Tuples.

11.2.4.1 create a tuple;

11.2.2.1 perform access to the elements of strings, lists, tuples;

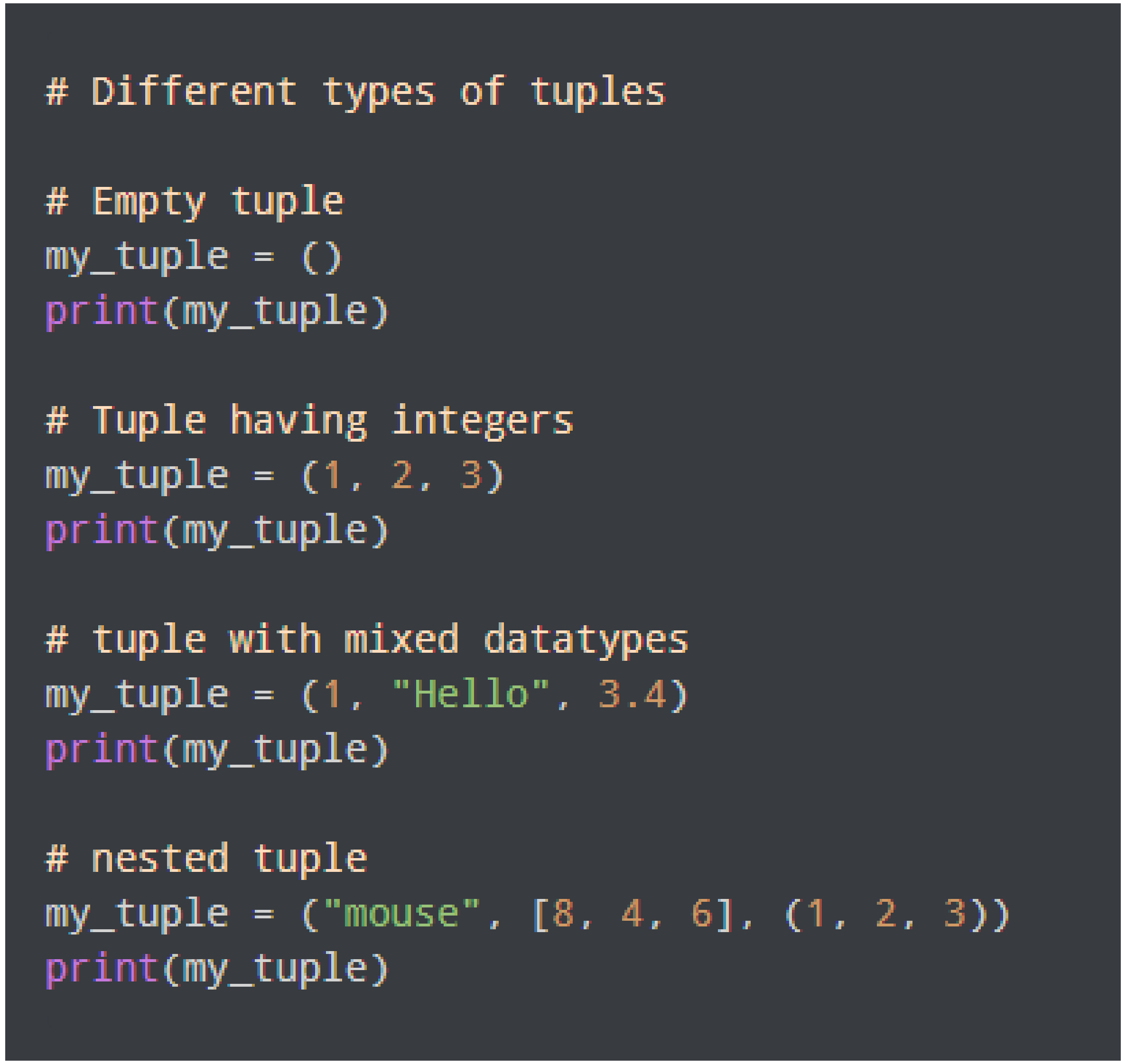
11.2.4.2 convert from one data structure to another;

11.4.3.2 solve applied problems of various subject areas.

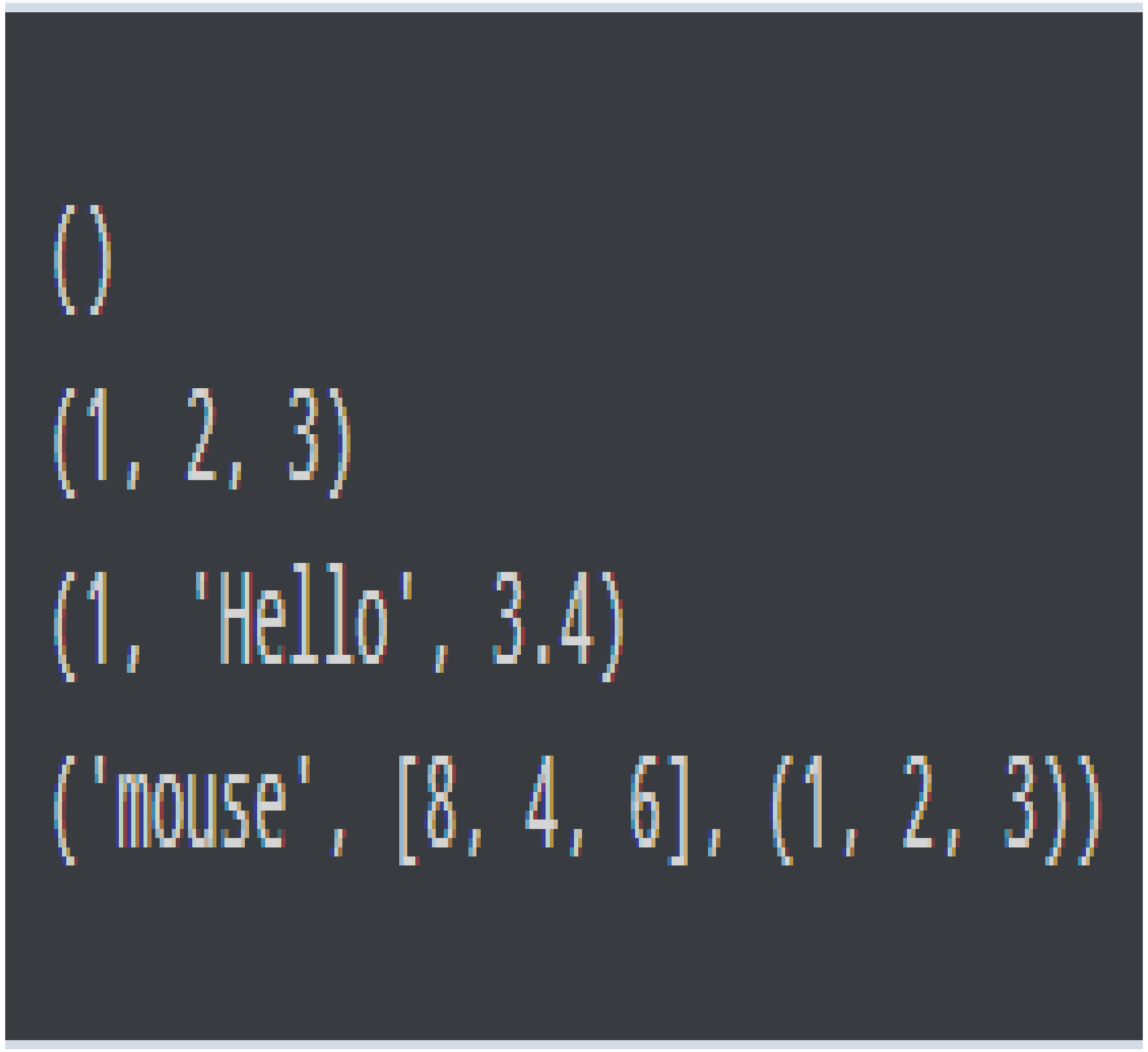
Tuples

- A tuple in Python is **similar to a list**. The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas we can change the elements of a list.
- **Ordered** - When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.
- **Unchangeable** - Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.
- **Allow Duplicates** - Since tuples are indexed, they can have items with the same value:

Creating a Tuple

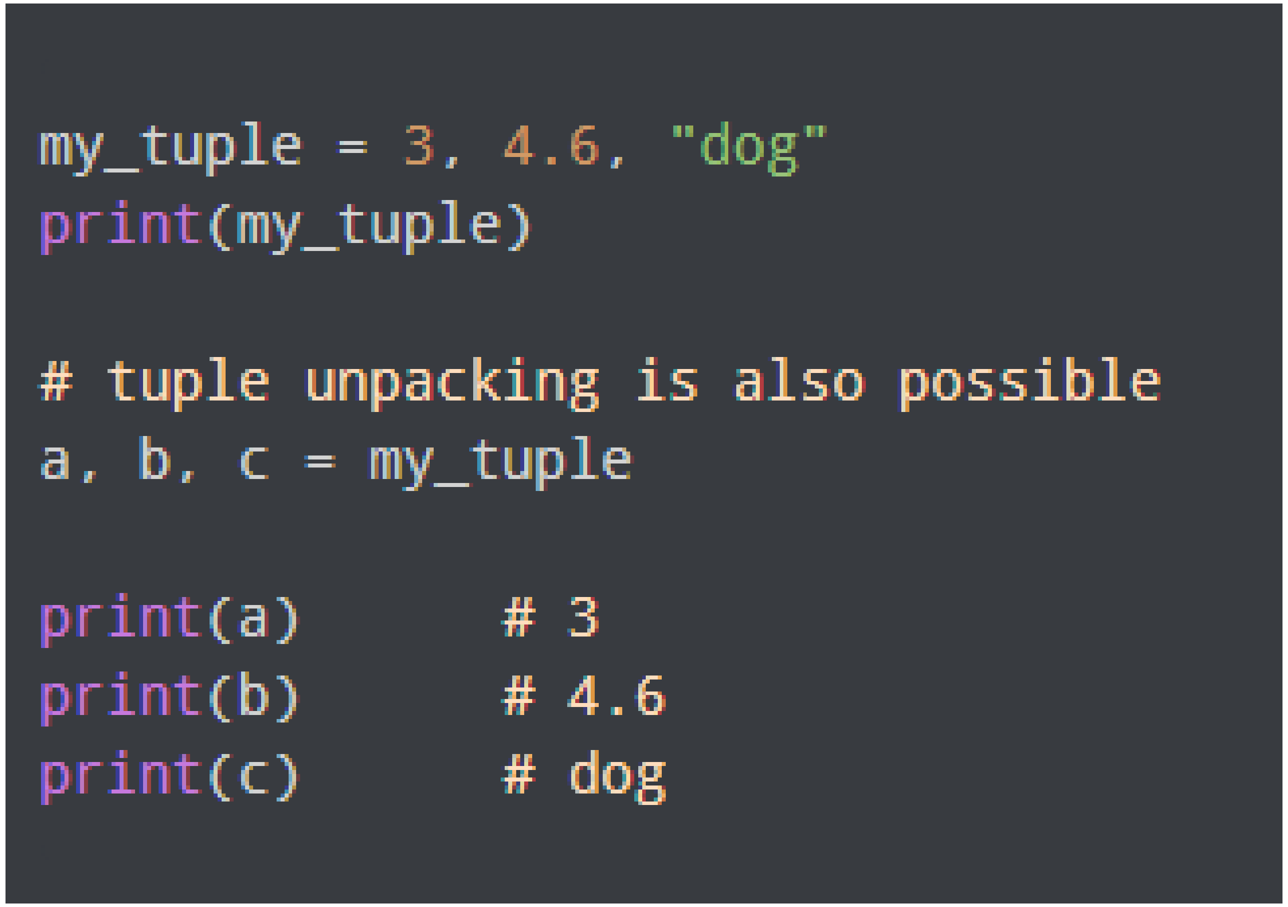


```python
# Different types of tuples

# Empty tuple
my_tuple = ()
print(my_tuple)

# Tuple having integers
my_tuple = (1, 2, 3)
print(my_tuple)

# tuple with mixed datatypes
my_tuple = (1, "Hello", 3.4)
print(my_tuple)

# nested tuple
my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
print(my_tuple)
```

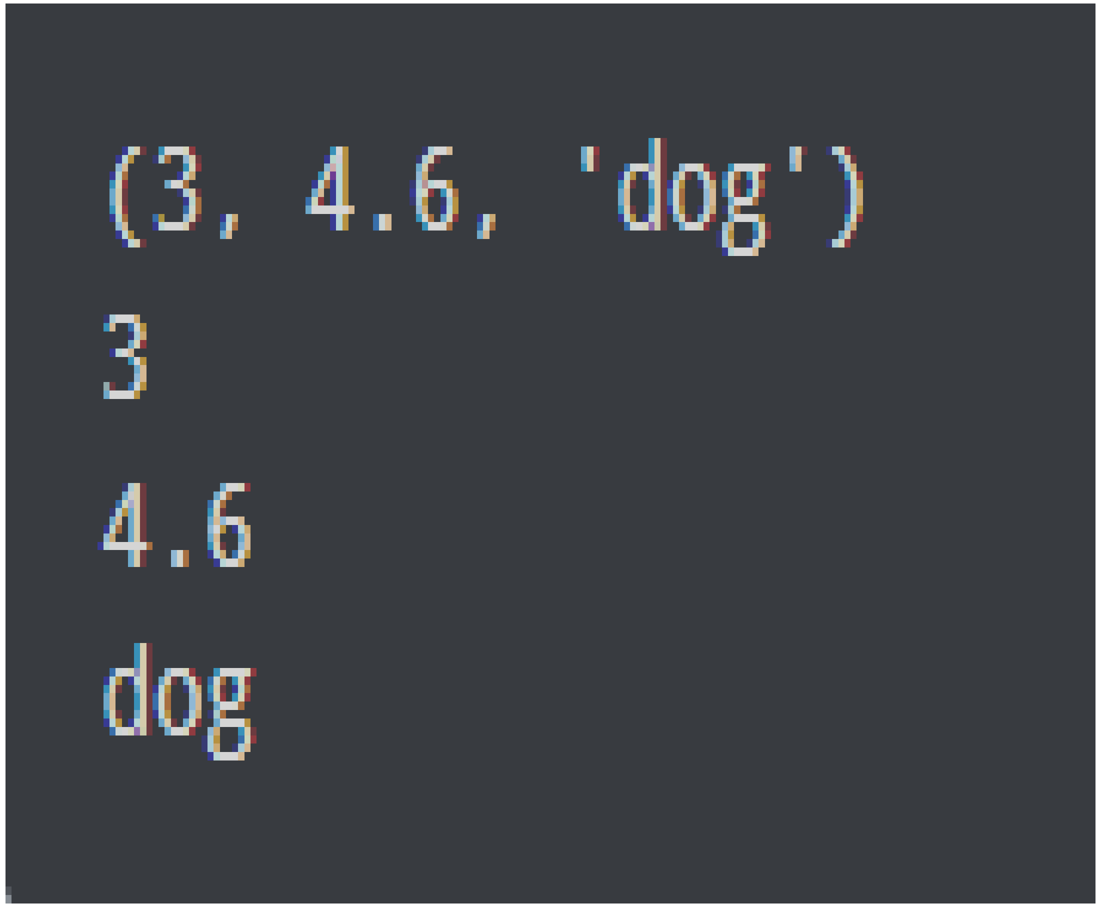


```
()
(1, 2, 3)
(1, 'Hello', 3.4)
('mouse', [8, 4, 6], (1, 2, 3))
```

A tuple can also be created without using parentheses. This is known as tuple packing.

