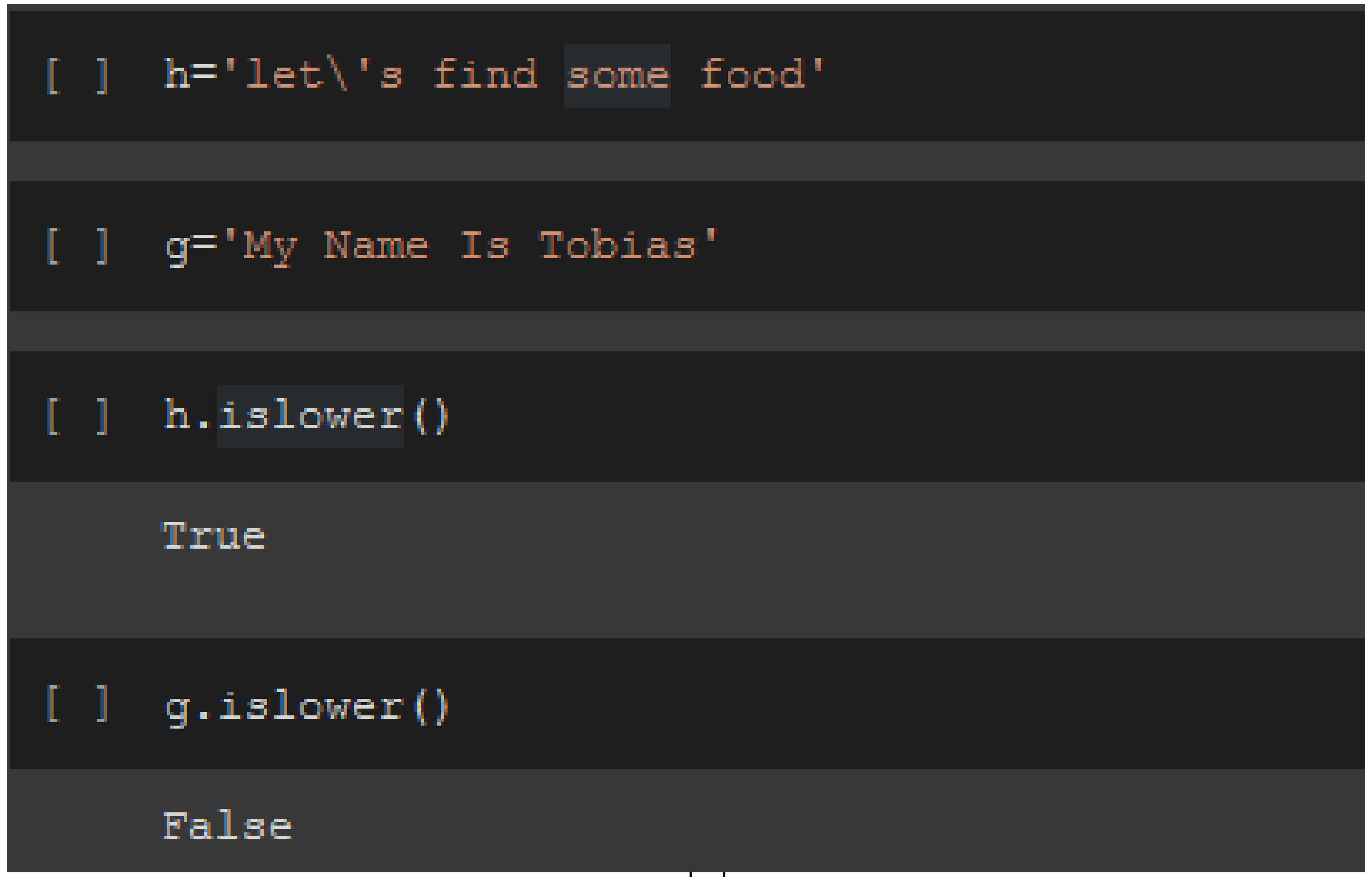
- To check sentences are upper case using isupper(). Returns True if all characters in the string are upper case. If there is 1 lowercase letter it will be false.



```
[ ]  g='My Name Is Tobias'

[ ]  h='let\'s find some food'

[ ]  h.isupper()

     False

[ ]  g.isupper()

     False
```

UPPER AND LOWER CHECK

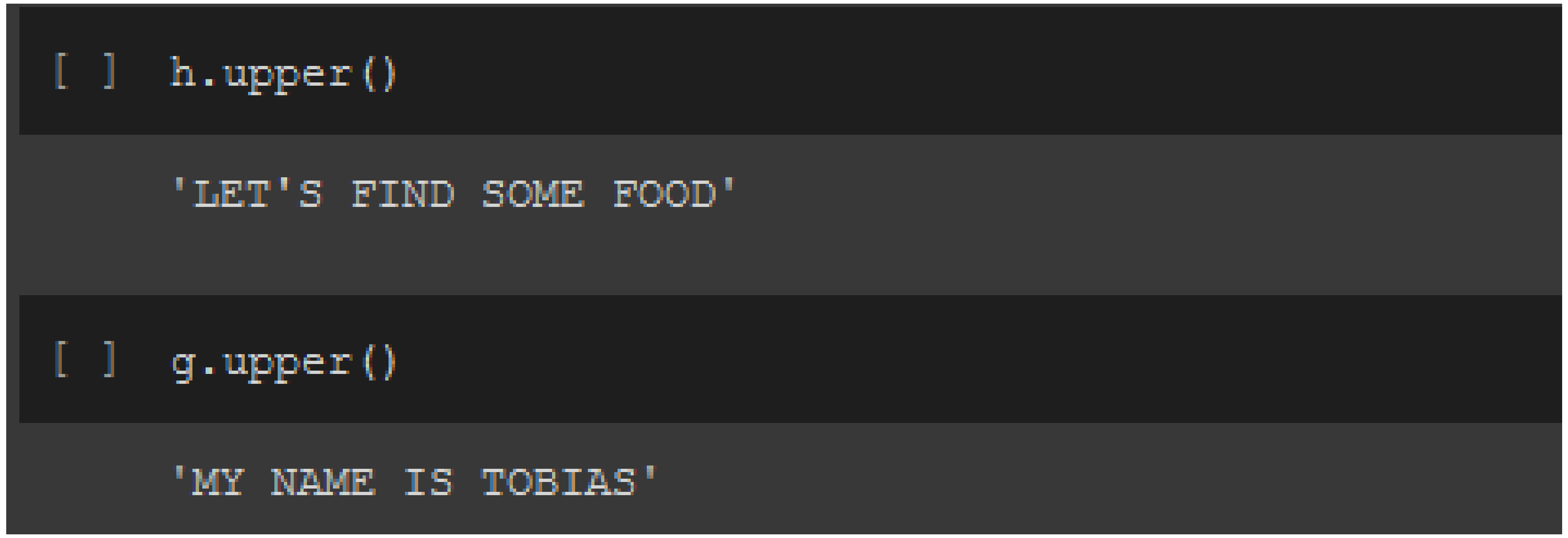
- To check sentences are lower case using islower(). Returns True if all characters in the string are lower case. If there is 1 uppercase letter it will be false.



```
[ ]  h='let\'s find some food'

[ ]  g='My Name Is Tobias'

[ ]  h.islower()

     True

[ ]  g.islower()

     False
```

UPPER AND LOWER FORMATTING

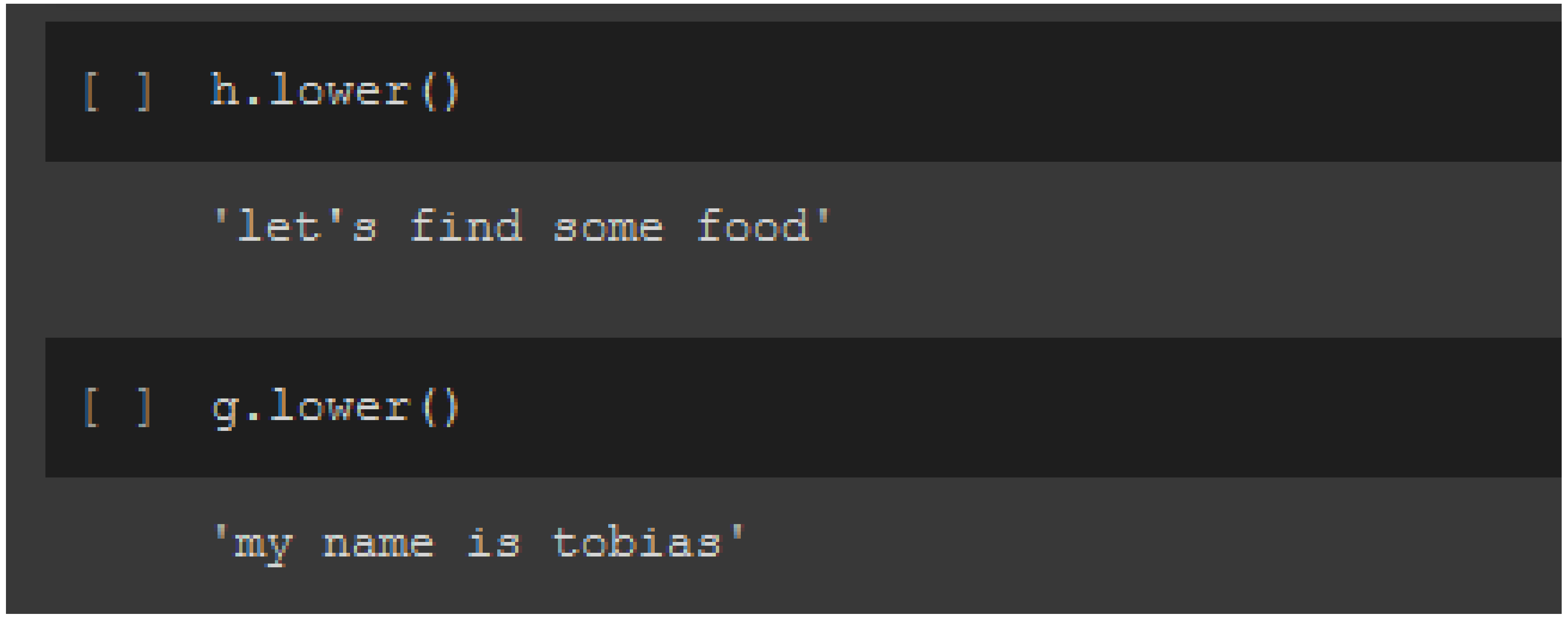
- The upper() method returns a string where all characters are in upper case.
- Symbols and Numbers are ignored.



```
[ ]  h.upper()

     'LET'S FIND SOME FOOD'

[ ]  g.upper()

     'MY NAME IS TOBIAS'
```

UPPER AND LOWER FORMATTING

- The lower() method returns a string where all characters are lower case.
- Symbols and Numbers are ignored.



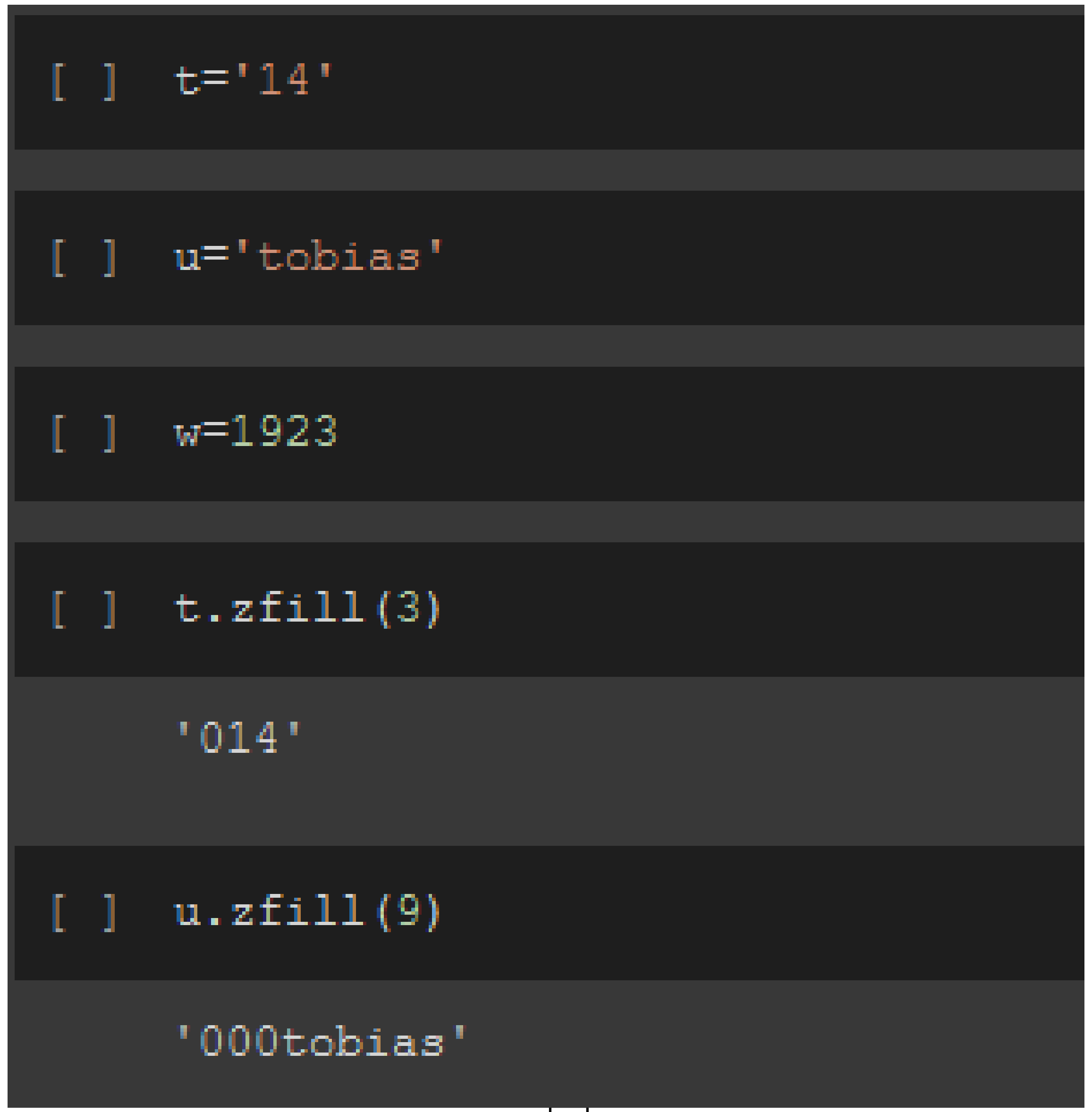
```
[ ]  h.lower()

     'let's find some food'

[ ]  g.lower()

     'my name is tobias'
```

