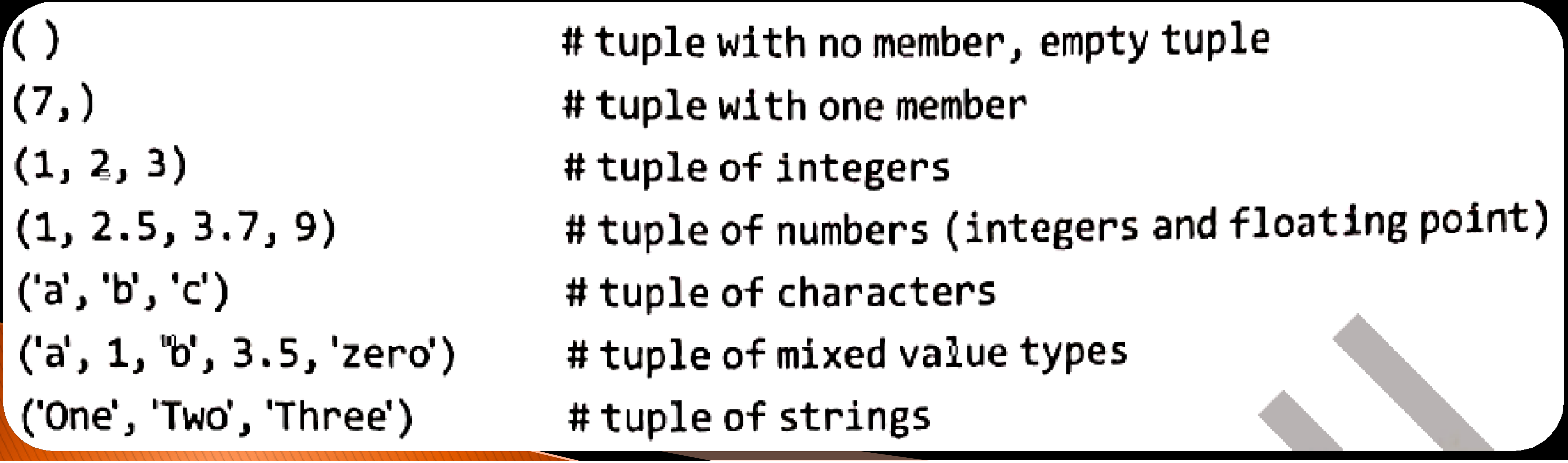
Tuples in Python

Tuples in Python

- The Tuples are depicted through parentheses
- i.e., round brackets
- Tuples are immutable sequences



```
( )                        # tuple with no member, empty tuple
(7,)                       # tuple with one member
(1, 2, 3)                  # tuple of integers
(1, 2.5, 3.7, 9)           # tuple of numbers (integers and floating point)
('a', 'b', 'c')            # tuple of characters
('a', 1, 'b', 3.5, 'zero') # tuple of mixed value types
('One', 'Two', 'Three')    # tuple of strings
```

Creating Tuples

- To create a tuple, put a number of expressions, separated by commas in parentheses.
- That is, to create a tuple you can write in the form given below:

```
T = ( )
T = (value, ...)
```

- This construct is known as a tuple display construct.

047 CREATING TUPLES

```
T = (1, 2, 3, 4, 5)

print(T)
```

Creating Tuples

- **Creating Empty Tuple**
- The empty tuple is ().
- You can also create an empty tuple as :

```
T = tuple ( )
```

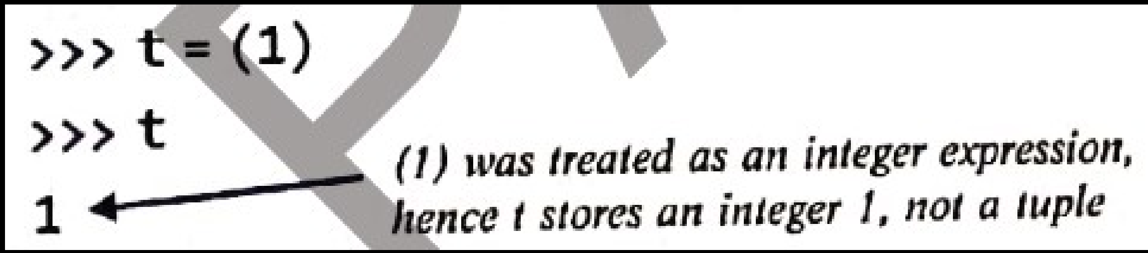
048 CREATING EMPTY TUPLE

```
t=1

print(t)
```

```
t=3

print(t)
```