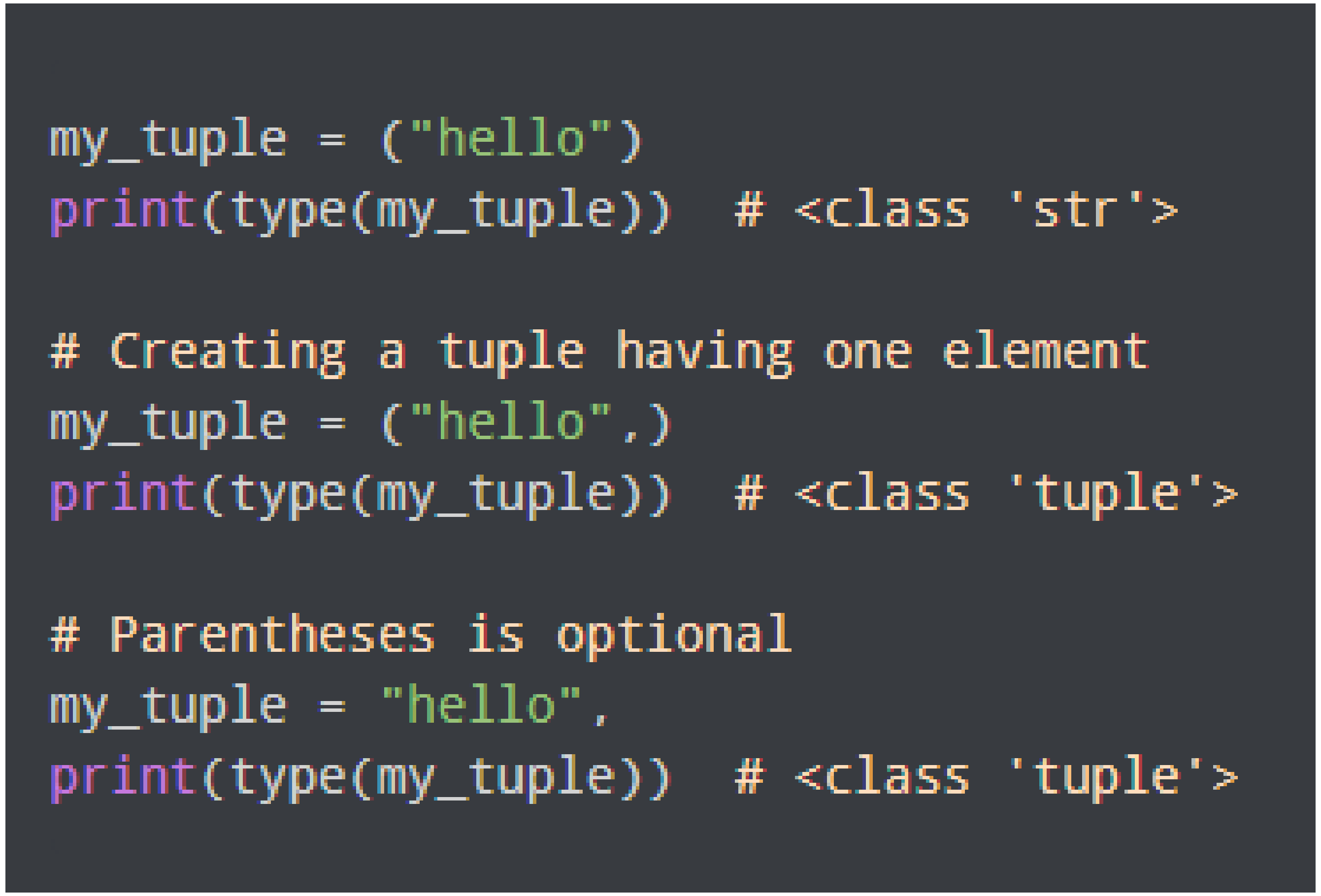


```
my_tuple = 3, 4.6, "dog"
print(my_tuple)

# tuple unpacking is also possible
a, b, c = my_tuple

print(a)      # 3
print(b)      # 4.6
print(c)      # dog
```

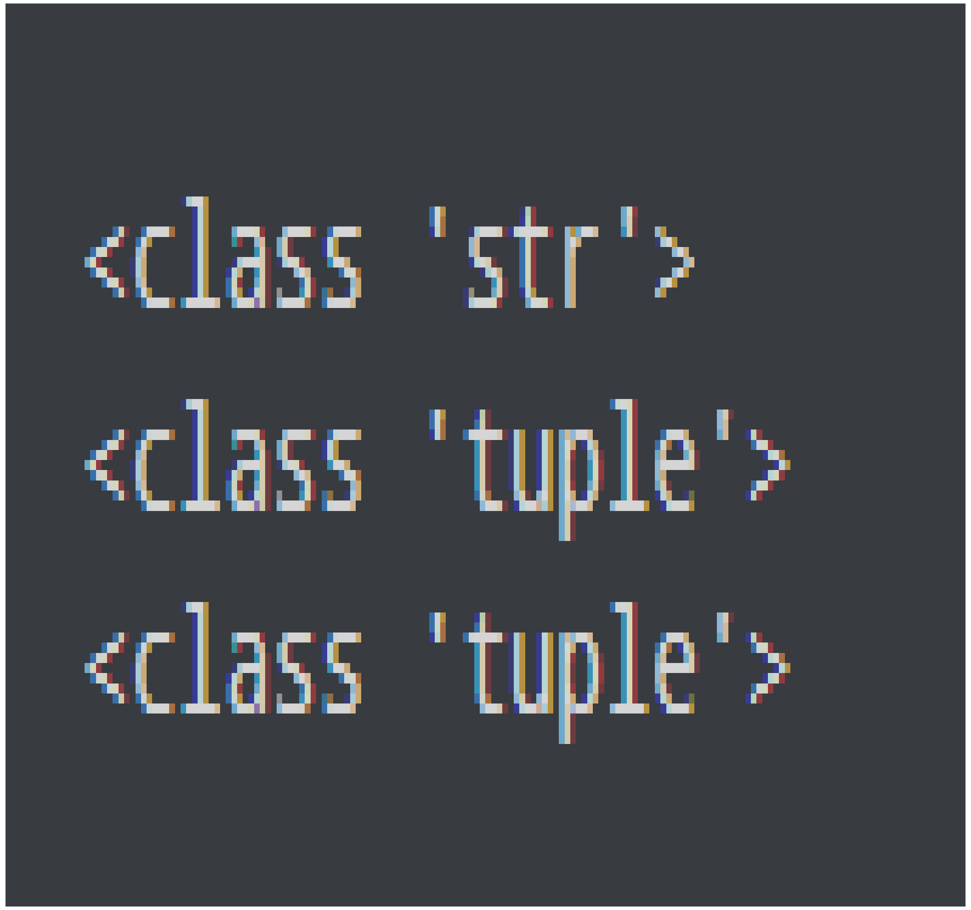


```
(3, 4.6, 'dog')
3
4.6
dog
```

Creating a tuple with one element

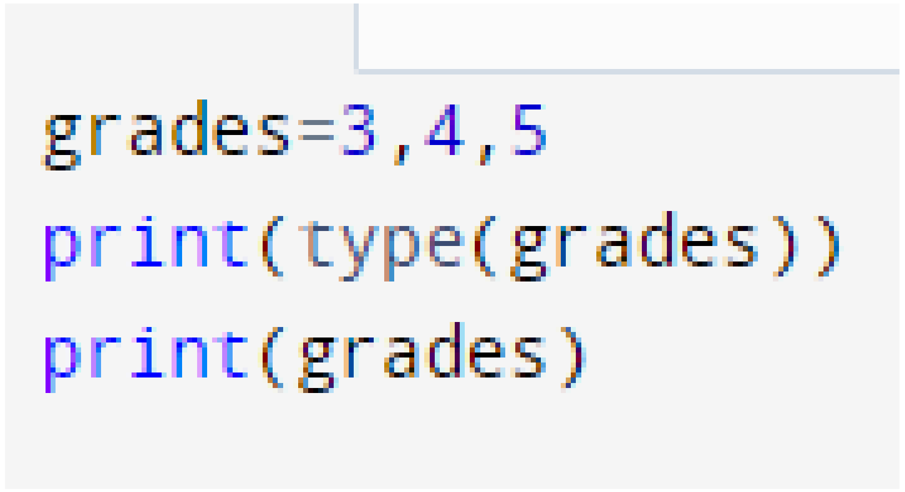


```python
my_tuple = ("hello")
print(type(my_tuple))  # <class 'str'>

# Creating a tuple having one element
my_tuple = ("hello",)
print(type(my_tuple))  # <class 'tuple'>

# Parentheses is optional
my_tuple = "hello",
print(type(my_tuple))  # <class 'tuple'>
```

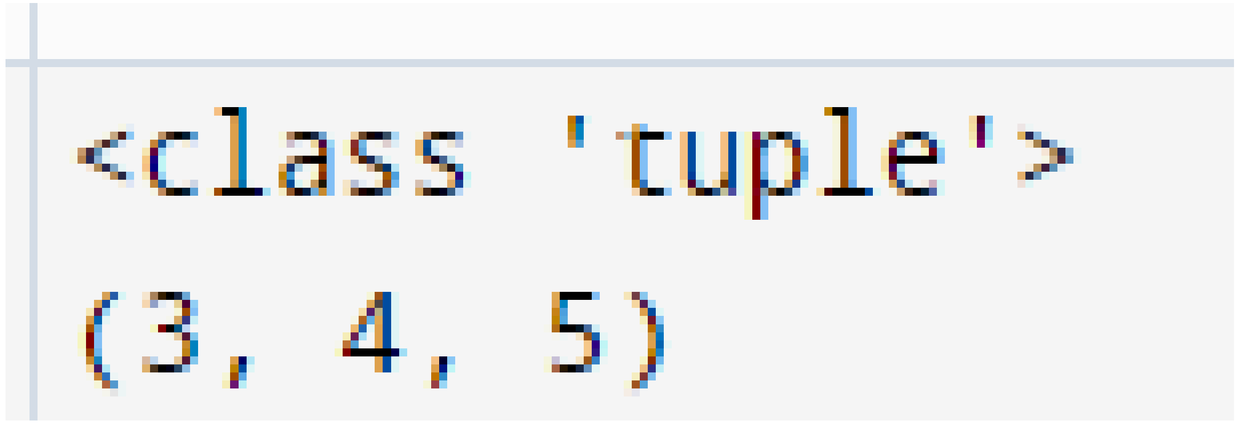


```
<class 'str'>
<class 'tuple'>
<class 'tuple'>
```

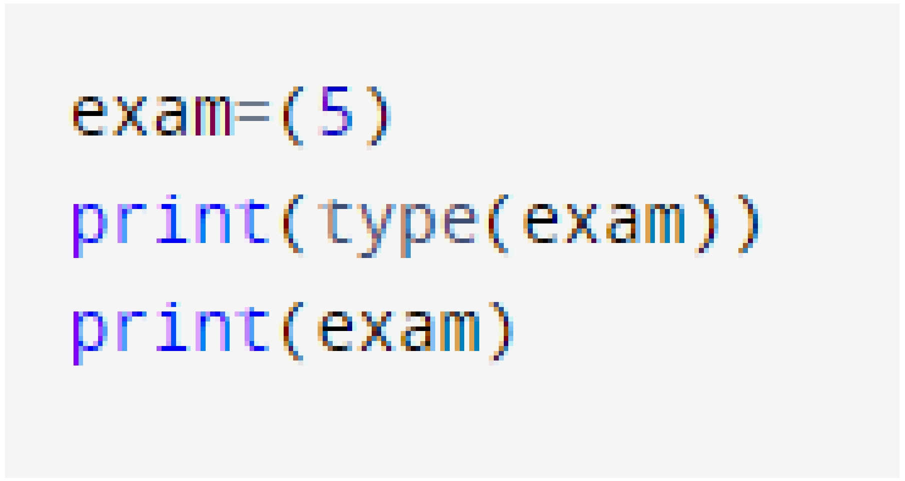
What will the program output?