STRING FORMATTING

ZFILL

ZFILL

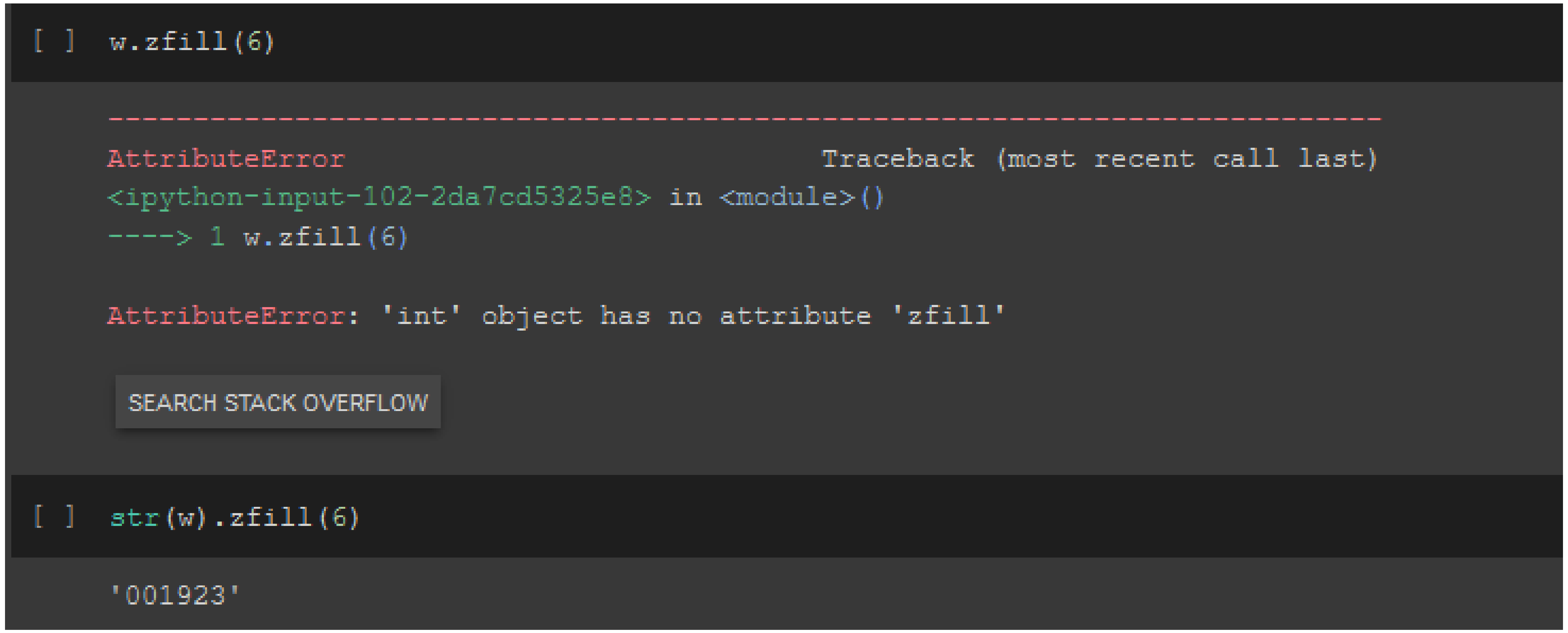
- The zfill() method adds zeros (0) at the beginning of the string, until it reaches the specified length.



```
[ ]  t='14'

[ ]  u='tobias'

[ ]  w=1923

[ ]  t.zfill(3)

     '014'

[ ]  u.zfill(9)

     '000tobias'
```

ZFILL

- if other than string then won't be able to work. So you have to change the string first for it to work

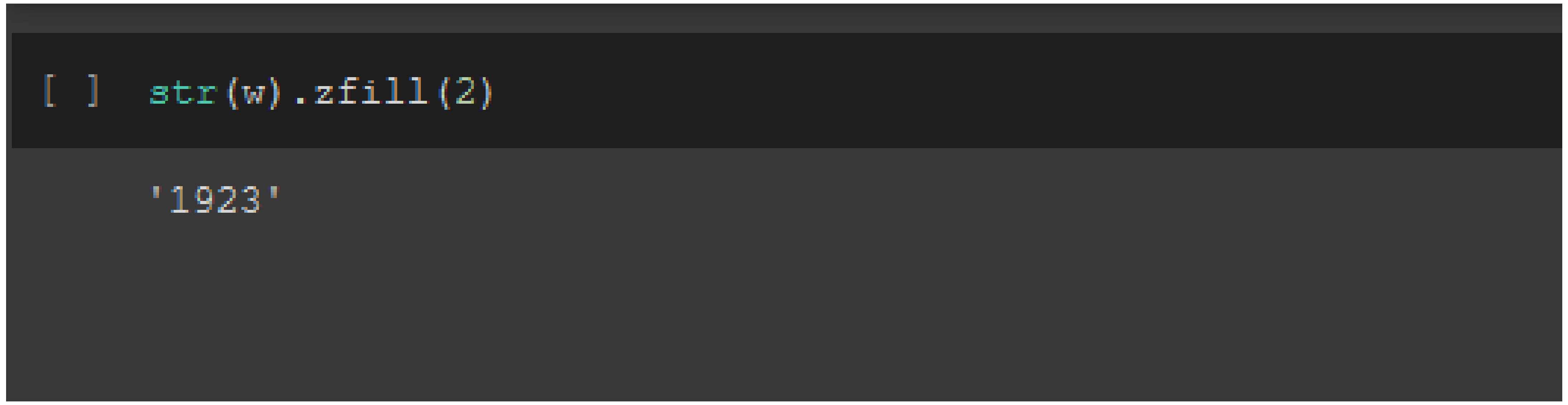


```
[ ]  w.zfill(6)

     ---------------------------------------------------------------------------
     AttributeError                            Traceback (most recent call last)
     <ipython-input-102-2da7cd5325e8> in <module>()
     ----> 1 w.zfill(6)

     AttributeError: 'int' object has no attribute 'zfill'
```

```
[ ]  str(w).zfill(6)

     '001923'
```

ZFILL

- If the value of the len parameter is less than the length of the string, no filling is done.



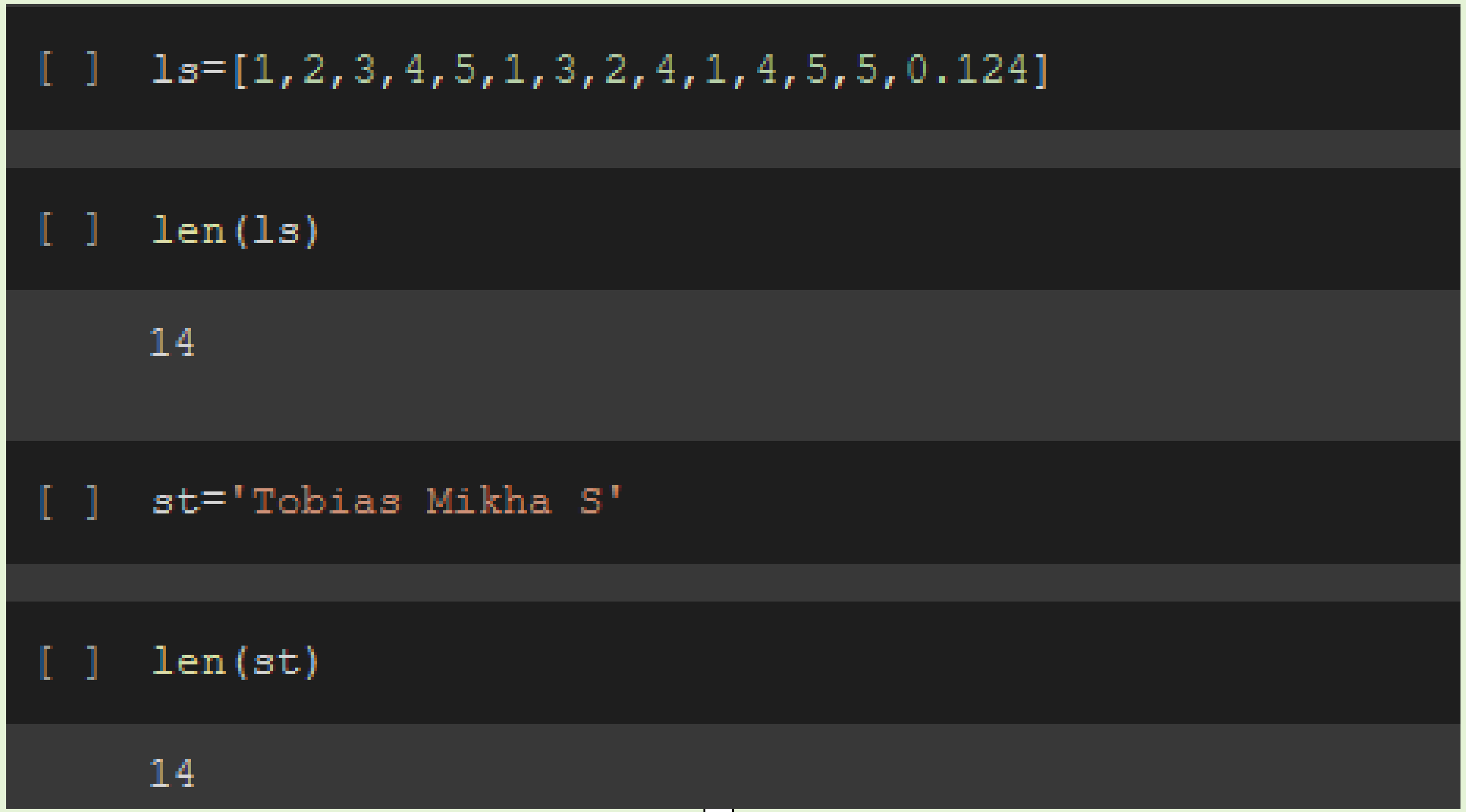
```
[ ]  str(w).zfill(2)

     '1923'
```

OPERATION STRING, LIST, SET

LEN

LEN

- Used to count the amount of existing data or the number of characters in srings



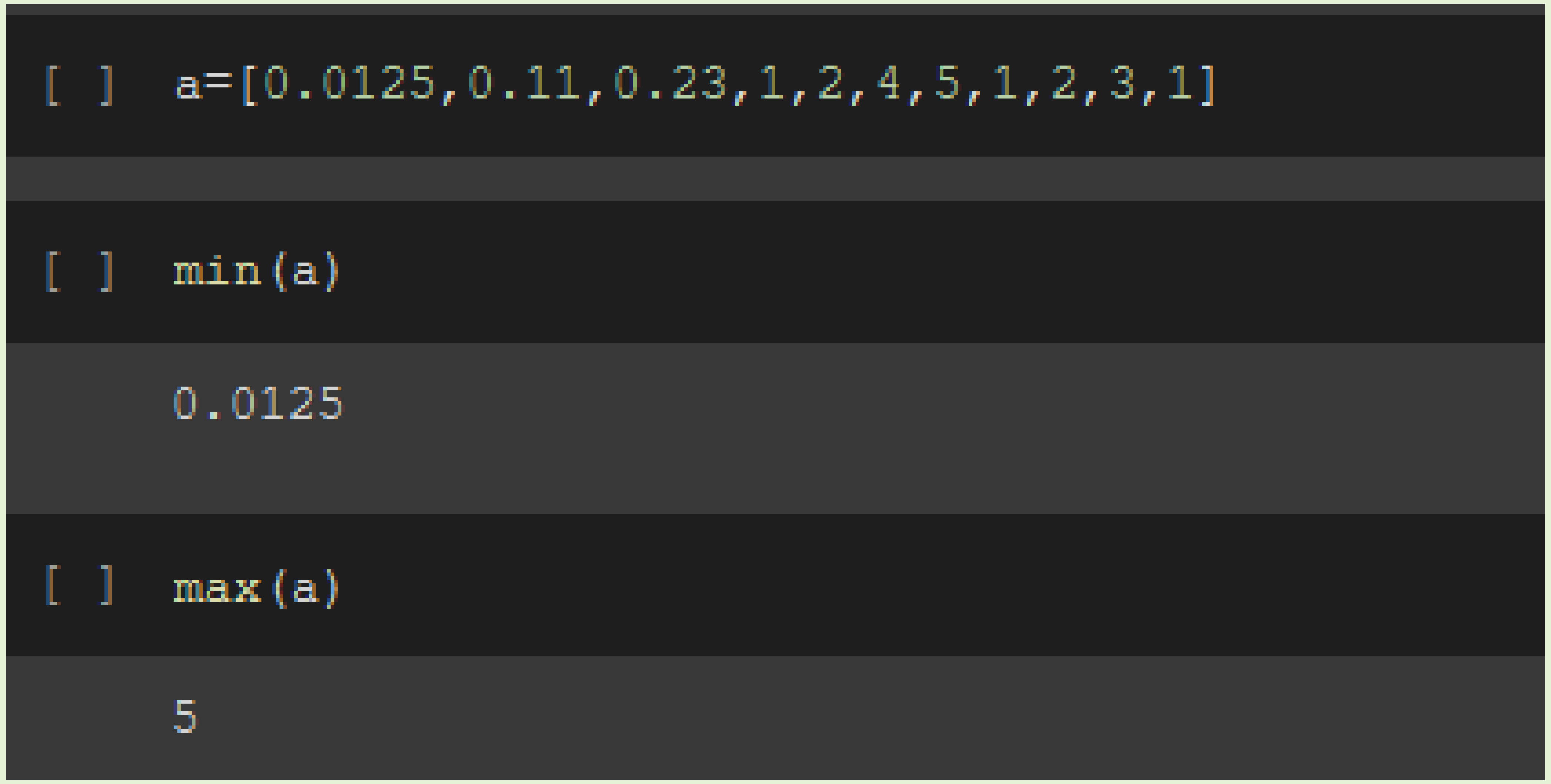
```
[ ]  ls=[1,2,3,4,5,1,3,2,4,1,4,5,5,0.124]

[ ]  len(ls)

     14

[ ]  st='Tobias Mikha S'

[ ]  len(st)

     14
```

OPERATION STRING, LIST, SET

Min and Max

MIN AND MAX

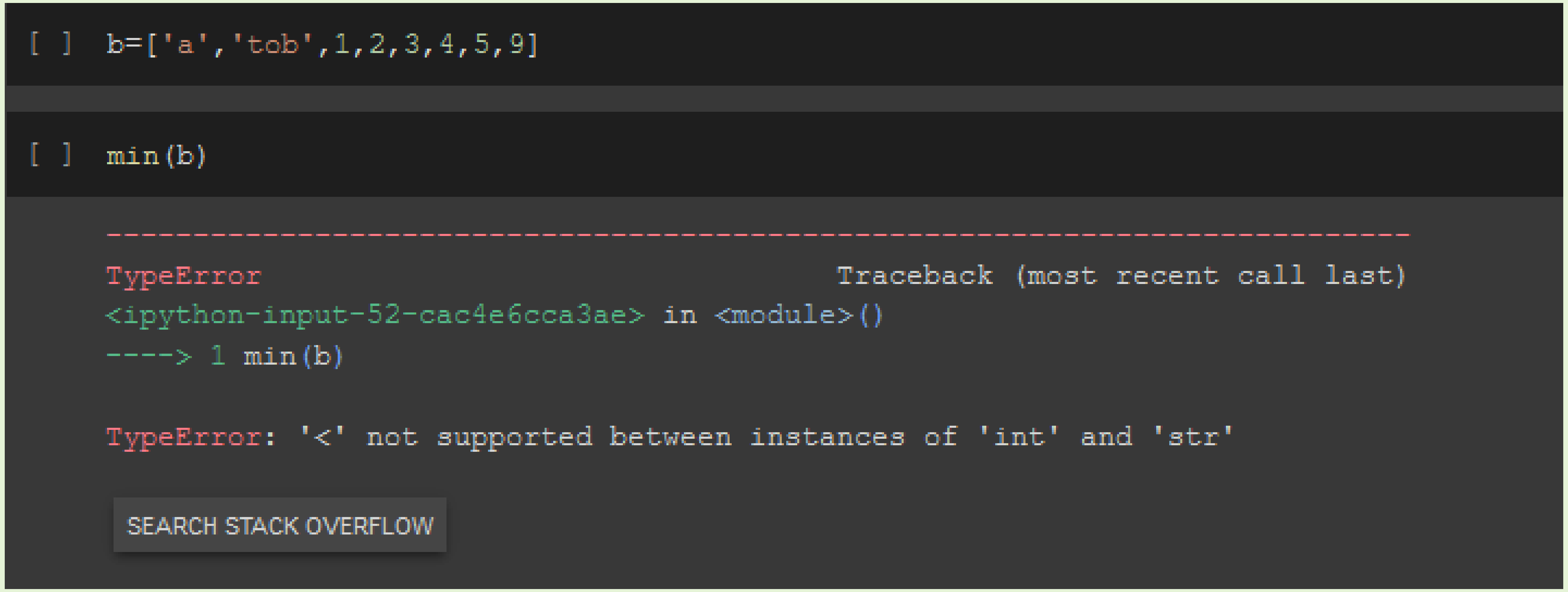
- used to find the smallest and largest values in a list



```
[ ]  a=[0.0125,0.11,0.23,1,2,4,5,1,2,3,1]

[ ]  min(a)

     0.0125

[ ]  max(a)

     5
```

MIN AND MAX

- When the list contains characters/strings, it will not be able to find the largest or smallest value



```
[ ]  b=['a','tob',1,2,3,4,5,9]
```

```
[ ]  min(b)
```

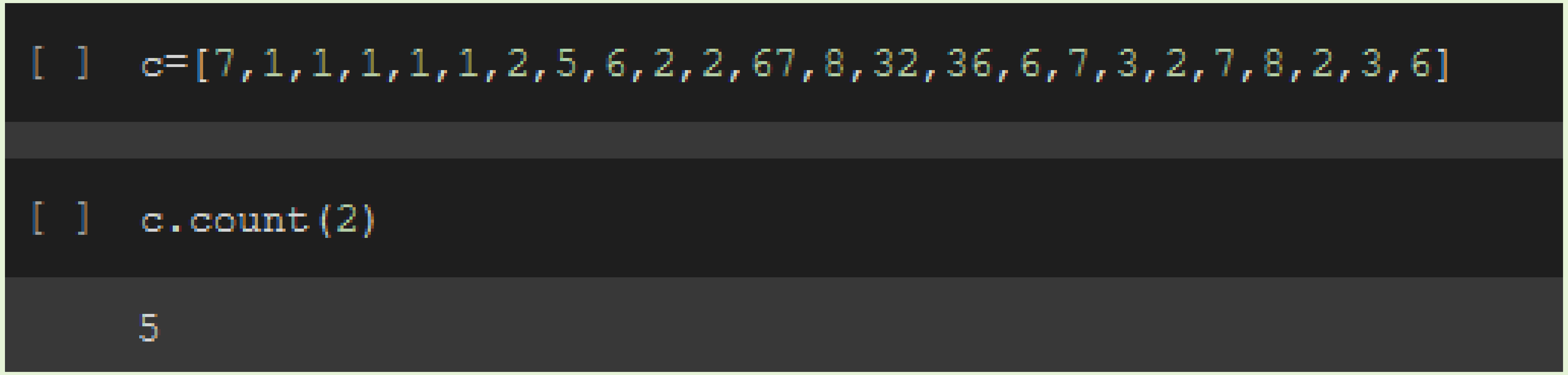
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-52-cac4e6cca3ae> in <module>()
----> 1 min(b)

TypeError: '<' not supported between instances of 'int' and 'str'
```

OPERATION STRING, LIST, SET

COUNT

COUNT

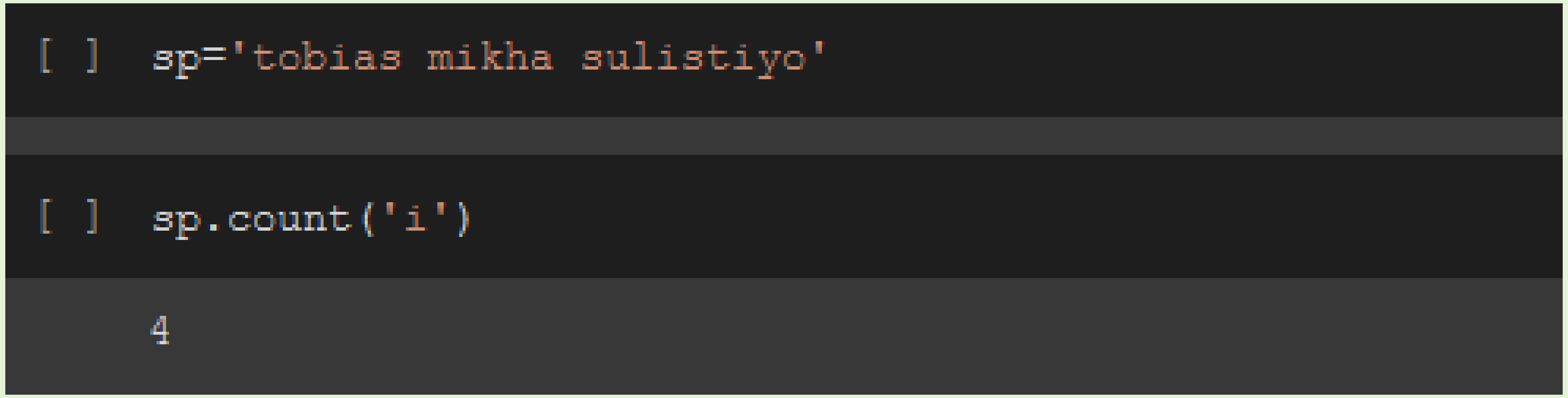
- Count is fuction to calculate a specific value



```
[ ]  c=[7,1,1,1,1,1,2,5,6,2,2,67,8,32,36,6,7,3,2,7,8,2,3,6]

[ ]  c.count(2)

     5
```

COUNT

- The count function can also be used to find the number of letters or words that appear in a sentence in a string format



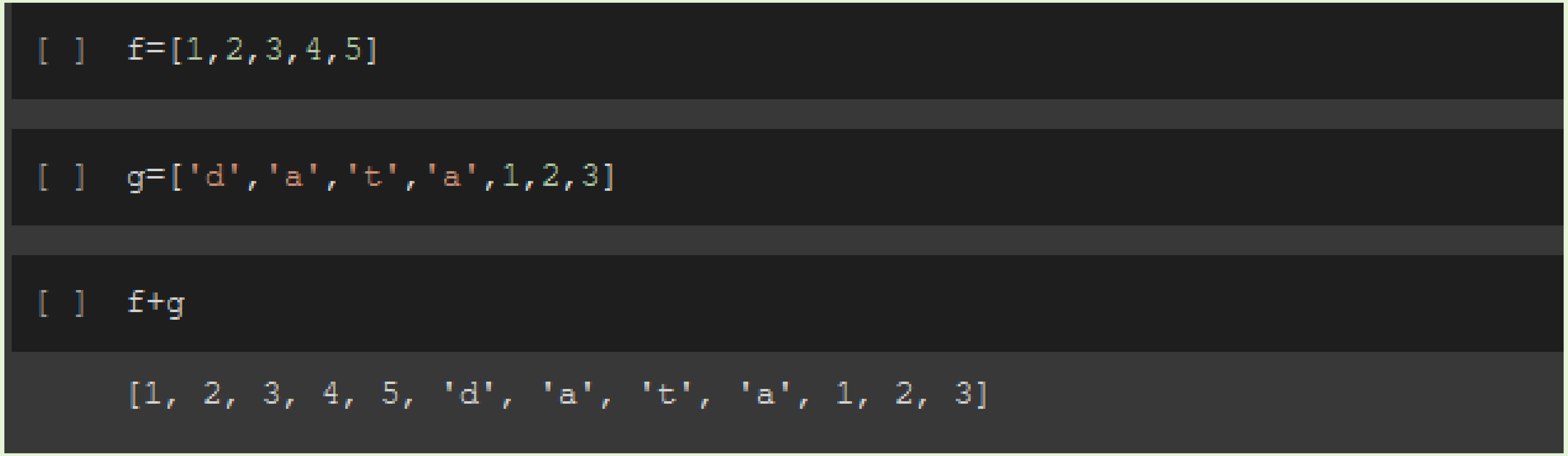
```
[ ]  sp='tobias mikha sulistiyo'