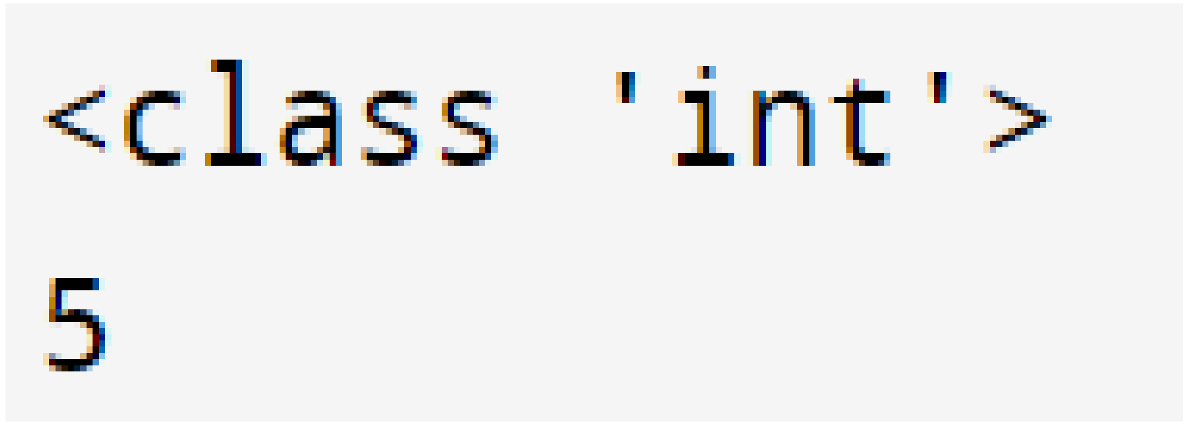
```python
grades=3,4,5
print(type(grades))
print(grades)
```



```
<class 'tuple'>
(3, 4, 5)
```



```python
exam=(5)
print(type(exam))
print(exam)
```



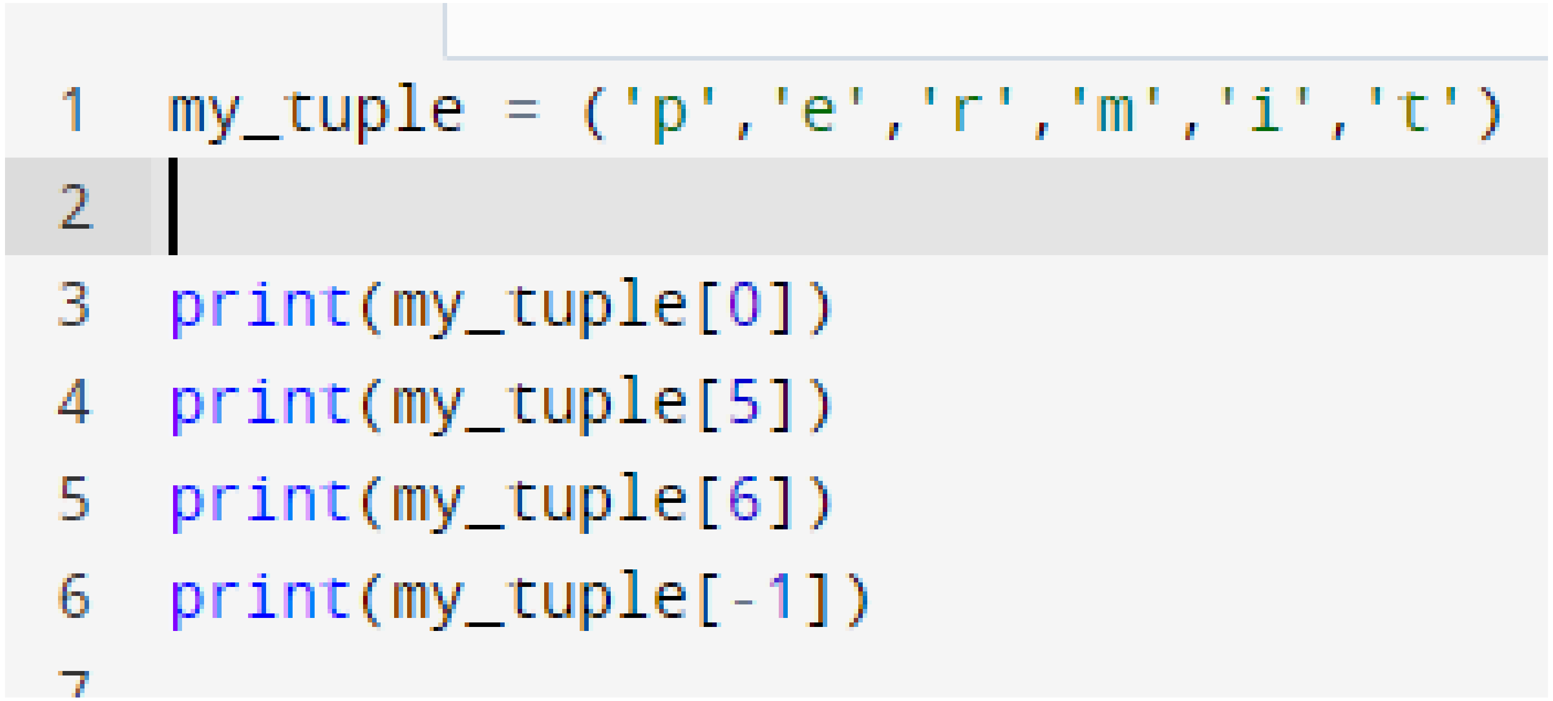
```
<class 'int'>
5
```

Access Tuple Elements

- **Indexing**
- We can use the index operator [] to access an item in a tuple, where the index starts from 0.

- The index must be an integer, so we cannot use float or other types. This will result in TypeError

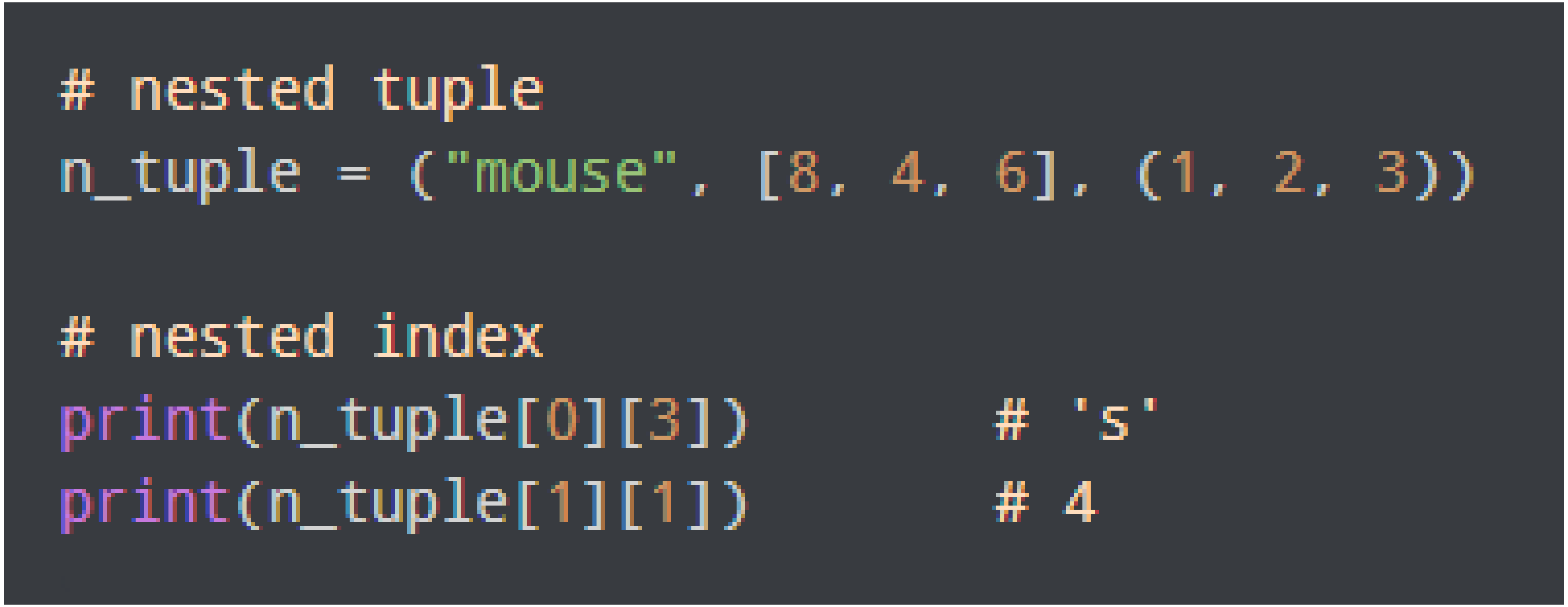
What will the program output?



```
my_tuple = ('p','e','r','m','i','t')

print(my_tuple[0])
print(my_tuple[5])
print(my_tuple[6])
print(my_tuple[-1])
```

- p
- t
- IndexError: tuple index out of rang
- t

Nested tuples are accessed using nested indexing



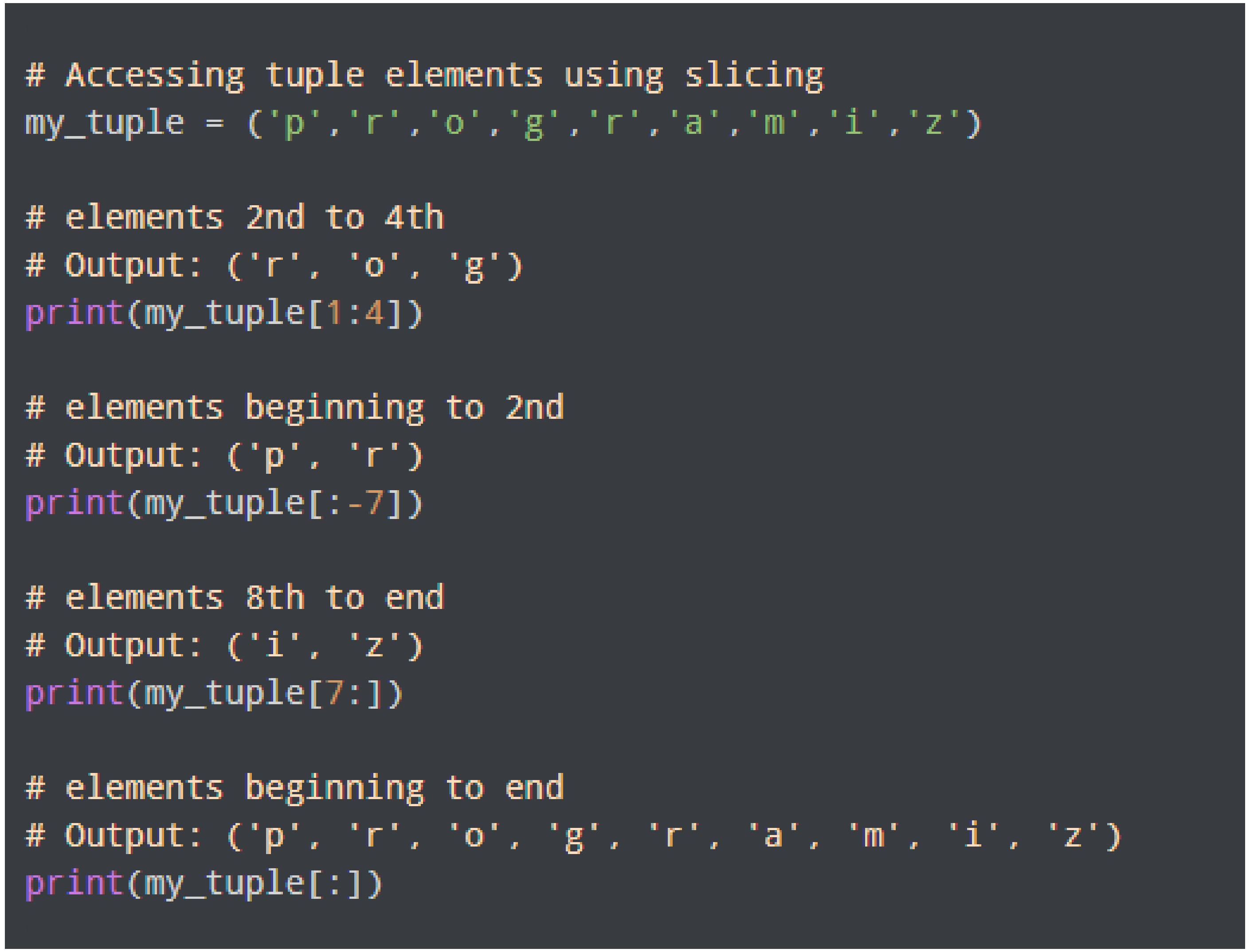
```
# nested tuple
n_tuple = ("mouse", [8, 4, 6], (1, 2, 3))

# nested index
print(n_tuple[0][3])       # 's'
print(n_tuple[1][1])       # 4
```

What will the program output?

```
student =('surname',[3,4,5])


print(student[0,3])
print(student[0][3])
print(student[1][3])
print(student[1][2])
```

- TypeError: tuple indices must be integers
- N
- IndexError: list index out of range
- 5

Slicing



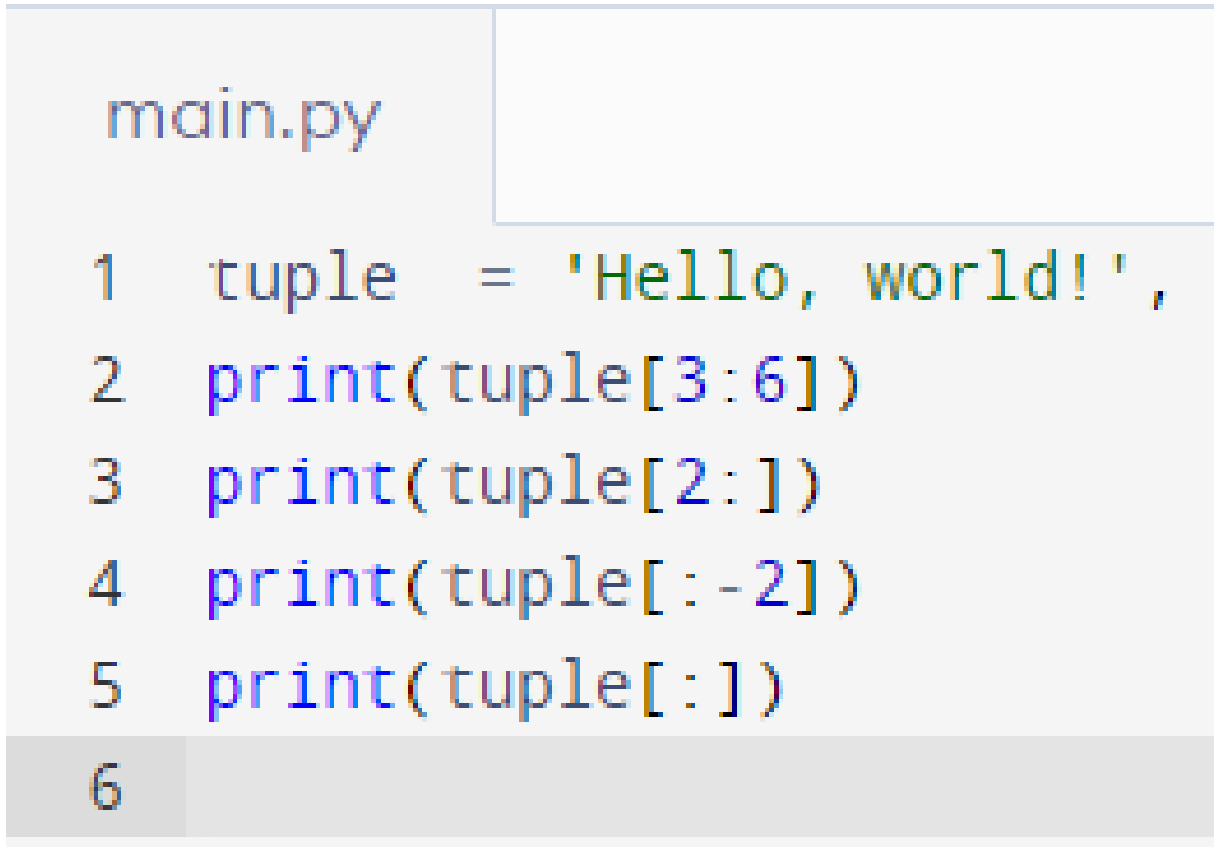
```python
# Accessing tuple elements using slicing
my_tuple = ('p','r','o','g','r','a','m','i','z')

# elements 2nd to 4th
# Output: ('r', 'o', 'g')
print(my_tuple[1:4])

# elements beginning to 2nd
# Output: ('p', 'r')
print(my_tuple[:-7])

# elements 8th to end
# Output: ('i', 'z')
print(my_tuple[7:])

# elements beginning to end
# Output: ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')
print(my_tuple[:])
```

What will the program output?