[ ]  sp.count('i')

     4
```

OPERATION STRING, LIST, SET

JOIN AND REPLICATION

JOIN

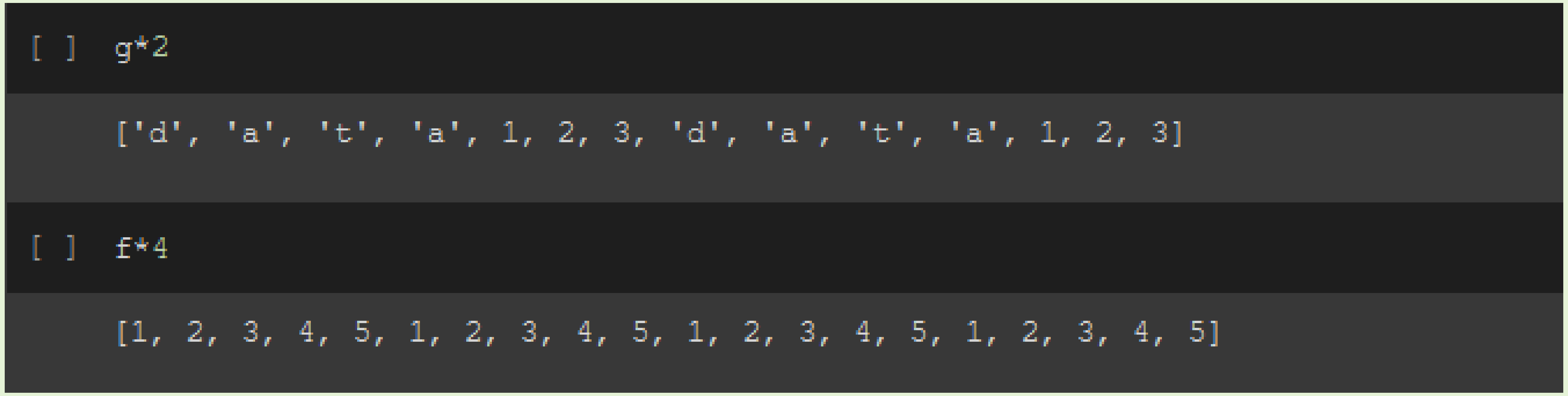
- For the concatenation function is like a mathematical operation but what is operated is a list



```
[ ]  f=[1,2,3,4,5]

[ ]  g=['d','a','t','a',1,2,3]

[ ]  f+g

     [1, 2, 3, 4, 5, 'd', 'a', 't', 'a', 1, 2, 3]
```

REPLICATION

- Replication or repetition is to repeat the amount of data as much as desired. For replication operations use (*)



```
[ ]  g*2

     ['d', 'a', 't', 'a', 1, 2, 3, 'd', 'a', 't', 'a', 1, 2, 3]

[ ]  f*4

     [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

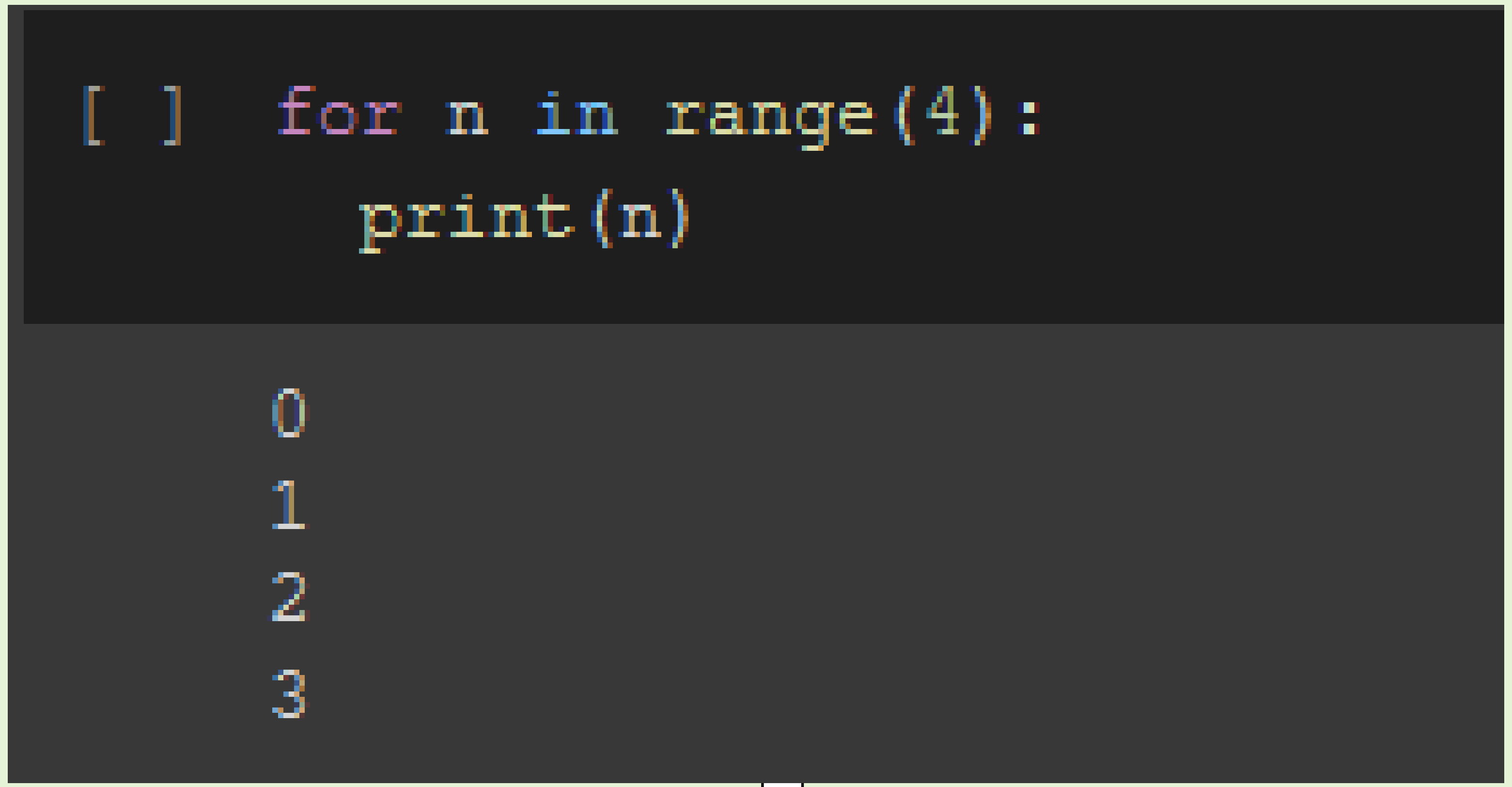
OPERATION STRING, LIST, SET

RANGE

RANGE

- The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

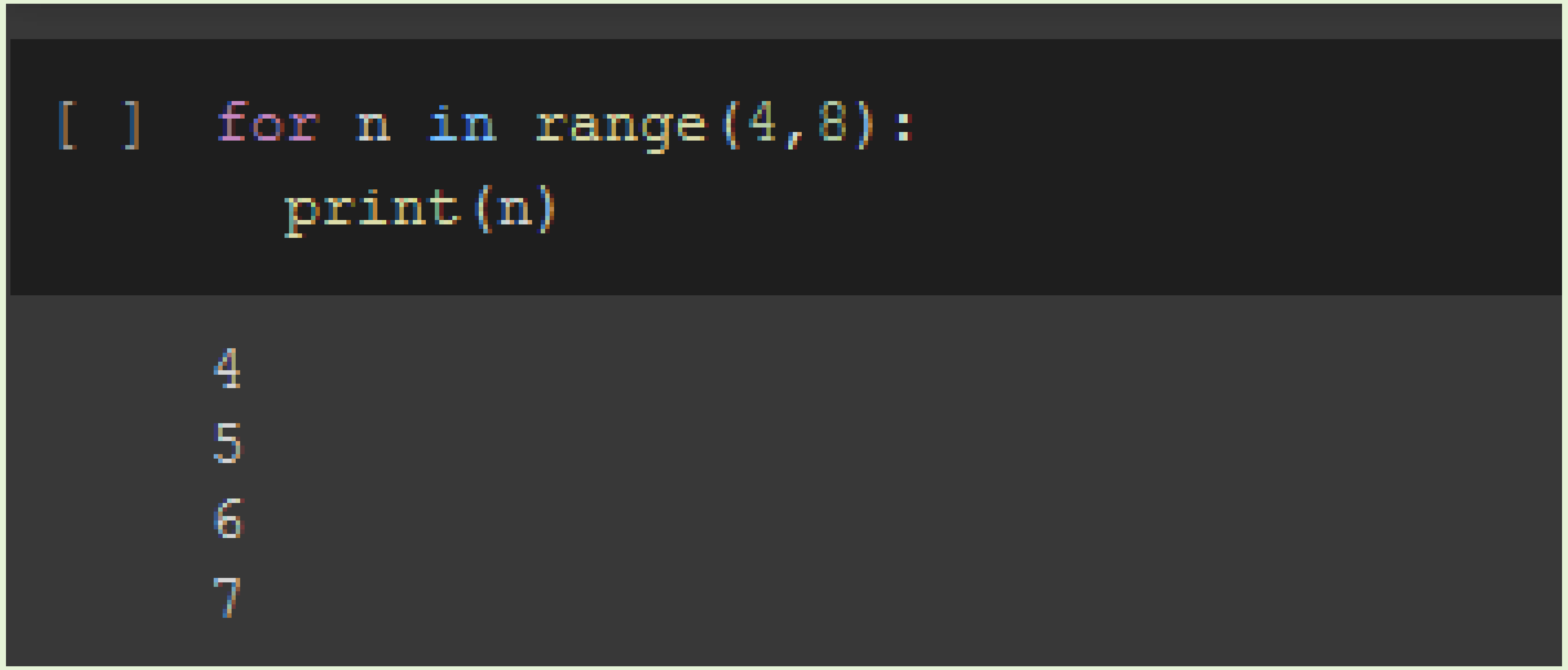
Function:
range(start,stop,step)



```
[ ]  for n in range(4):
       print(n)

     0
     1
     2
     3
```

RANGE

Displays the results range from the beginning of the specified number (0 default) to end. Or in python the calculation is -1 of the desired list

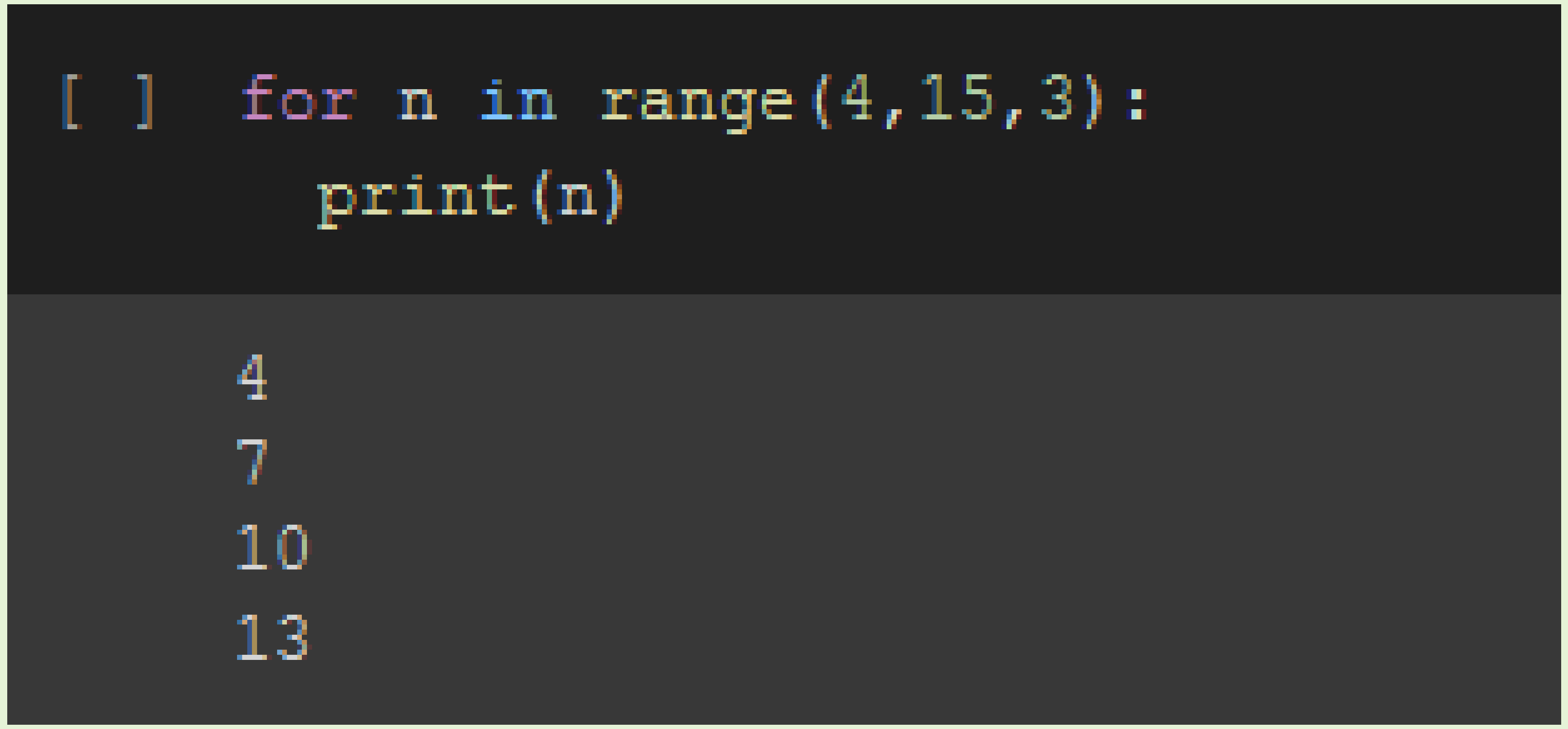


```
[ ]  for n in range(4,8):
       print(n)

     4
     5
     6
     7
```