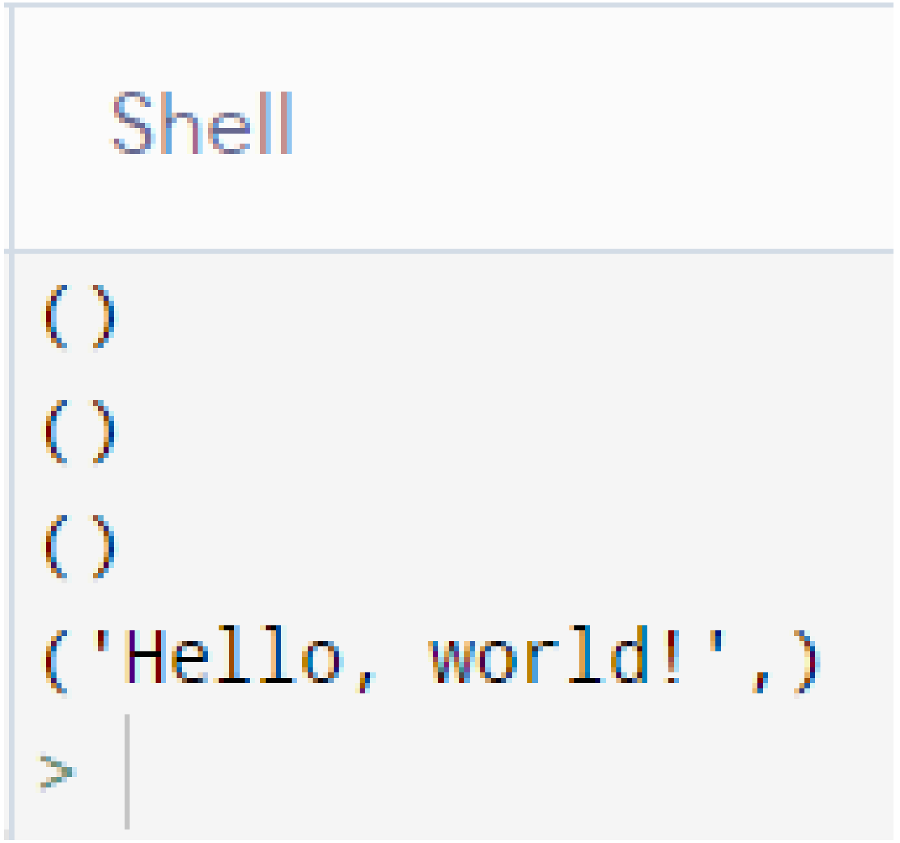
```
str = 'Hello, world!’
print(str[3:6])
print(str[2:])
print(str[:-2])
print(str[:])
```

```
lo,
llo, world!
Hello, worl
Hello, world!
```

What will the program output?

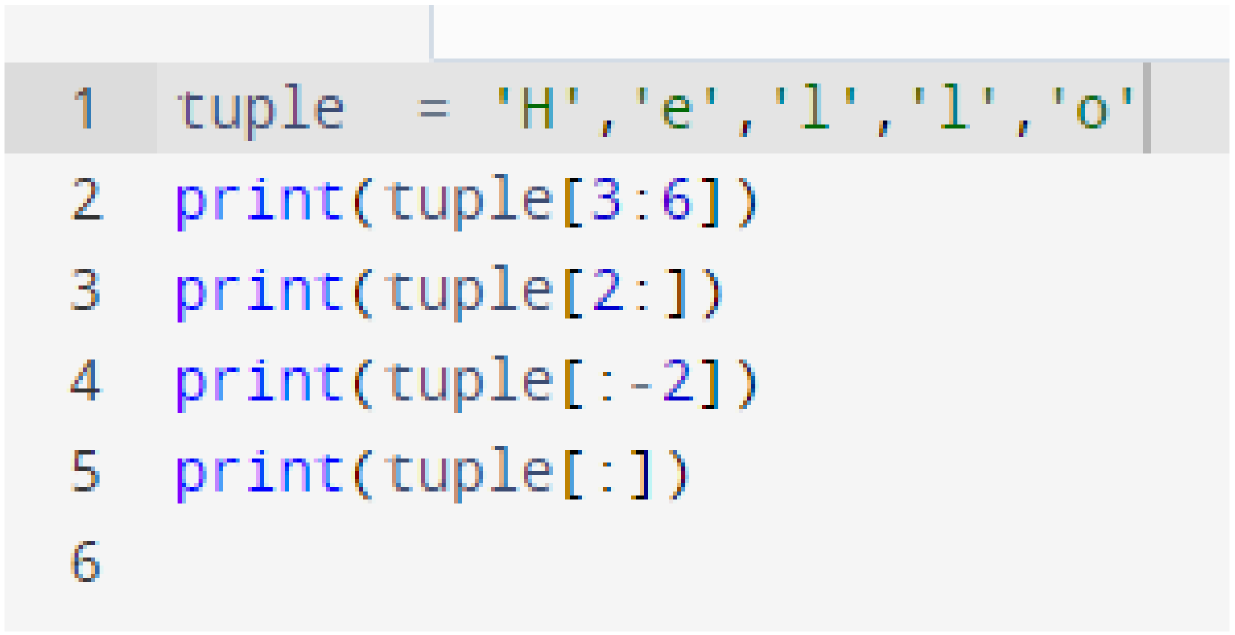

main.py

```python
tuple  = 'Hello, world!',
print(tuple[3:6])
print(tuple[2:])
print(tuple[:-2])
print(tuple[:])
```

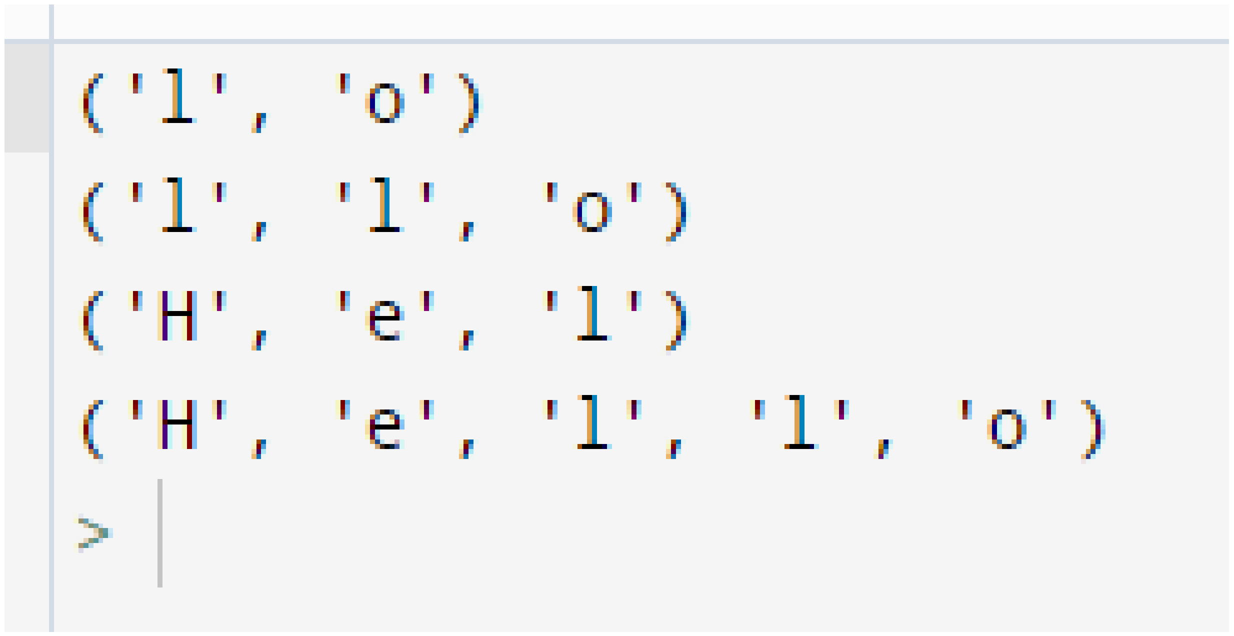

Shell

```
()
()
()
('Hello, world!',)
>
```

What will the program output?



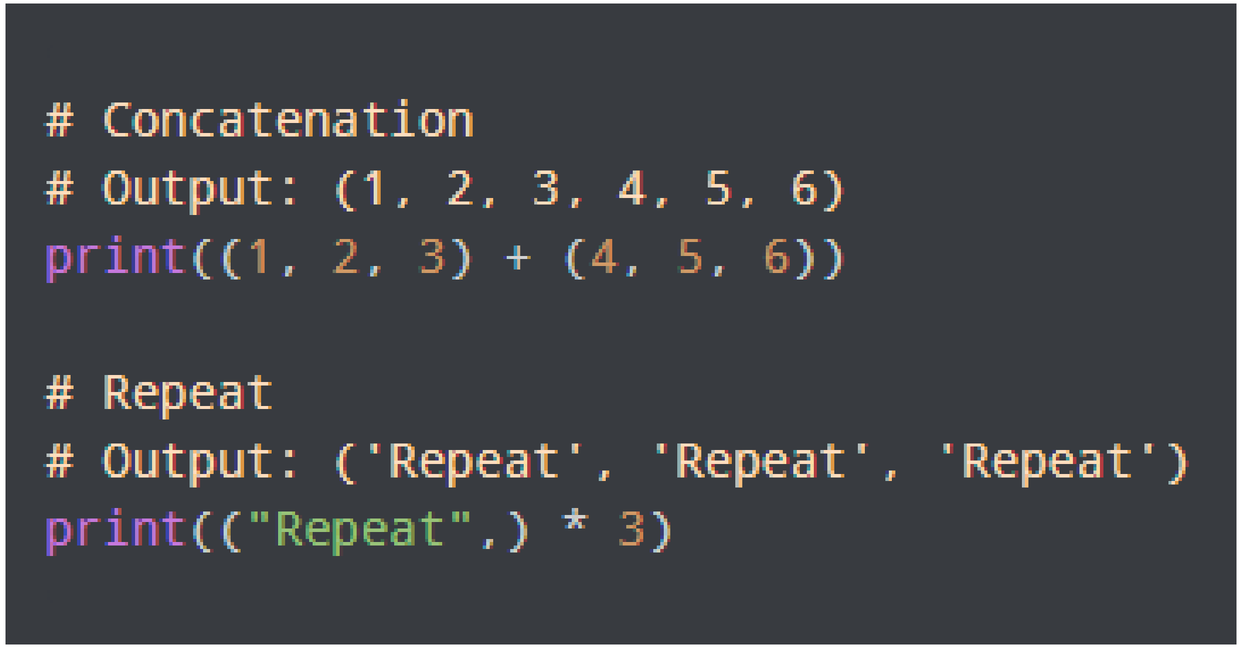
```python
tuple  = 'H','e','l','l','o'
print(tuple[3:6])
print(tuple[2:])
print(tuple[:-2])
print(tuple[:])
```



```
('l', 'o')
('l', 'l', 'o')
('H', 'e', 'l')
('H', 'e', 'l', 'l', 'o')
>
```

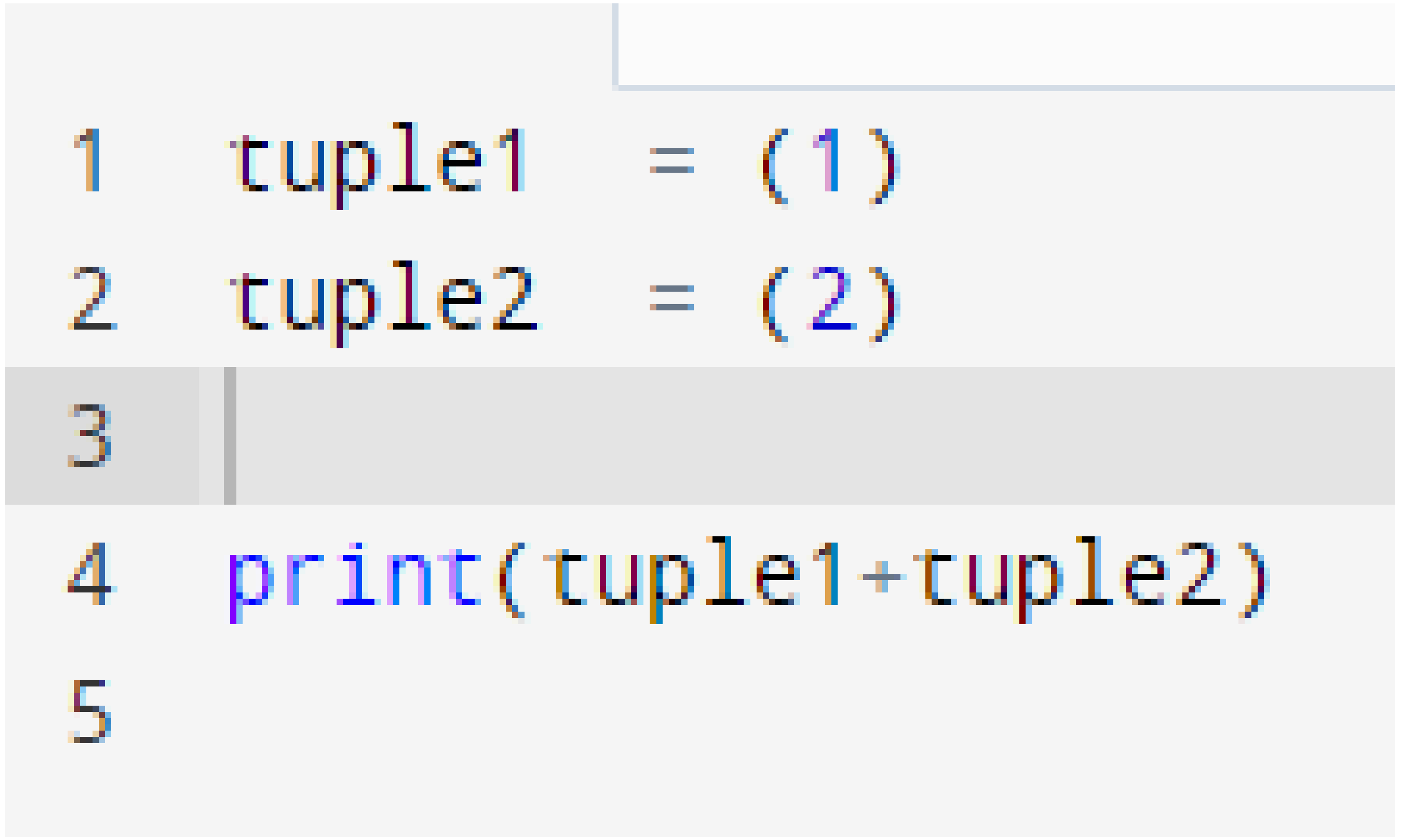
- We can use + operator to combine two tuples. This is called concatenation.

- We can also repeat the elements in a tuple for a given number of times using the * operator.



```
# Concatenation
# Output: (1, 2, 3, 4, 5, 6)
print((1, 2, 3) + (4, 5, 6))

# Repeat
# Output: ('Repeat', 'Repeat', 'Repeat')
print(("Repeat",) * 3)
```

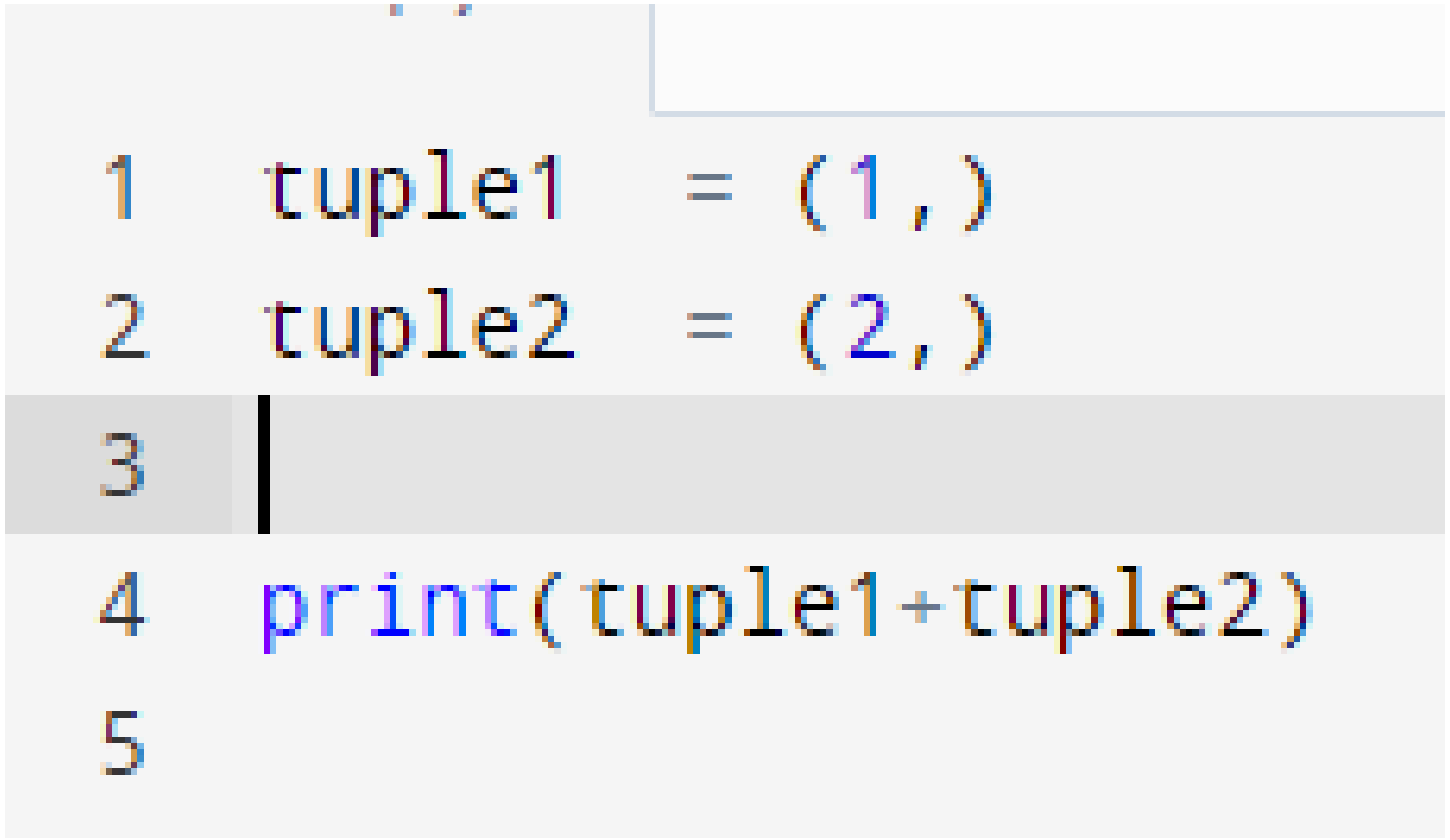
What will the program output?



```
tuple1  = (1)
tuple2  = (2)

print(tuple1+tuple2)

```

3

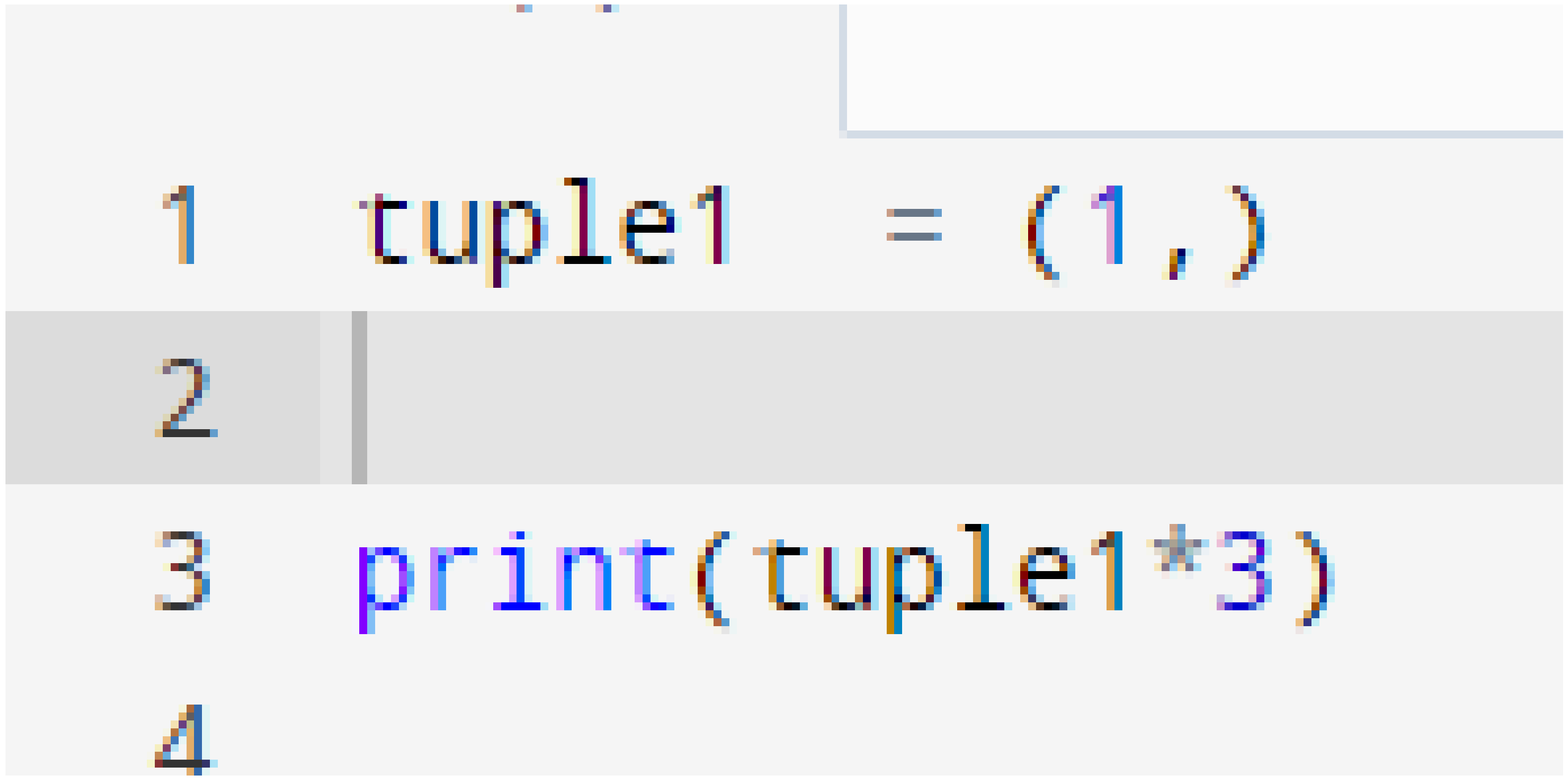
What will the program output?



```
tuple1  = (1,)
tuple2  = (2,)

print(tuple1+tuple2)
```

(1, 2)

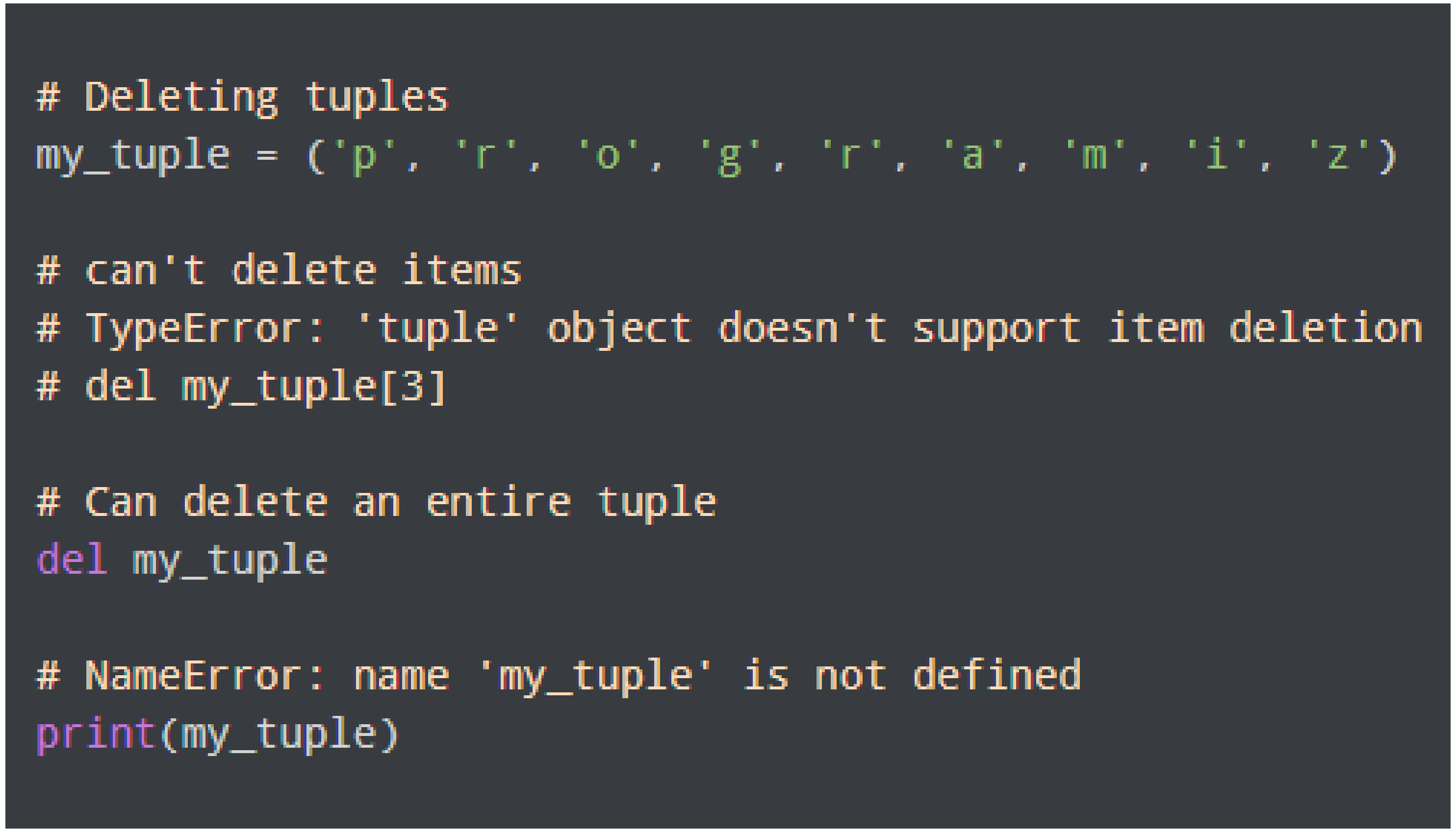
What will the program output?


```
tuple1  = (1,)

print(tuple1*3)
```

(1, 1, 1)

Deleting a Tuple

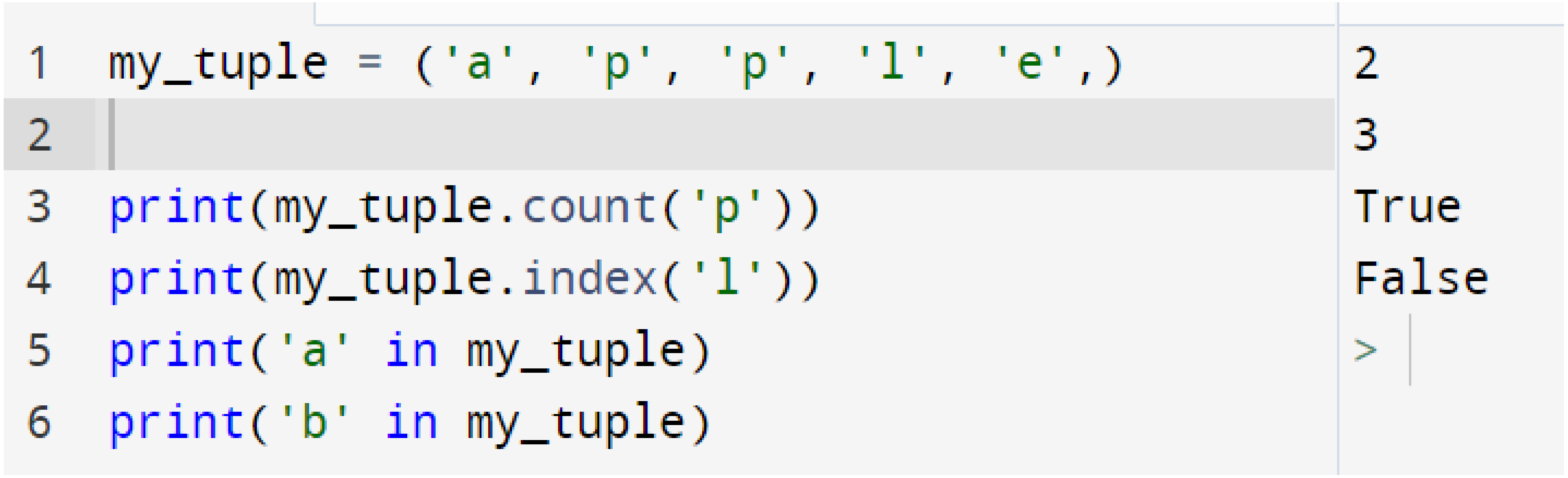
- As discussed above, we cannot change the elements in a tuple. It means that we cannot delete or remove items from a tuple.
- Deleting a tuple entirely, however, is possible using the keyword del.



```
# Deleting tuples
my_tuple = ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

# can't delete items
# TypeError: 'tuple' object doesn't support item deletion
# del my_tuple[3]

# Can delete an entire tuple
del my_tuple

# NameError: name 'my_tuple' is not defined
print(my_tuple)
```

Tuple Methods

- Methods that add items or remove items are not available with tuple.
- Only the following two methods are available.
- We can test if an item exists in a tuple or not, using the keyword in.