RANGE

- Displays the results range from the beginning of the specified number (0 default) to end with step.

Function:
range(start,stop,step)



```
[ ]  for n in range(4,15,3):
       print(n)
```

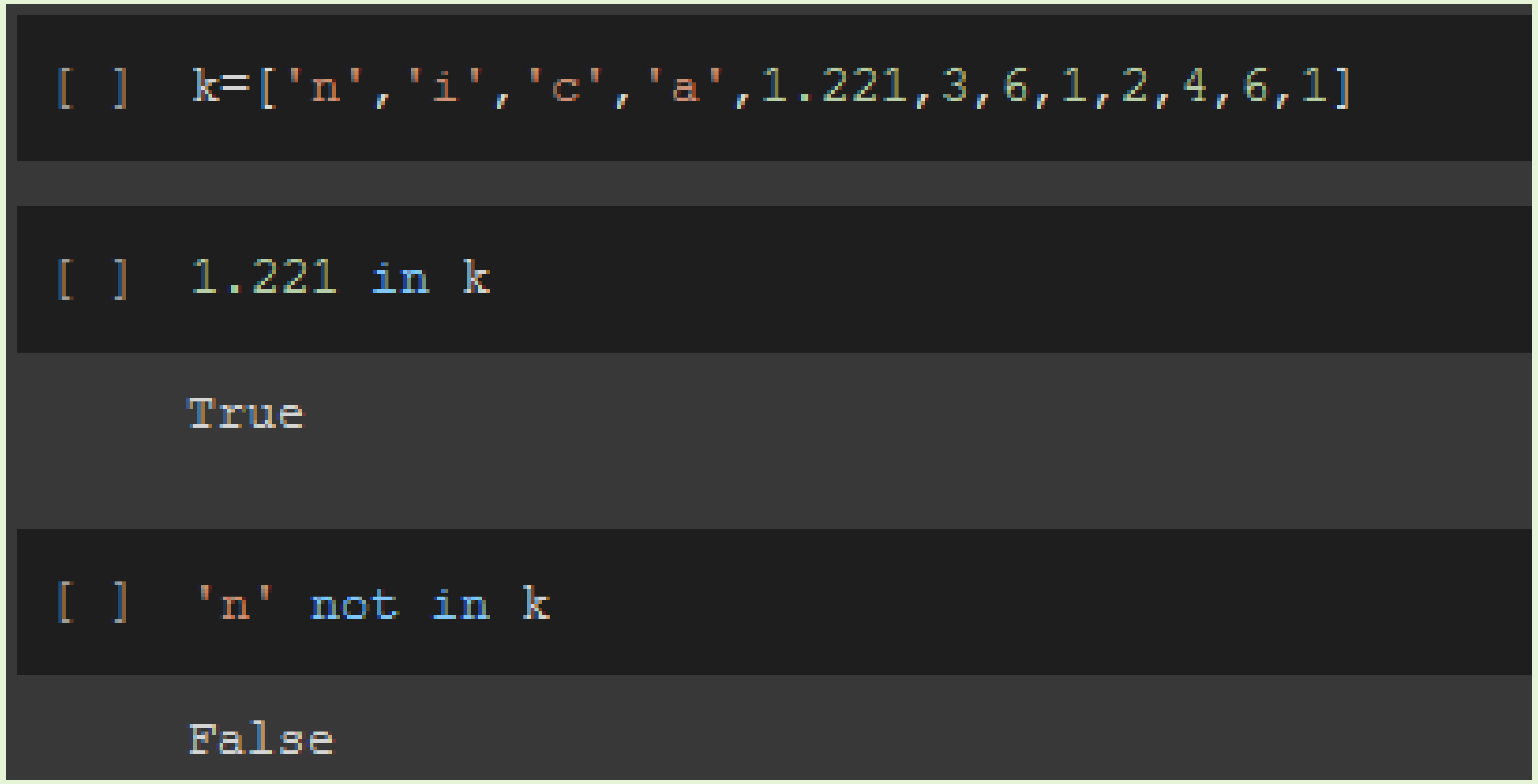
```
4
7
10
13
```

OPERATION STRING, LIST, SET

IN AND NOT IN

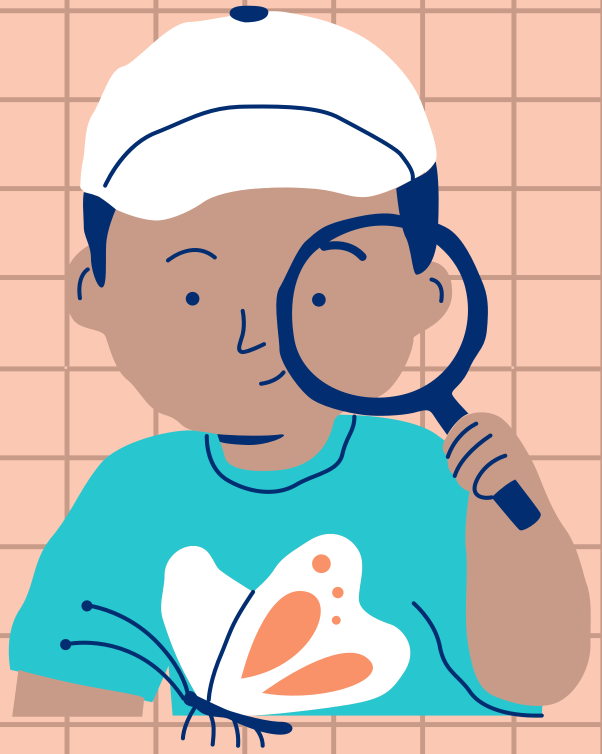
IN AND NOT IN

To display the value in the list or variable data is true or false



```
[ ]  k=['n','i','c','a',1.221,3,6,1,2,4,6,1]

[ ]  1.221 in k

     True

[ ]  'n' not in k

     False
```

Operator,Operand,and Expression



What is Operator in Python?

Operator

Operators are special symbols in Python that carry out arithmetic or logical computation

Operand

a + b

Addition (+) is one of the operators in Python

How about Operand ?

a + b

a and b
are the Operand

Operand

The value that the operator operates on is called the operand.

How about Expression ?

Expression

OA sequence of operands and operators is called an expression. Python supports many operators for combining data objects into expressions.

Example of Expression

a + b

PYTHON OPERATORS

Python has several types of operators, these are some of them :

Arithmetic Operators

Comparison Operators

Logical Operators

Assignment Operators

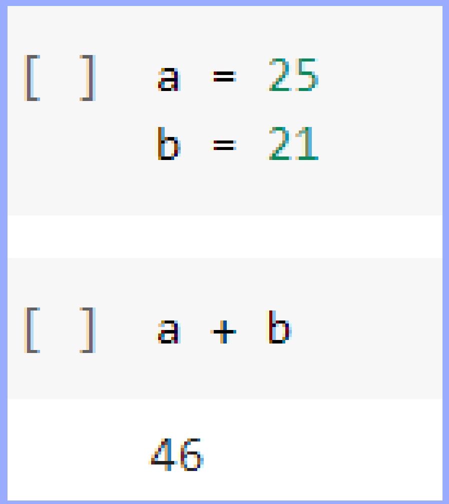
ARITHMETIC OPERATORS

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, etc.

Operator	Symbol	Meaning
Addition	+	Add two operands or more, or used as unary plus
Subtraction	-	Subtract right operand from the left or used as unary minus
Multiplication	*	Multiply two operands or more
Division	/	Divide left operand by the right one
Exponentiation	**	Left operand raised to power of right operand
Floor Division	//	Division that results into whole number adjusted to the left in the number line
Modulo	%	Return the remainder of the division of left operand by the right

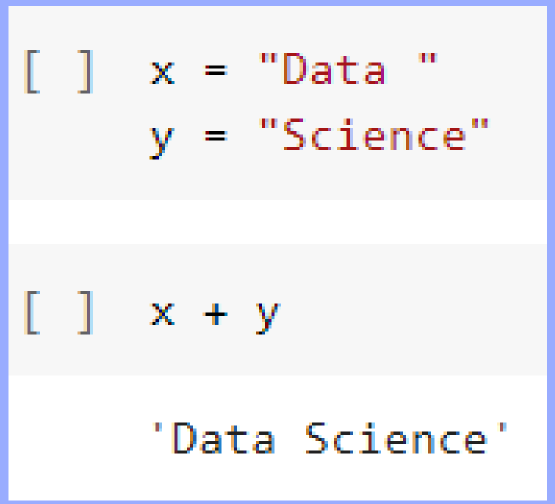
+

Example :



```
[ ]  a = 25
     b = 21

[ ]  a + b

     46
```

Number

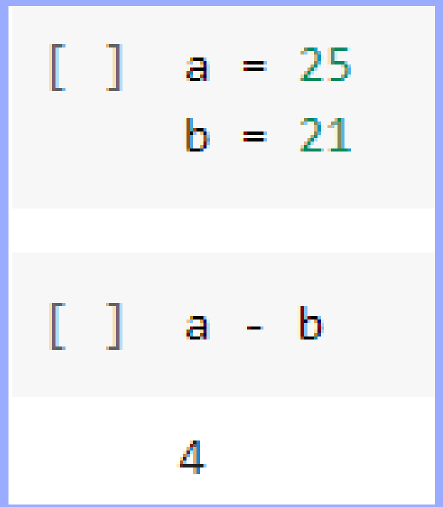


```
[ ]  x = "Data "
     y = "Science"

[ ]  x + y

     'Data Science'
```

String

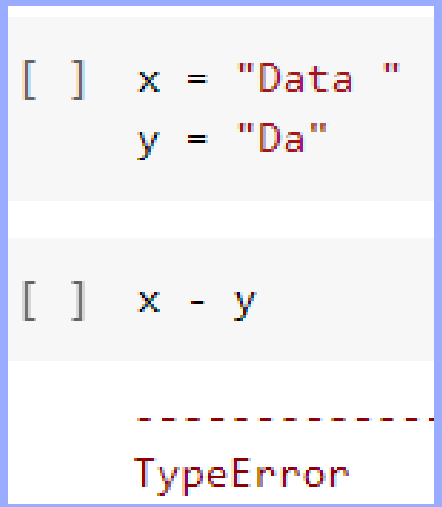
-

Example :



```
[ ]  a = 25
     b = 21

[ ]  a - b

     4
```

Number

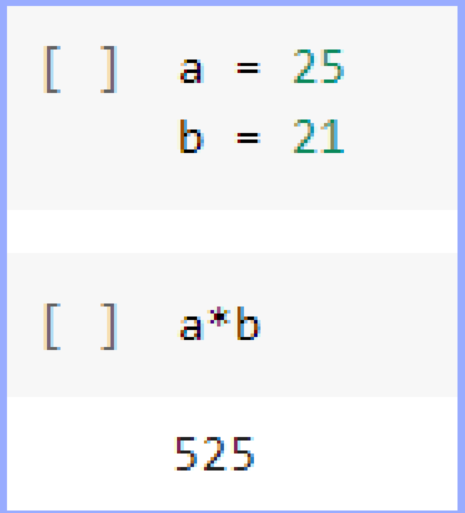


```
[ ]  x = "Data "
     y = "Da"

[ ]  x - y

     -------------
     TypeError
```