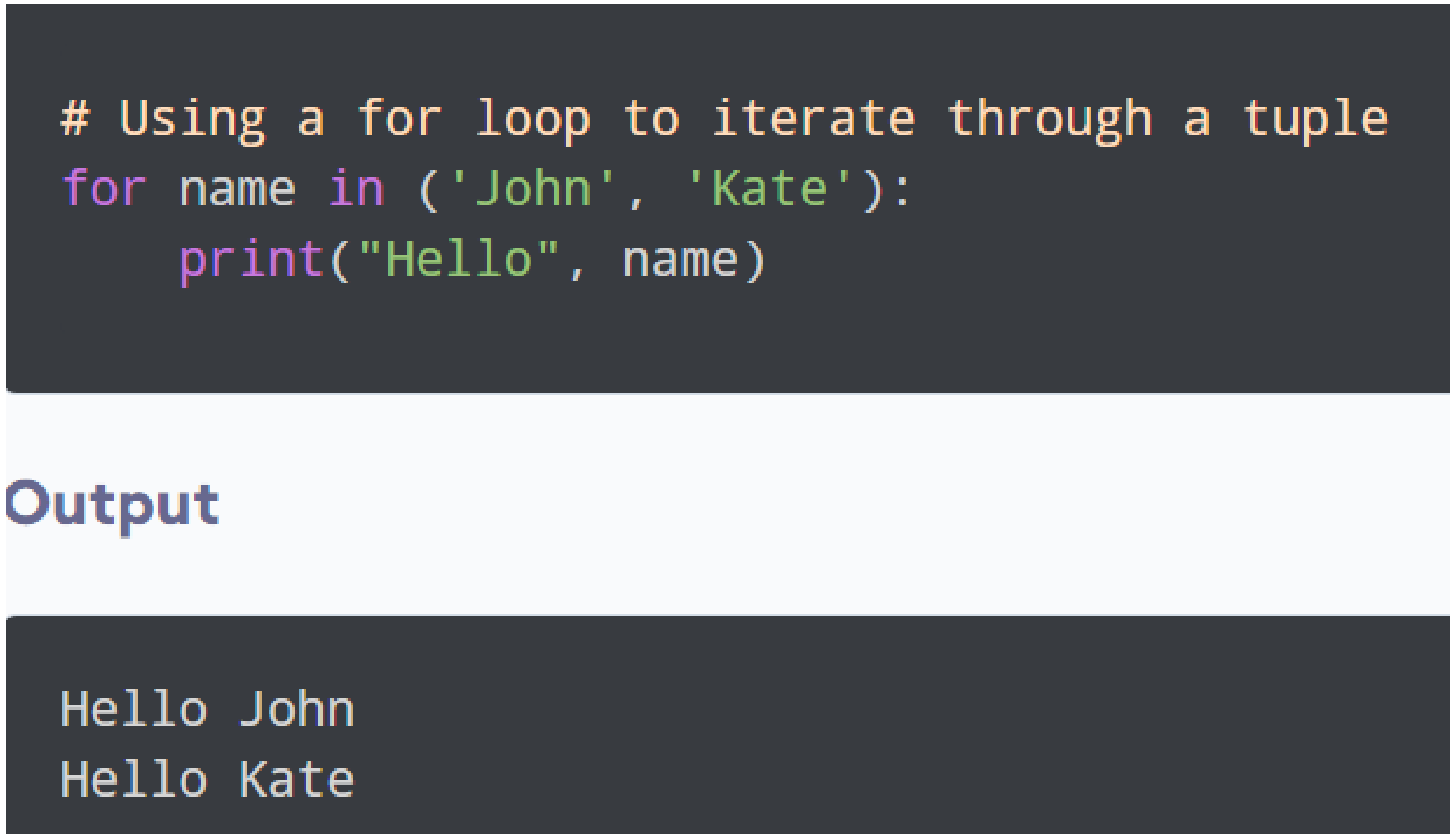


```python
my_tuple = ('a', 'p', 'p', 'l', 'e',)

print(my_tuple.count('p'))
print(my_tuple.index('l'))
print('a' in my_tuple)
print('b' in my_tuple)
```

```
2
3
True
False
>
```

Iterating Through a Tuple

We can use a for loop to iterate through each item in a tuple.

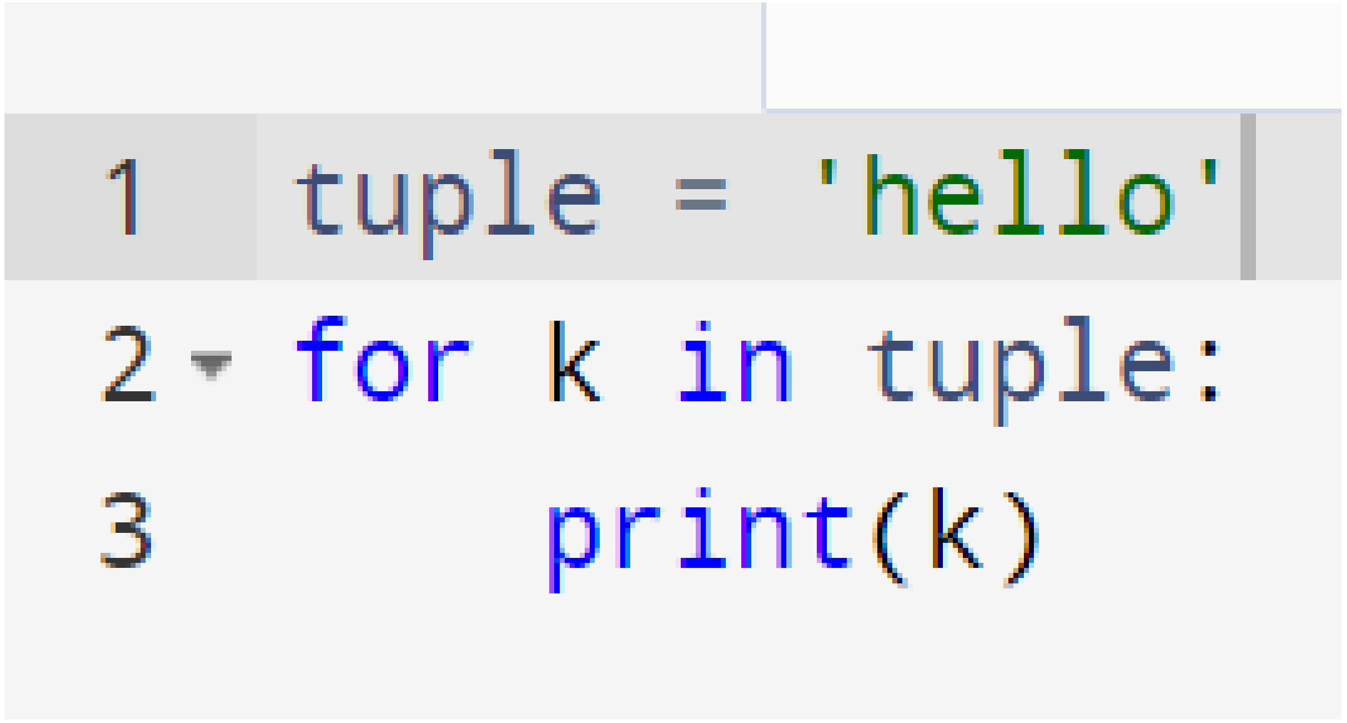


```python
# Using a for loop to iterate through a tuple
for name in ('John', 'Kate'):
    print("Hello", name)
```

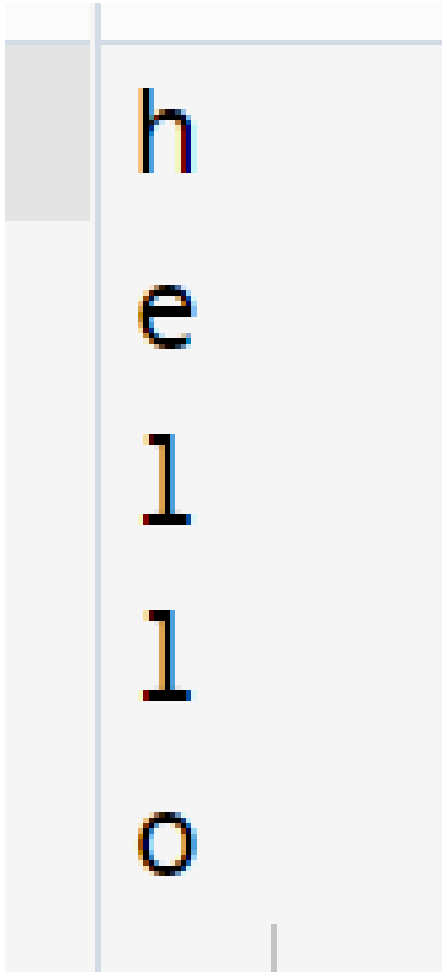
Output

```
Hello John
Hello Kate
```

What will the program output?

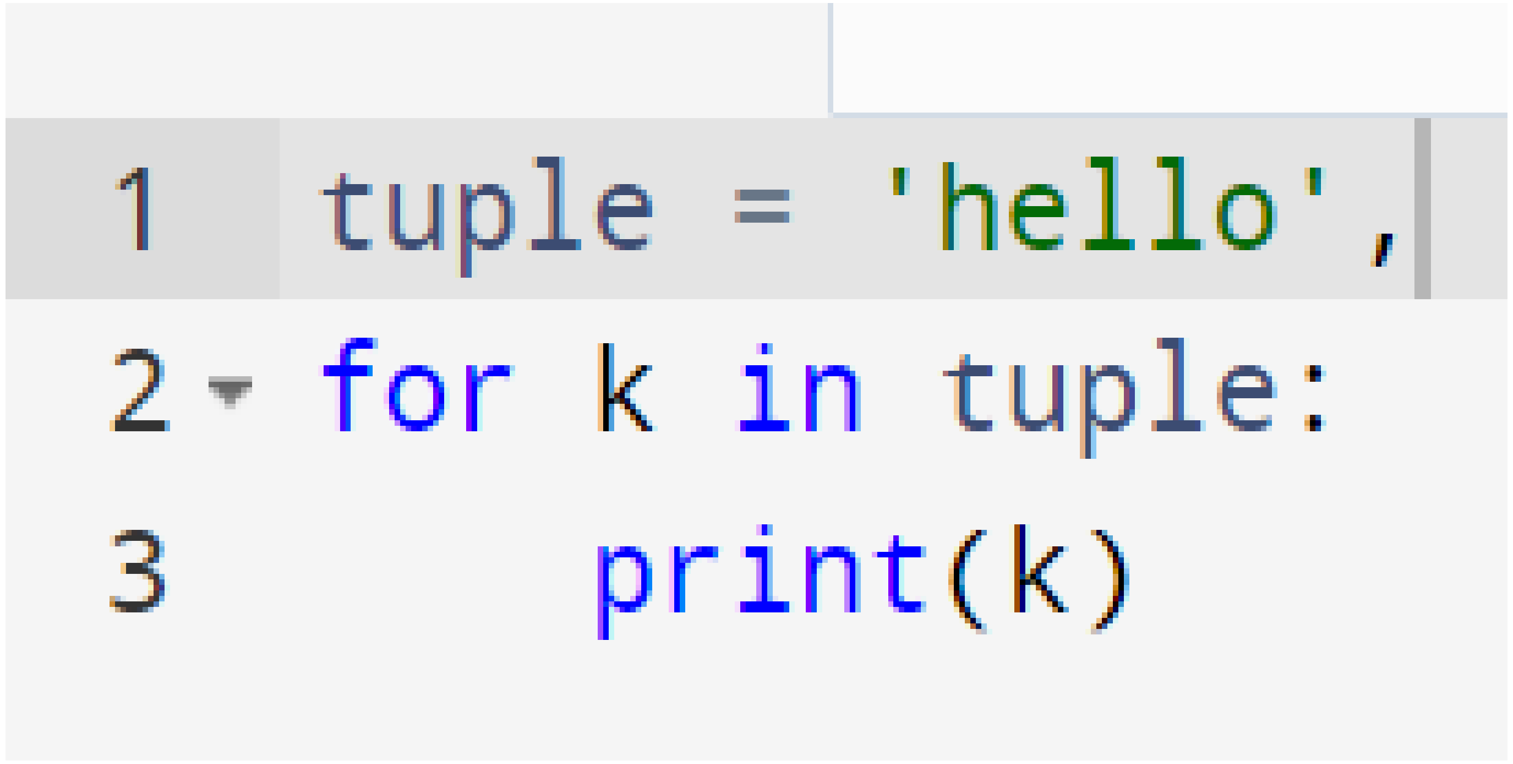


```
tuple = 'hello'
for k in tuple:
        print(k)
```