String

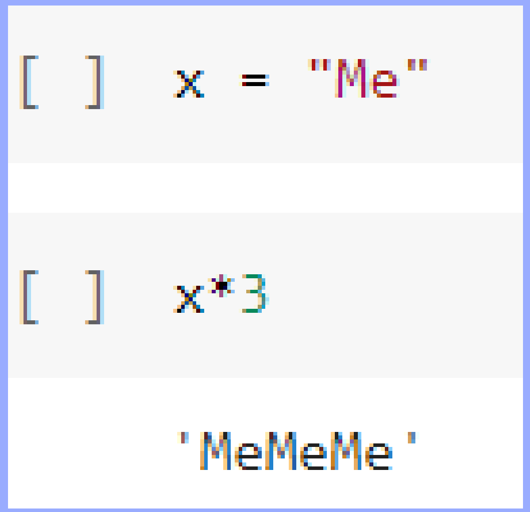
*

Example :



```
[ ]  a = 25
     b = 21

[ ]  a*b

     525
```

Number

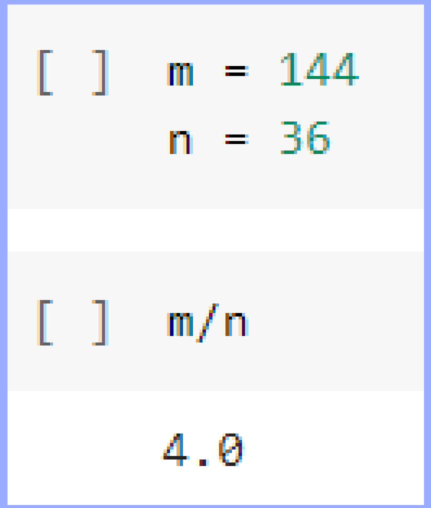


```
[ ]  x = "Me"

[ ]  x*3

     'MeMeMe'
```

String

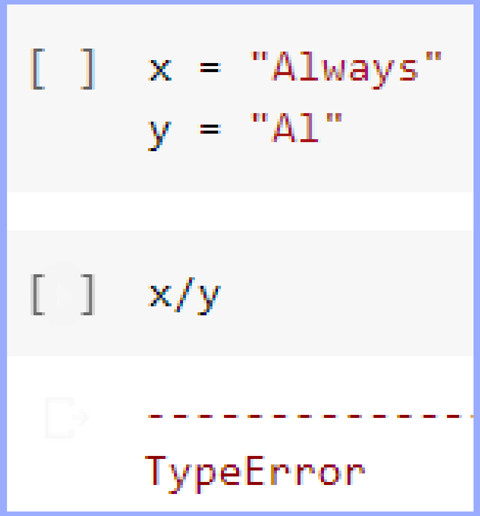
/

Example :



```
[ ]  m = 144
     n = 36

[ ]  m/n

     4.0
```

Number



```
[ ]  x = "Always"
     y = "Al"

[ ]  x/y

     -------------
     TypeError
```

String

Notes

1. The multiplication (*) operator can also be used for string data types to print as many string as multiplier number
2. The result of the division (/) operation will always be float type
3. The subtaction and division operator cannot be used for string types

**

Example :

```
[ ]  m = 4

[ ]  m**2

     16
```

Number

```
[ ]  n = "day"

[ ]  n**5

     ------------
     TypeError
```

String

//

Example :

```
[ ]  a = 22
     b = 6

[ ]  a // b

     3
```

```
[ ]  c = 31.5
     d = 7

[ ]  c // d

     4.0
```

%

Example :

```
[ ]  a = 22
     b = 6

[ ]  a%b

     4
```

```
[ ]  c = 31.5
     d = 7

[ ]  c%d

     3.5
```

Notes

1.Tips : to find the root value, use decimal powers

2.The Floor Division operator will return a result with a float if one of its operands is float type

3.The Exponentiation, Floor Division, and Modulo operator cannot be used for string types

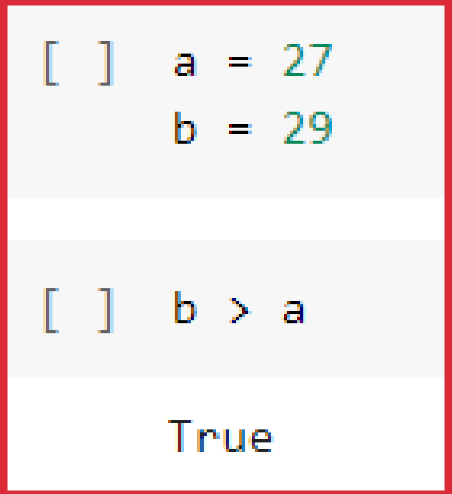
COMPARISON OPERATORS

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, etc.

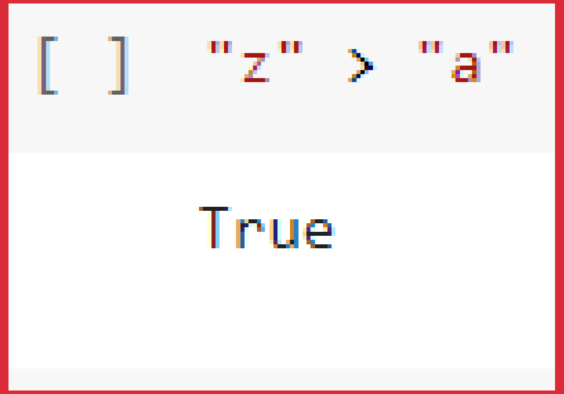
Operator	Symbol	Meaning
Greater than	>	Return True if left operand is greater than the right
Less than	<	Return True if left operand is less than the right
Equal to	==	Return True if both operands ar equal
Not Equal to	!=	Return True if operands are not equal
Greater than or equal to	>=	Return True if left operand is greater than or equal to the right
Less than or equal to	<=	Return True if left operand is less than or equal to the right

>

Example :



```
[ ]  a = 27
     b = 29

[ ]  b > a

     True
```

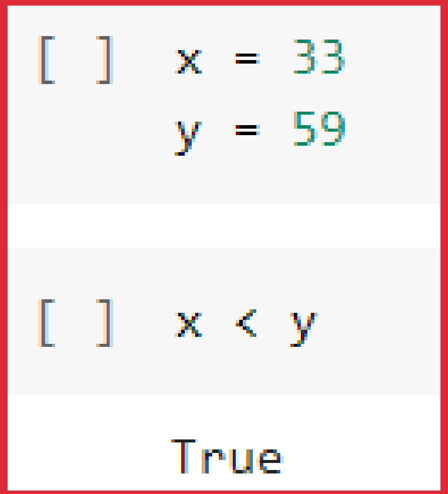


```
[ ]  "z" > "a"

     True
```

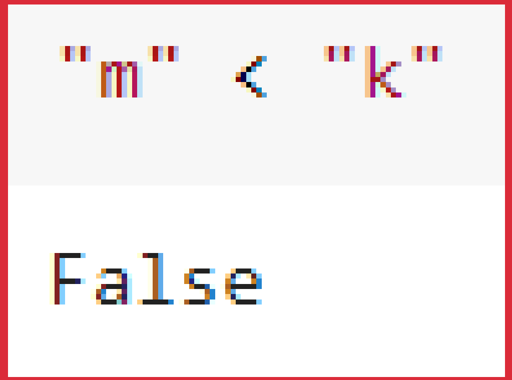
Number String

<

Example :



```
[ ]  x = 33
     y = 59

[ ]  x < y

     True
```

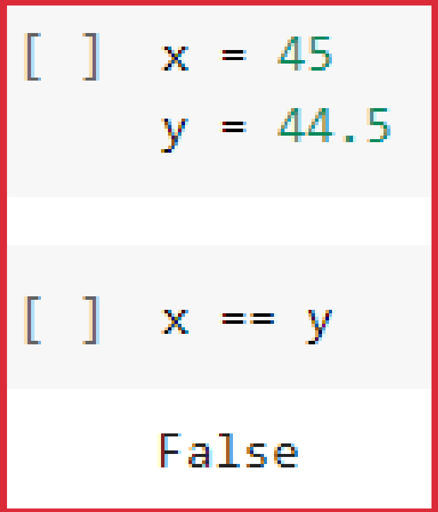


```
"m" < "k"

False
```

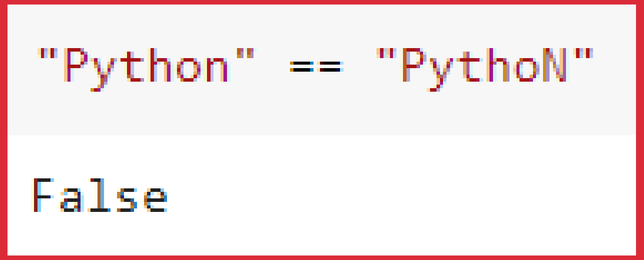
Number String

==

Example :



```
[ ]  x = 45
     y = 44.5

[ ]  x == y

     False
```

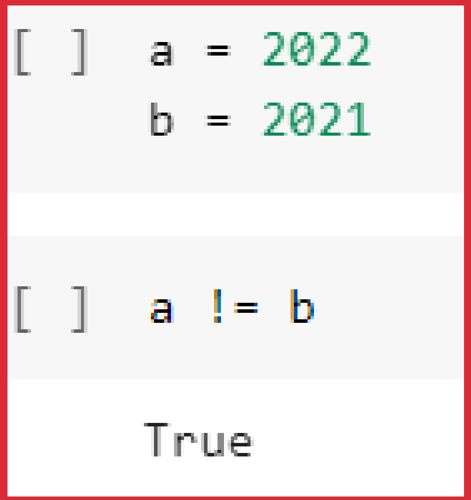


```
"Python" == "PythoN"

False
```

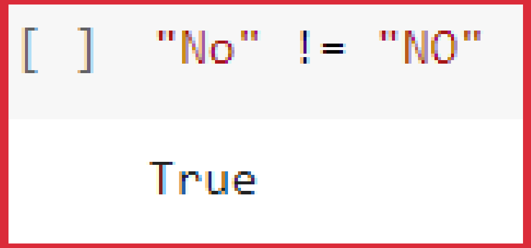
Number String

!=

Example :



```
[ ]  a = 2022
     b = 2021

[ ]  a != b

     True
```

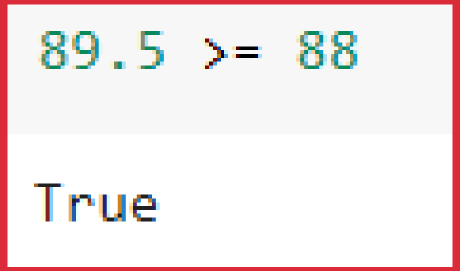


```
[ ]  "No" != "NO"

     True
```

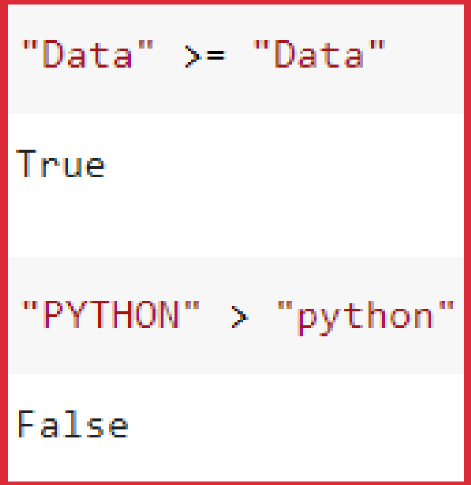
Number String

>=

Example :



```
89.5 >= 88

True
```

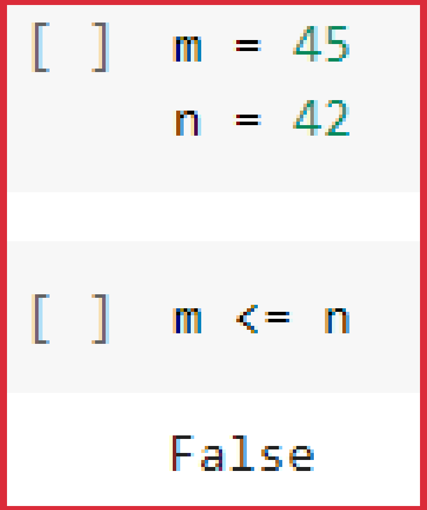


```
"Data" >= "Data"

True

"PYTHON" > "python"

False
```