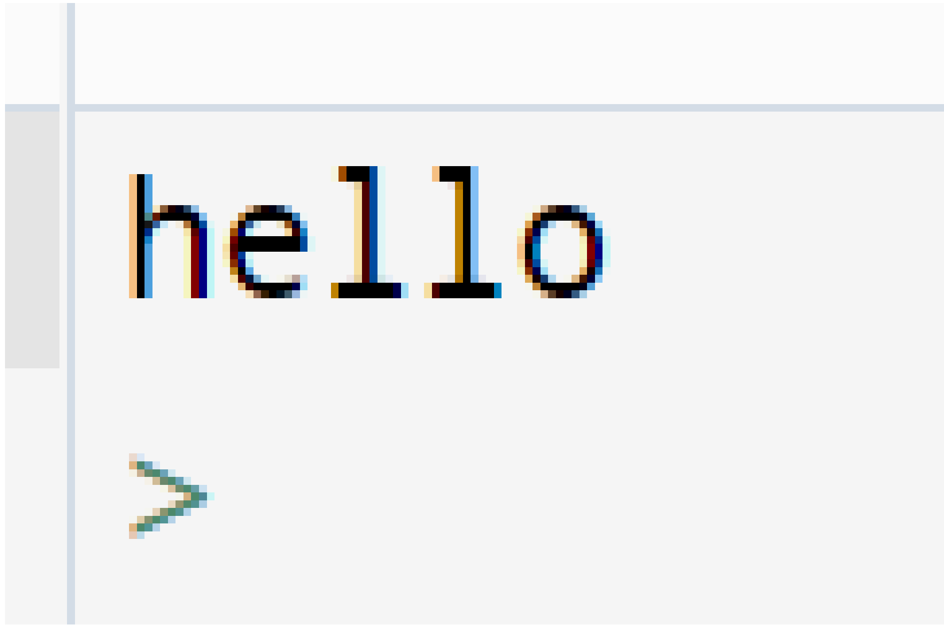


```
h
e
l
l
o
```

What will the program output?

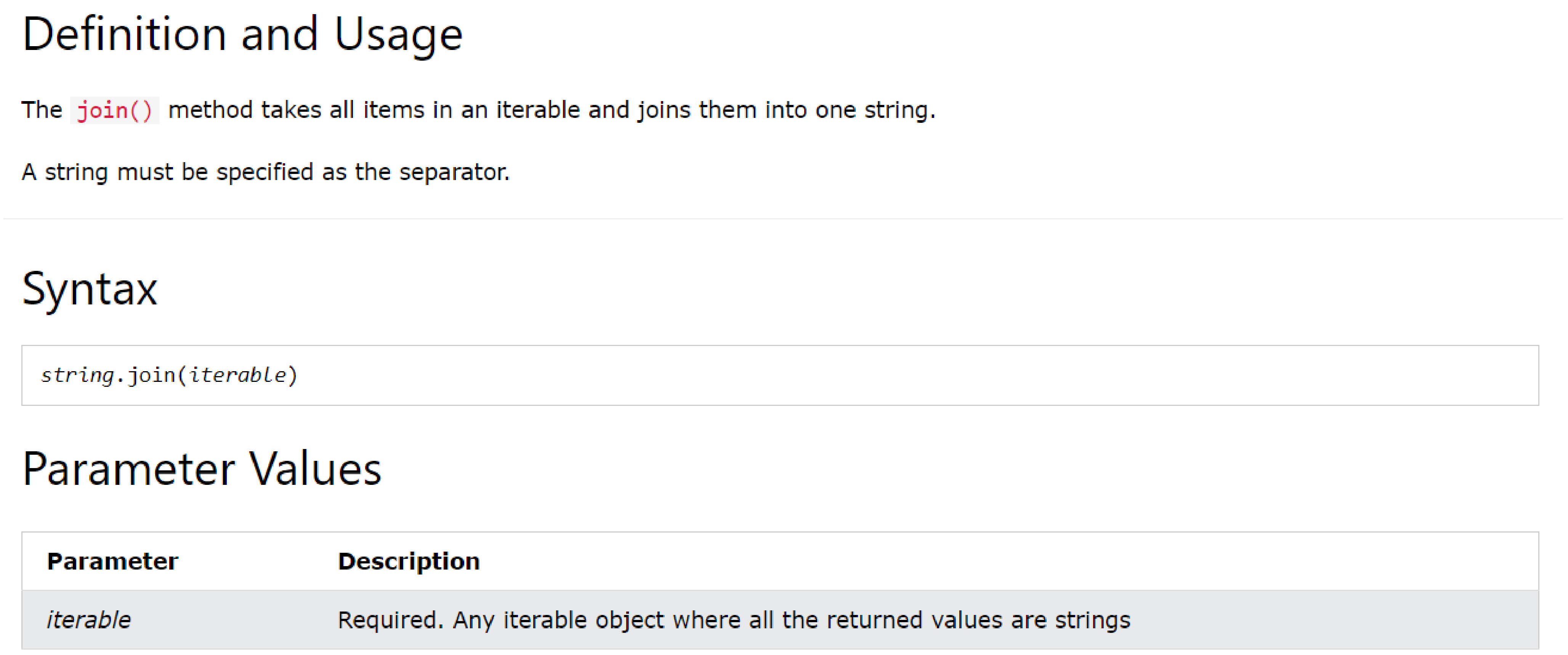


```python
tuple = 'hello',
for k in tuple:
        print(k)
```



```
hello

>
```

String methods: join() and split()



Definition and Usage

The `join()` method takes all items in an iterable and joins them into one string.

A string must be specified as the separator.

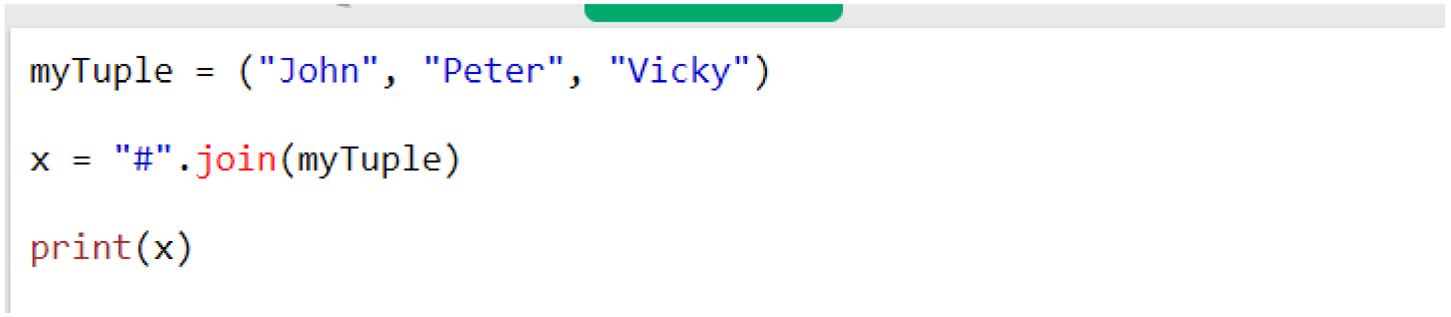
Syntax

```
string.join(iterable)
```

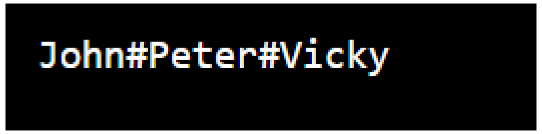
Parameter Values

Parameter	Description
iterable	Required. Any iterable object where all the returned values are strings

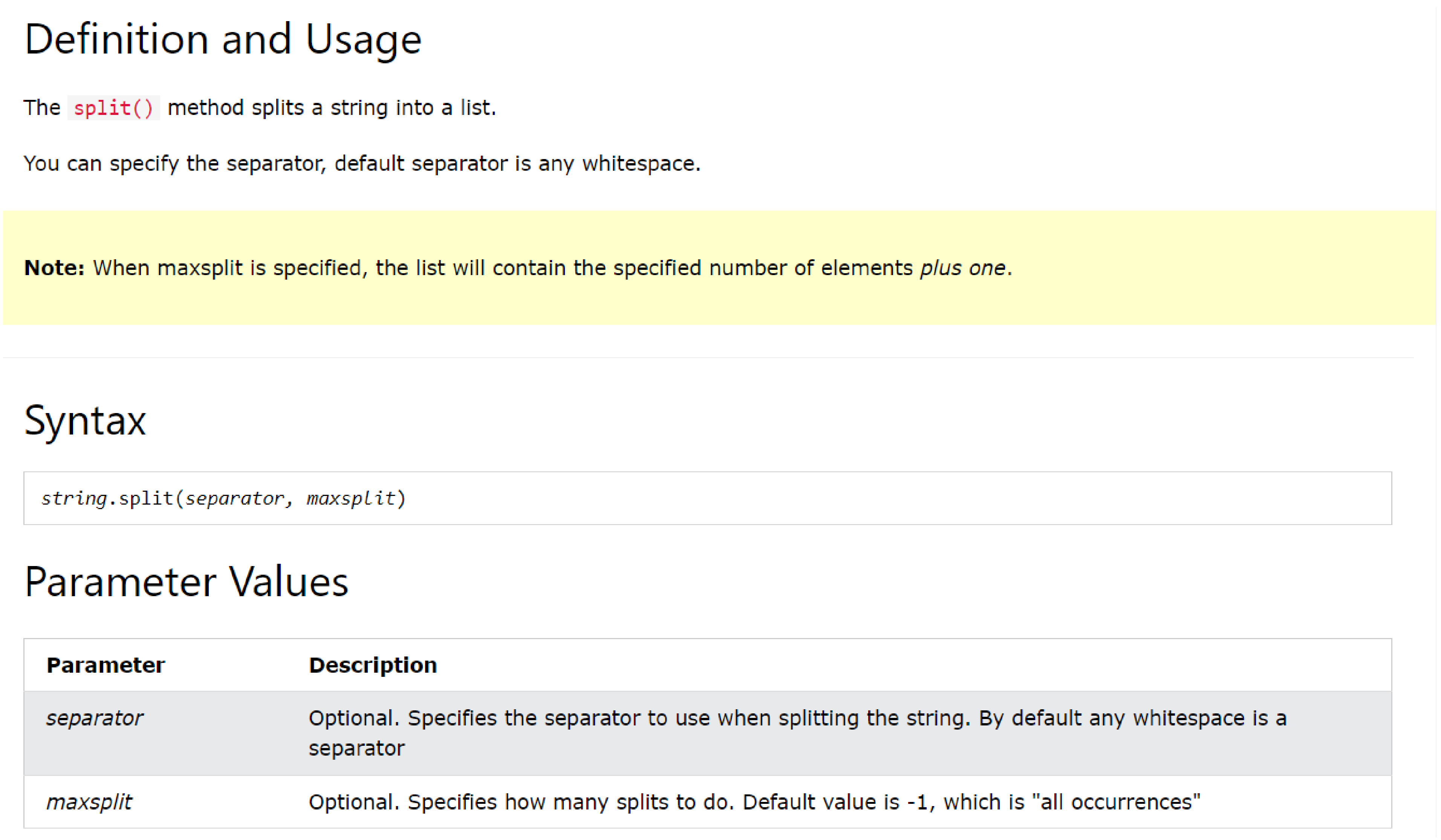
What is the output?



```
myTuple = ("John", "Peter", "Vicky")

x = "#".join(myTuple)

print(x)
```



```
John#Peter#Vicky
```



Definition and Usage

The `split()` method splits a string into a list.

You can specify the separator, default separator is any whitespace.

Note: When maxsplit is specified, the list will contain the specified number of elements *plus one*.

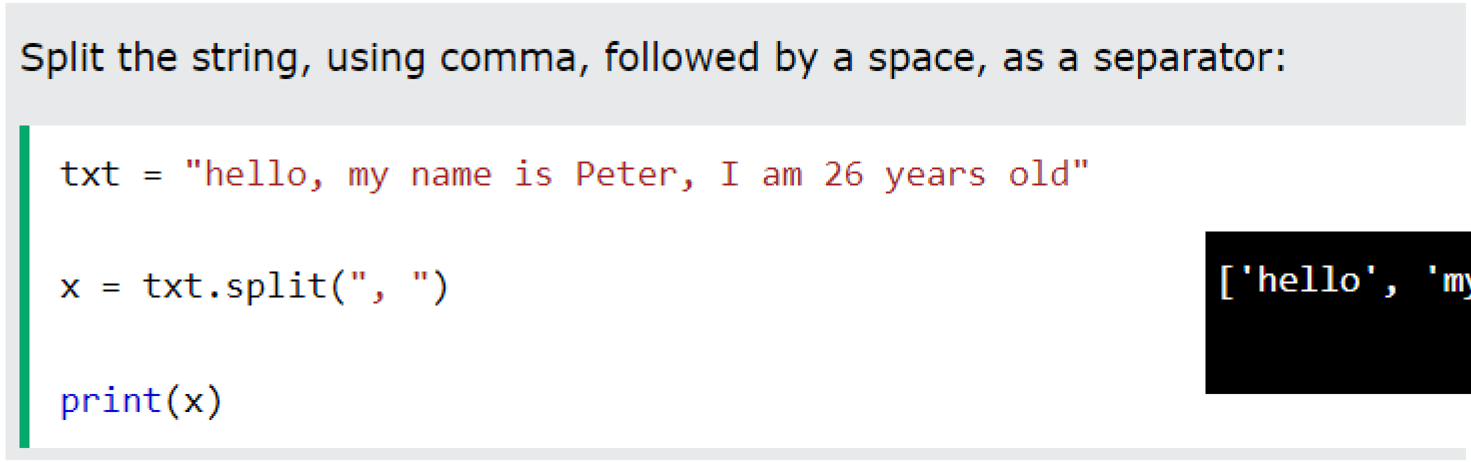
Syntax

```
string.split(separator, maxsplit)
```

Parameter Values

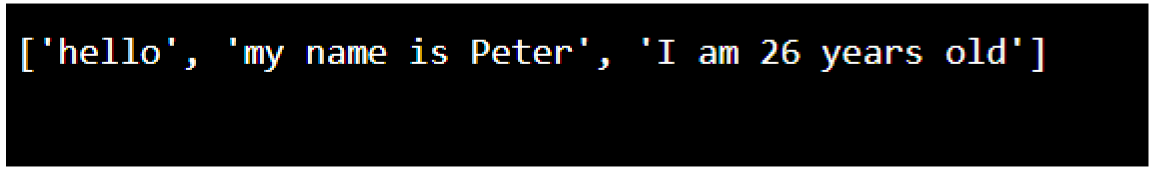
Parameter	Description
separator	Optional. Specifies the separator to use when splitting the string. By default any whitespace is a separator
maxsplit	Optional. Specifies how many splits to do. Default value is -1, which is "all occurrences"

What is the output?