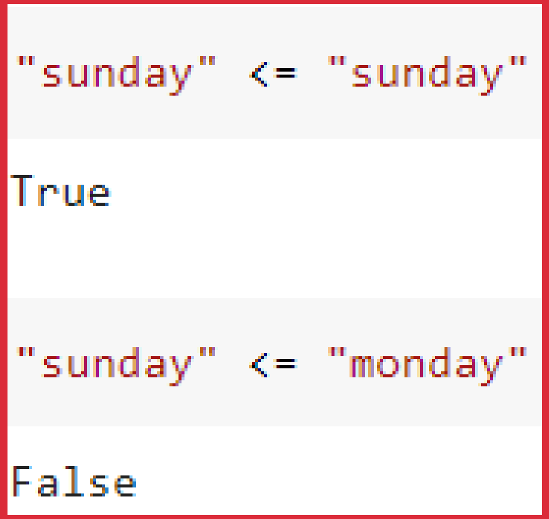
Number String

<=

Example :



```
[ ]  m = 45
     n = 42

[ ]  m <= n

     False
```



```
"sunday" <= "sunday"

True

"sunday" <= "monday"

False
```

Number String

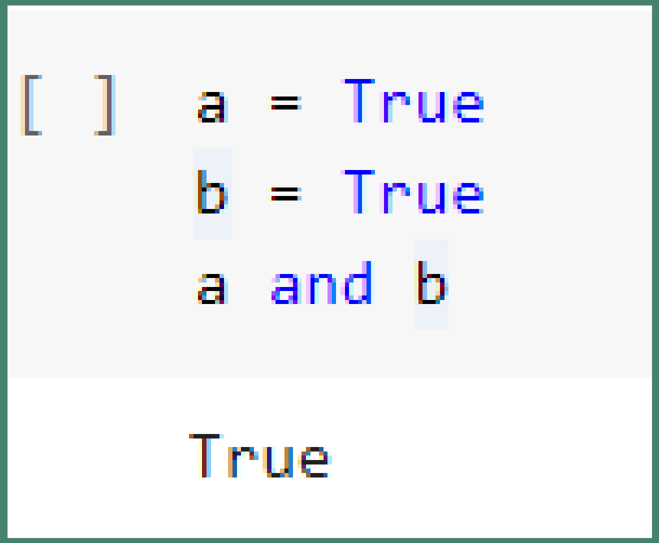
LOGICAL OPERATORS

In Python, Logical operators are used on conditional statements (either True or False). They perform Logical AND, Logical OR and Logical NOT operations

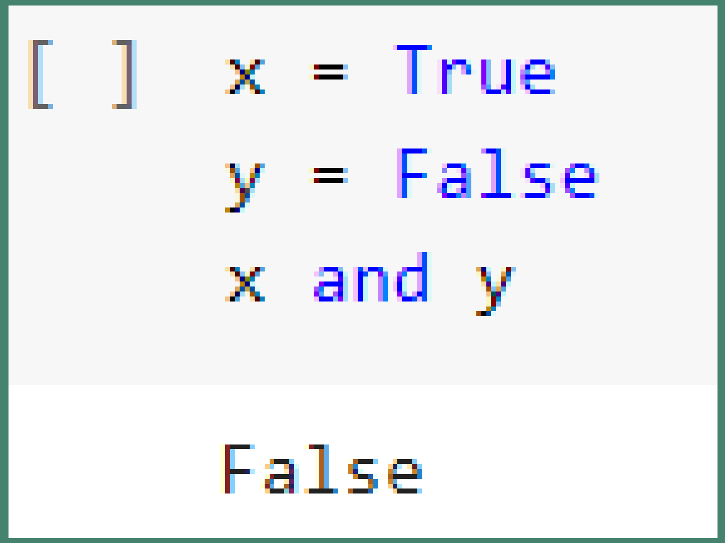
Operator	Meaning
and	Return True, only if both operands are true and return False if one or both operands are false
or	Return True, if one or both of the operands are true and return False only if both operands are false
not	Return True if operand is false (complements the operand)

and

Example :



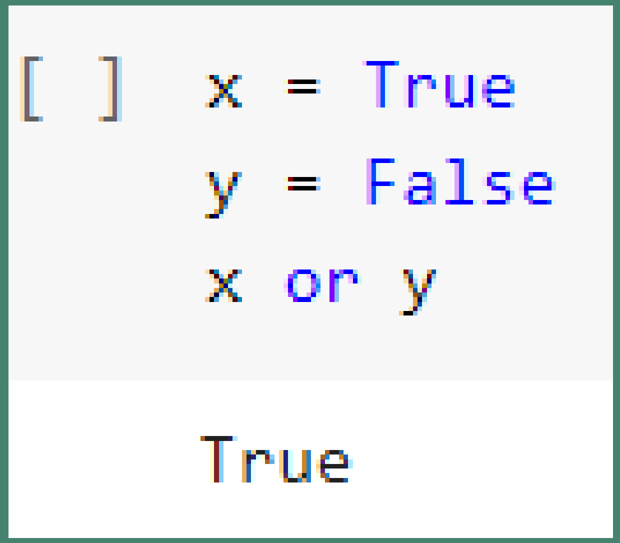
```
[ ]  a = True
     b = True
     a and b

     True
```



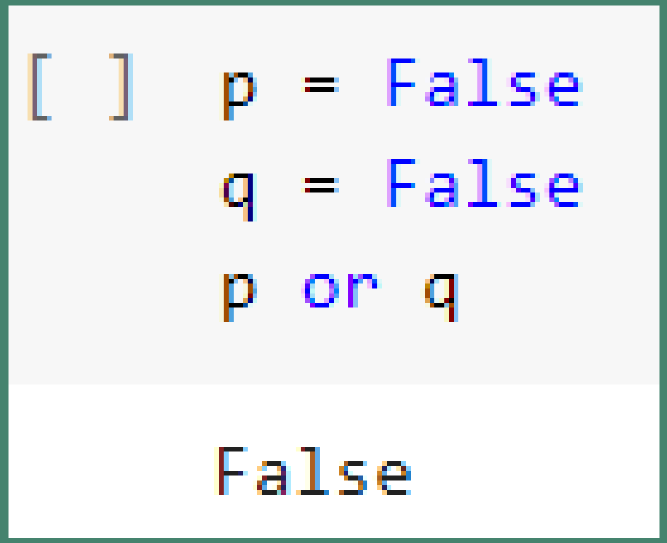
```
[ ]  x = True
     y = False
     x and y

     False
```

or

Example :



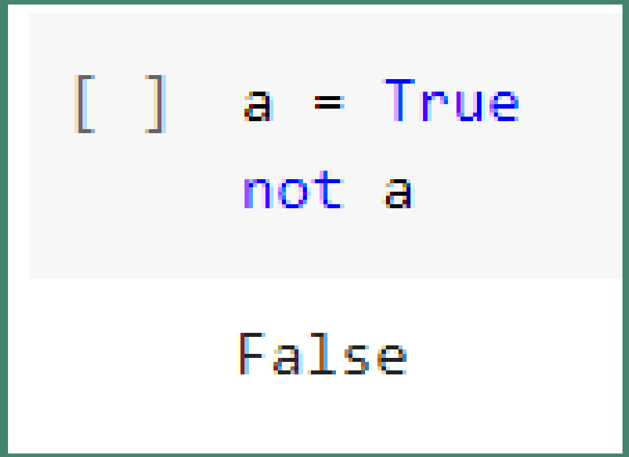
```
[ ]  x = True
     y = False
     x or y

     True
```



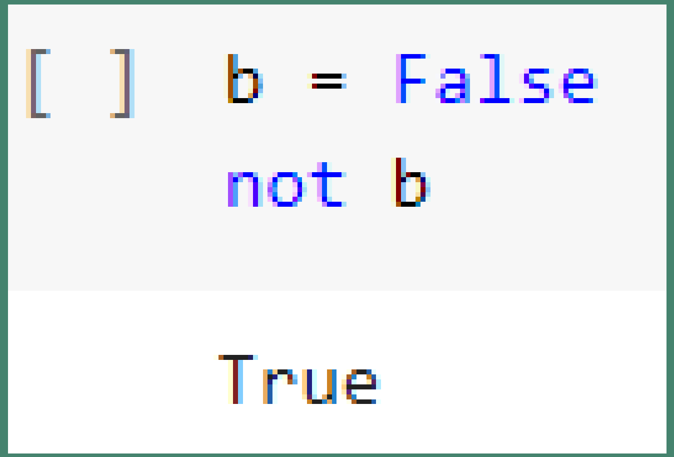
```
[ ]  p = False
     q = False
     p or q

     False
```

not

Example :



```
[ ]  a = True
     not a

     False
```



```
[ ]  b = False
     not b

     True
```

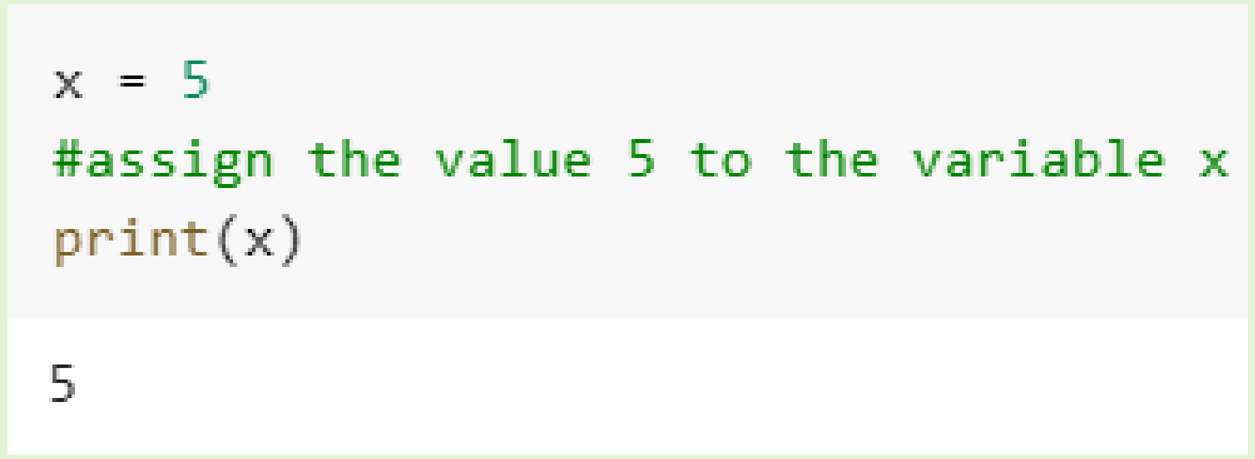
ASSIGNMENT OPERATORS

Assignment operators are used in Python to assign values to variables

=

Used to assign the value
to variable

Example :

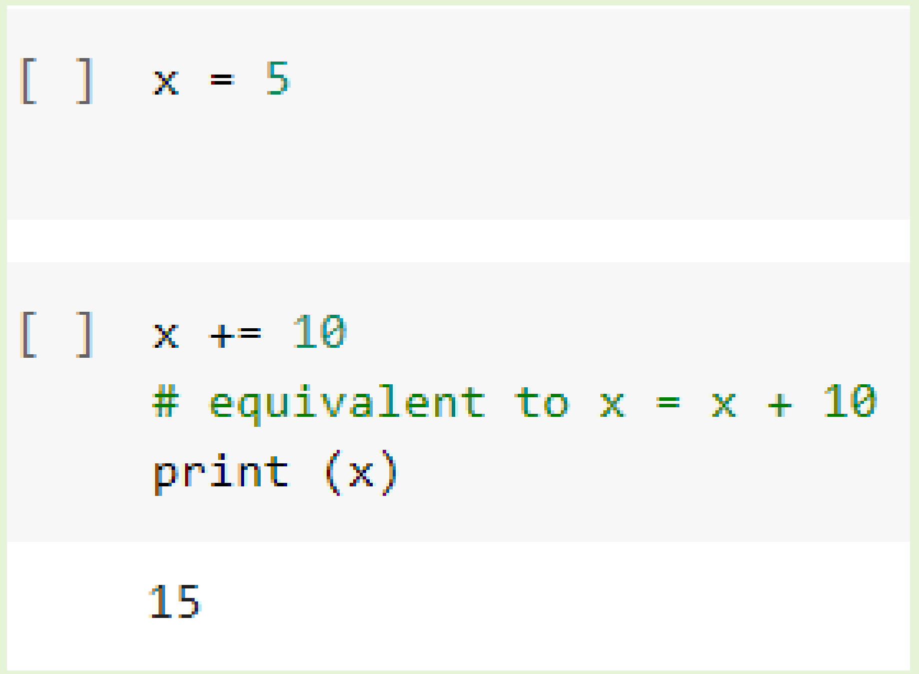


```
x = 5
#assign the value 5 to the variable x
print(x)
```

```
5
```

+=

Example :



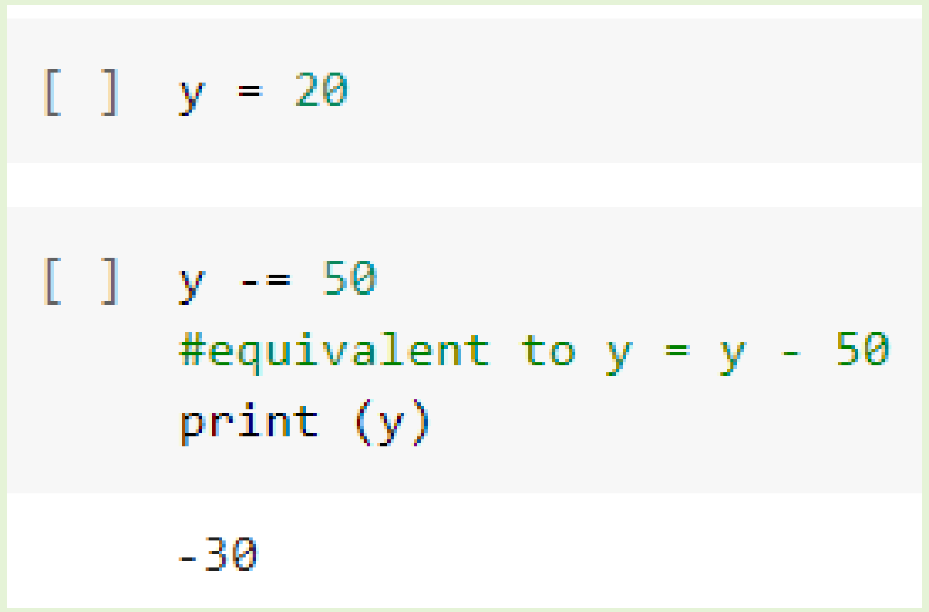
```
[ ]  x = 5
```

```
[ ]  x += 10
     # equivalent to x = x + 10
     print (x)
```

```
     15
```

-=

Example :



```
[ ]  y = 20
```

```
[ ]  y -= 50
     #equivalent to y = y - 50
     print (y)
```

```
     -30
```

*=

Example :

```
[ ]  a = 20
```

```
[ ]  a *= 2
     #equivalent to a = a*20
     print (a)
```

```
     40
```

**=

Example :

```
[ ]  c = 3
```

```
[ ]  c **= 4
     #equivalent to c = c**3
     print (c)
```

```
     81
```

/=

Example :

```
[ ]  b = 10
```

```
[ ]  b /= 5
     #equivalent to b = b/5
     print (b)
```

```
     2.0
```

//=

Example :

```
[ ]  d = 34
```

```
[ ]  d //= 3
     #equivalent to d = d//3
     print (d)
```

```
     11
```

%=

Example :

```
[ ]  e = 21
```

```
[ ]  e %= 10
     #equivalent to e = e%10
     print (e)
```

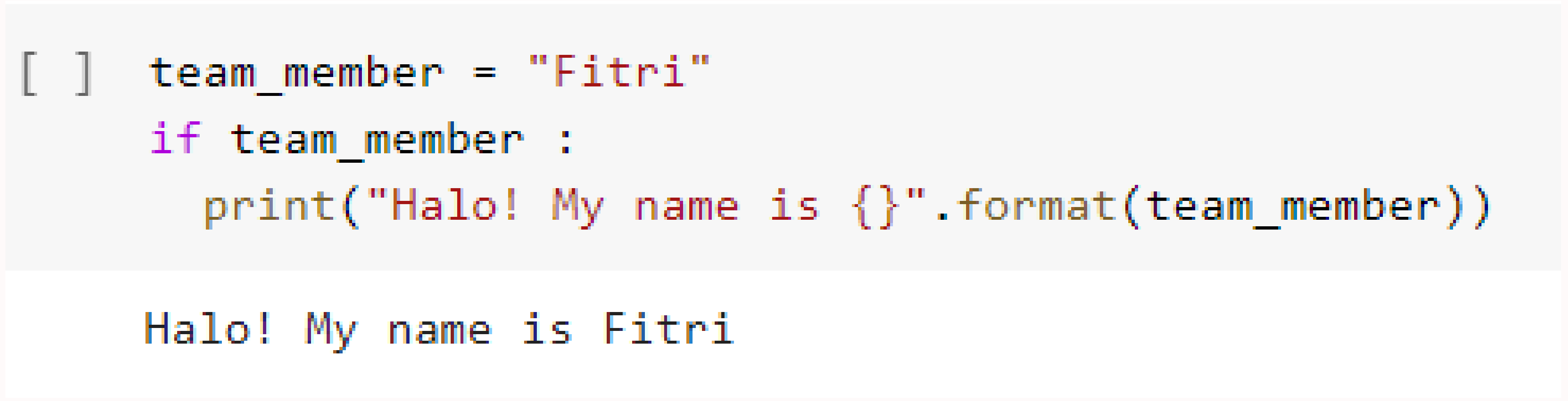
```
     1
```

CONDITIONAL EXPRESSIONS

A conditional expression in Python is an expression (in other words, a piece of code that evaluates to a result) whose value depends on a condition.

IF

- The statement will only be executed depending on whether the condition expression evaluates to True. If the condition expression evaluates to False, the statement is not executed.
- In Python, the body of the if statement is indicated by the indentation. The body starts with an indentation and the first unindented line marks the end.
- Python interprets non-zero values as True. None and 0 are interpreted as False

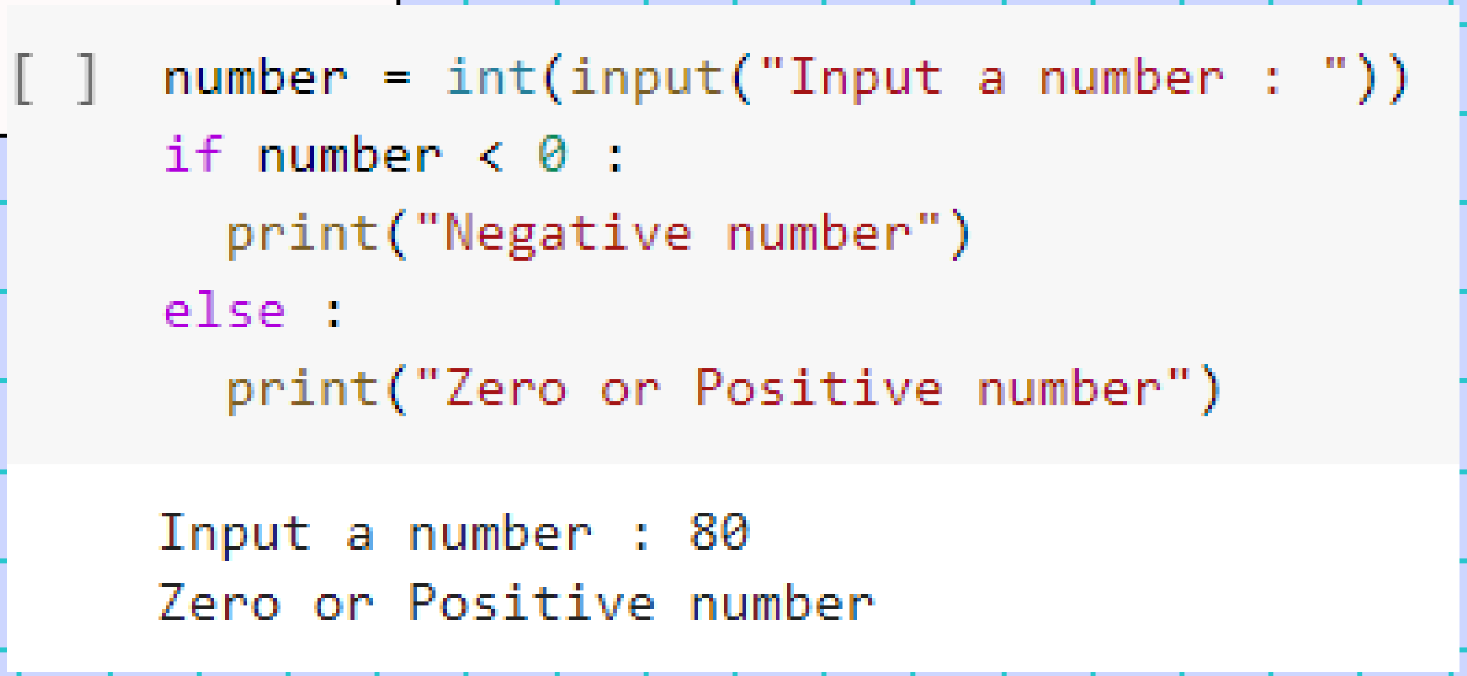


```
[ ]  team_member = "Fitri"
     if team_member :
       print("Halo! My name is {}".format(team_member))
```

```
Halo! My name is Fitri
```

IF...ELSE

- The if..else statement evaluates test expression and will execute the body of if only when the test condition is True.
- If the condition is False, the body of else is executed
- Indentation is used to separate the blocks.

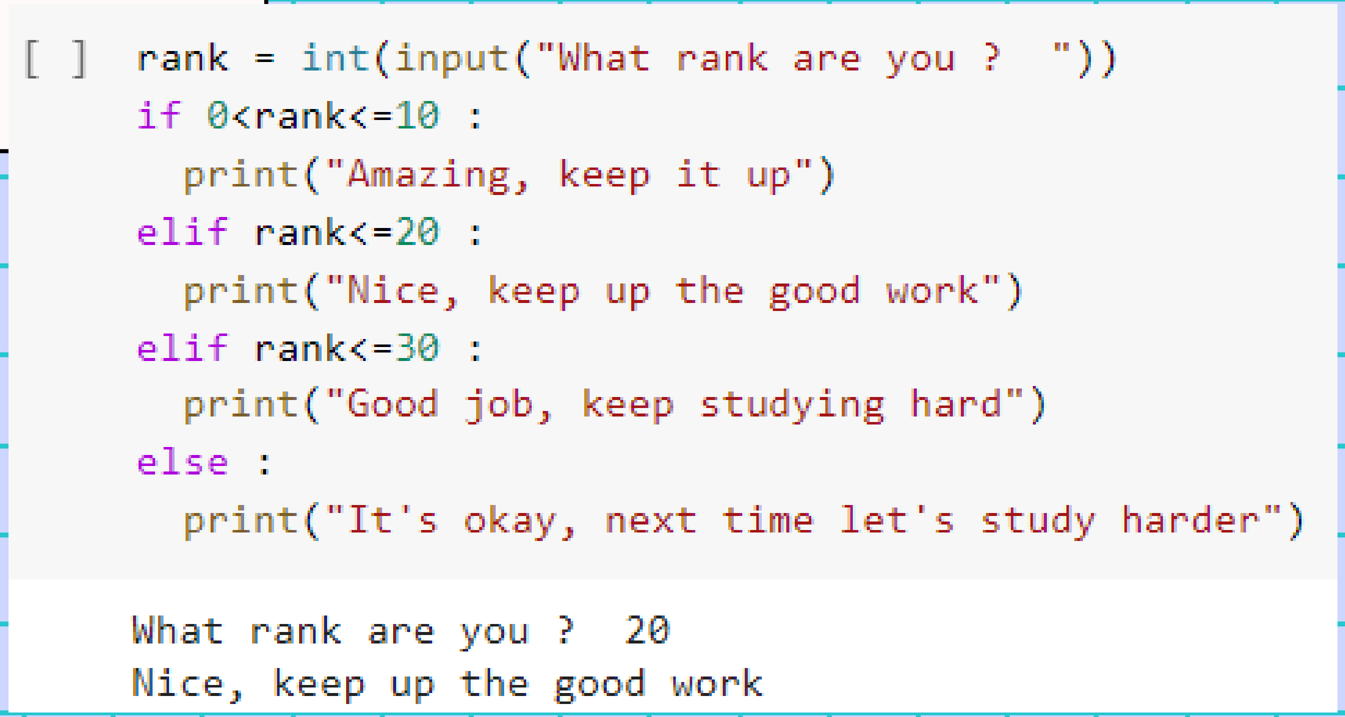


```
[ ]  number = int(input("Input a number : "))
     if number < 0 :
       print("Negative number")
     else :
       print("Zero or Positive number")
```

```
Input a number : 80
Zero or Positive number
```

IF..ELIF..ELSE

- The elif is short for else if. It allows us to check for multiple expressions.
- If the condition for if is False, it checks the condition of the next elif block and so on.
- If all the conditions are False, the body of else is executed.
- Only one block among the several if...elif...else blocks is executed according to the condition.
- The if block can have only one else block. But it can have multiple elif blocks.



```
[ ]  rank = int(input("What rank are you ?  "))
     if 0<rank<=10 :
       print("Amazing, keep it up")
     elif rank<=20 :
       print("Nice, keep up the good work")
     elif rank<=30 :
       print("Good job, keep studying hard")
     else :
       print("It's okay, next time let's study harder")
```

```
What rank are you ?  20
Nice, keep up the good work
```

Thanks!

That's our powerpoint for modify and processing data in python. For the github, you can access in the link on our LinkedIn comment section!

See you next time!

Matplotlib Group - Data Science Track B