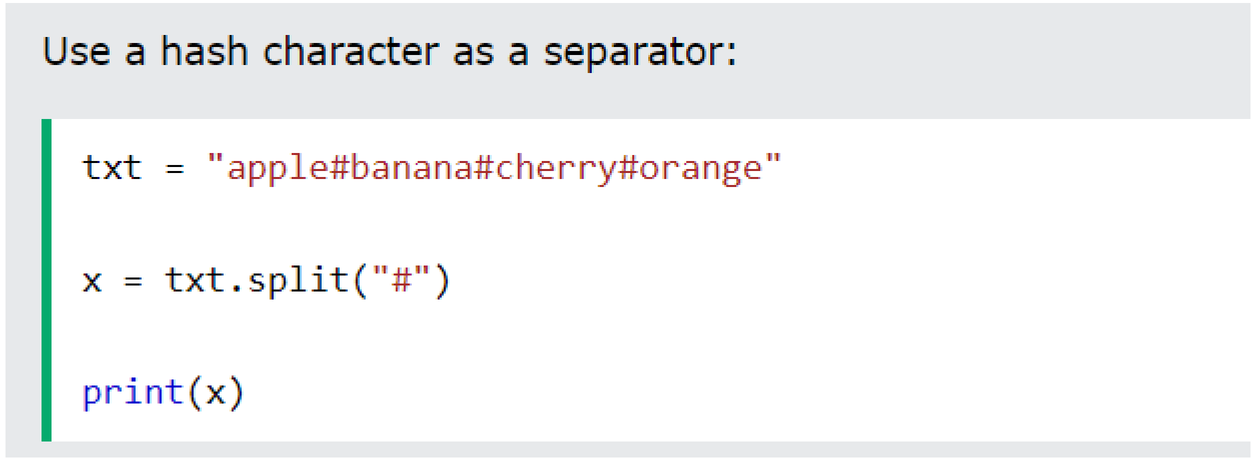


Split the string, using comma, followed by a space, as a separator:

```python
txt = "hello, my name is Peter, I am 26 years old"

x = txt.split(", ")

print(x)
```

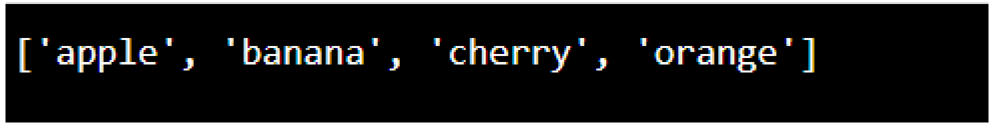


```
['hello', 'my name is Peter', 'I am 26 years old']
```



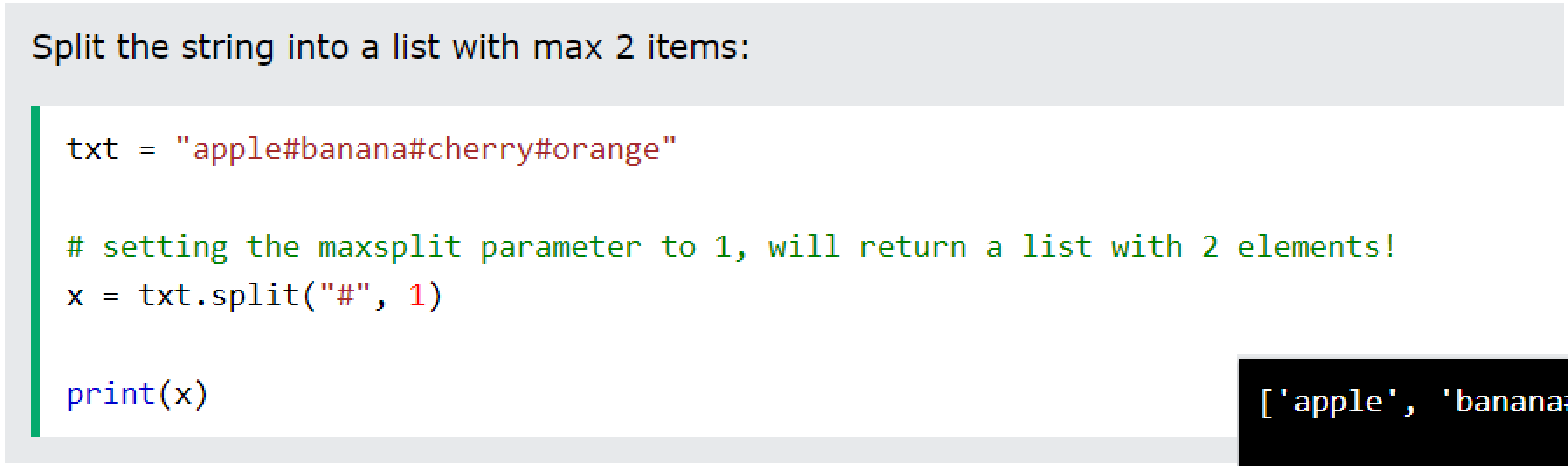
Use a hash character as a separator:

```python
txt = "apple#banana#cherry#orange"

x = txt.split("#")

print(x)
```

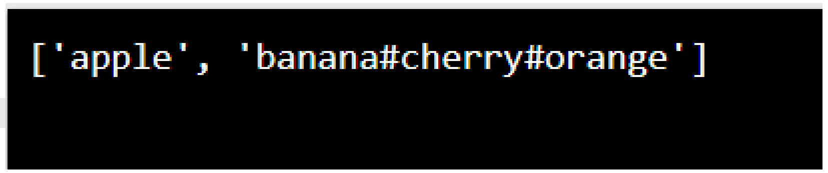


```
['apple', 'banana', 'cherry', 'orange']
```

What is the output?


Split the string into a list with max 2 items:

```python
txt = "apple#banana#cherry#orange"

# setting the maxsplit parameter to 1, will return a list with 2 elements!
x = txt.split("#", 1)

print(x)
```


```
['apple', 'banana#cherry#orange']
```

Task 1

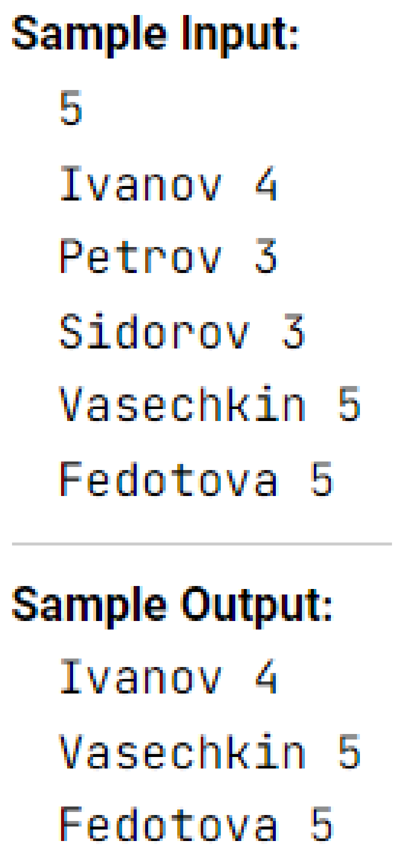
- Write a program in which you declare a tuple of the days of the week with values Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday
  - Using slices print work days
  - print the name of the day by the entered serial number
- Write a program that takes two values for two variables, then exchanges their values and prints them to the screen.

Task -2

- Write a program for analyzing temperature data. Use tuples to store temperature information every day. Implement functions to calculate the average temperature, search for the day with the highest and lowest temperature.

Task 3
Problem "Excellent and good students - 1"

- Write a program that identifies excellent and good students in computer science.
- **Input data format:**
- On the first line, enter an integer **n** - the number of students.
- On the following lines, enter the **n** surnames of the students and their grades.
- **Output data format:**
- Print line by line all excellent students and good students in the same sequence.



Sample Input:

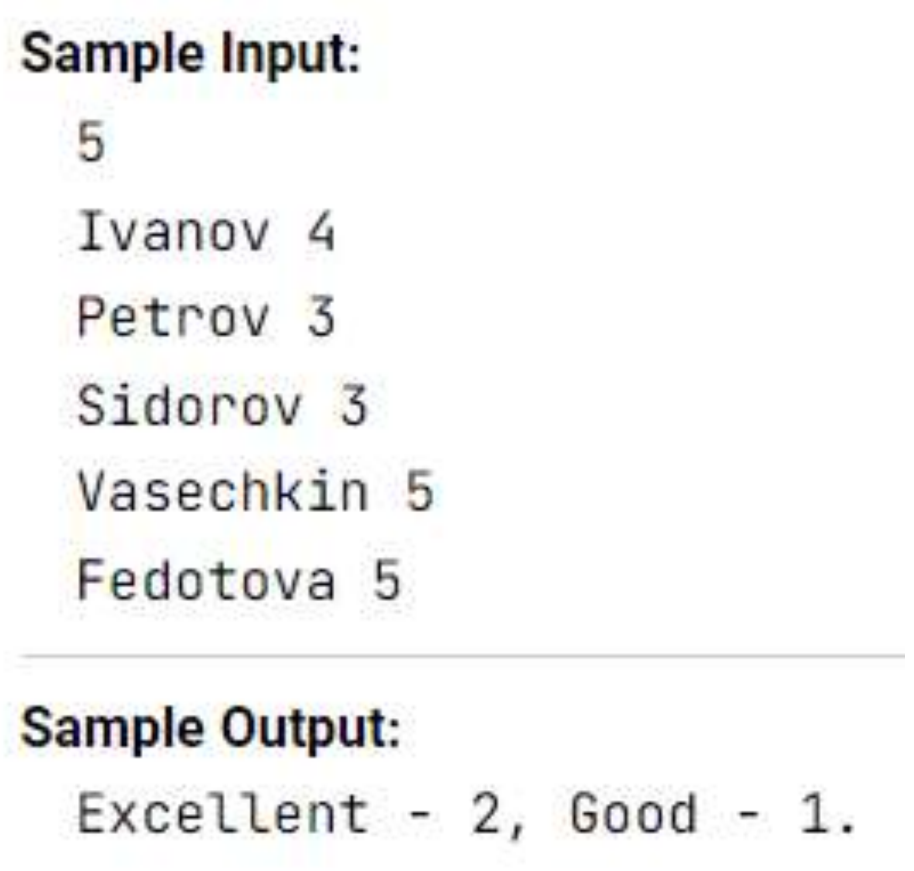
```
5
Ivanov 4
Petrov 3
Sidorov 3
Vasechkin 5
Fedotova 5
```

Sample Output:

```
Ivanov 4
Vasechkin 5
Fedotova 5
```

Problem "Excellent and good students - 2"

- Write a program that quantifies the number of excellent and good students in computer science.
- **Input data format:**
- On the first line, enter an integer **n** - the number of students.
- On the next lines, enter the **n** surnames of the students and their grades.
- **Output data format:**
- Output the number of excellent students anc good students in the format: "Excellent - (number), Good - (number)."



Sample Input:

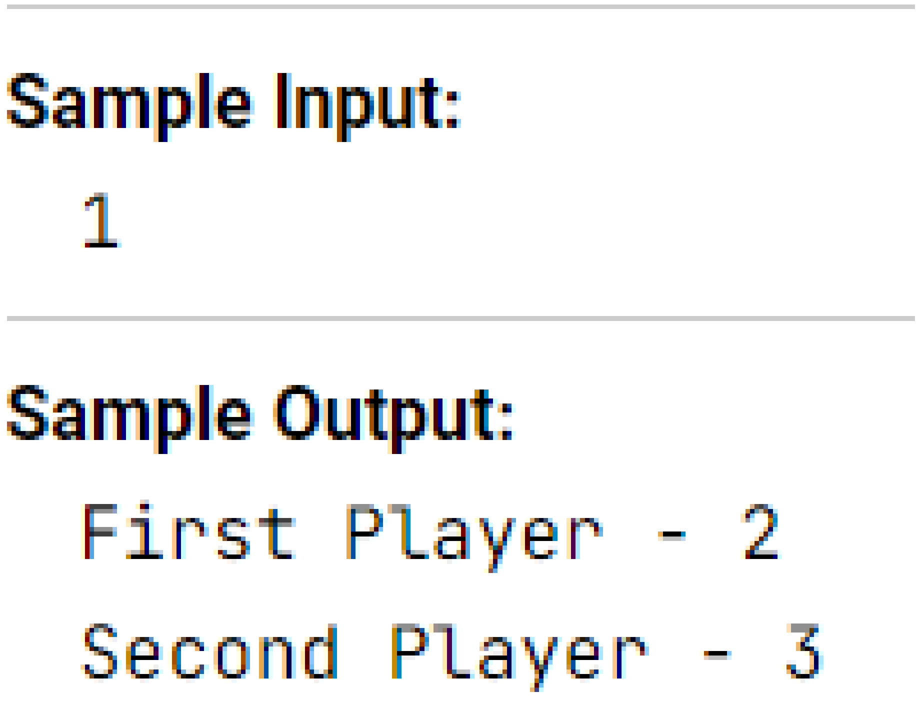
```
5
Ivanov 4
Petrov 3
Sidorov 3
Vasechkin 5
Fedotova 5
```

Sample Output:

```
Excellent - 2, Good - 1.
```

Task 4"Bones - 1"

- In a board game competition, two players take turns throwing six-sided dice pairwise.
- The entire chronology of the game is recorded as a list of tuples:
- [(2, 4), (5, 1), (6, 2), (4, 3), (5, 5), (6, 3), (2, 1), (4, 6), (6, 6), (3, 2), (4, 5), (3, 4), (6, 1), (1, 5), (5, 3), (1, 4)]
- Write a program that will determine for each player how many times **n** points are drawn.
- Write a program that will determine how many points each of the two players has scored by the end of the game.
- **Input data format:**
- The first line contains an integer **n** - the number of points on the dice.
- **Output data format:**
- At the output, indicate how many times this number of points fell for the first player, then for the second.



Sample Input:

```
1
```

Sample Output:

```
First Player - 2
Second Player - 3
```