Creating Tuples

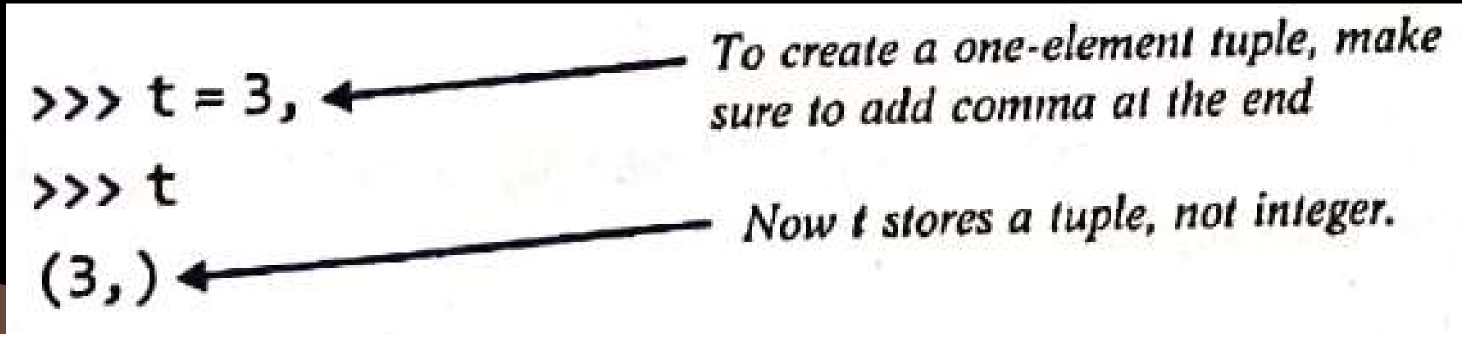
- **Creating single element tuple**
- Making a tuple with a single element is a tricky
- because if you just give a single element in round brackets, python considers it a value only,



```
>>> t = (1)
>>> t
1
```

(1) was treated as an integer expression, hence t stores an integer 1, not a tuple

- e.g., To construct a tuple with one element just add a comma after the single element as shown below:

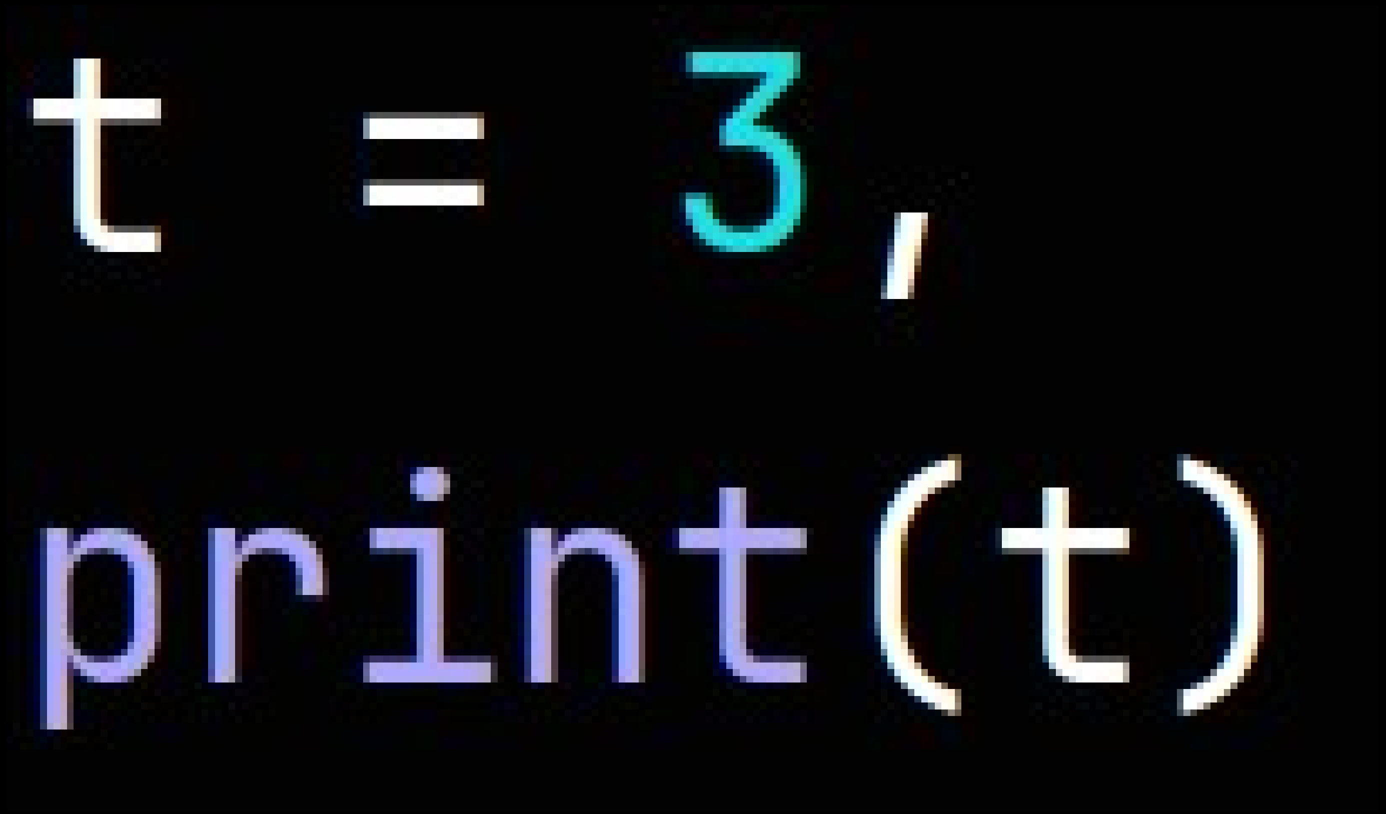


```
>>> t = 3,
>>> t
(3,)
```

To create a one-element tuple, make sure to add comma at the end

Now t stores a tuple, not integer.

049 CREATING SINGLE ELEMENT TUPLE



```
t = 3,
print(t)
```

Creating Tuples

- **Creating Tuples from existing sequence**
- You can also use the built-in tuple type object (tuple( ))
- to create Tuples from sequences as per the syntax given below:

```
T = tuple ( <sequence> )
```

- where <sequence> can be any kind of sequence object including strings, lists and tuples.

050 CREATING TUPLES FROM EXISTING SEQUENCE

```python
t1=tuple("hello")
print(t1)
# ('h','e','l','l','o')
l=['w','e','r','t','y' ]
t2=tuple(l)
print(t2)
```

Creating Tuples

- **Creating Tuples From Keyboard Input**
- You can use this method of creating tuples single characters or single digits via keyboard input.
- Consider the code below :

051 CREATING TUPLES FROM KEYBOARD INPUT

```
t1=tuple(input('enter tuple elements'))
print(t1)
```

Tuples Vs. Lists

- Tuples and lists are very similar yet different.
- This this section is going to talk about the same.

Tuples Vs. Lists

- **Similarity between Tuple and Lists**
- Tuples are similar to lists in following ways :
- **Length**
  - Function len (T) returns the number of items (count) in the tuple T.

Tuples Vs. Lists

- **Indexing and Slicing**
  - T [i] returns the item at index I (the first item has index 0),
  - and T [i:I] returns a new tuple, containing the objects between I and j.

052 INDEXING AND SLICING TUPLES

```
l = (1,3,5,4,6,8,7,9)
print (l[0:10:2])
print (l[2:10:3])
l = l[3:-2]
print (l)
l = l[3:7]
print (l)
```

Tuples Vs. Lists

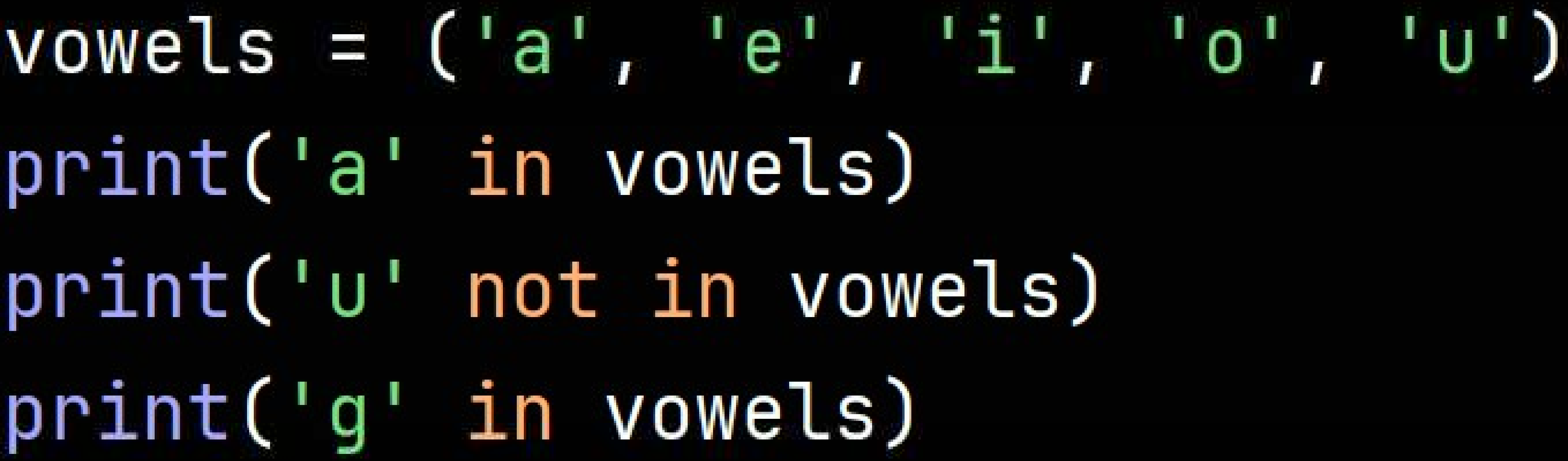
Membership Operators

- Both in and not in operators work in Tuples also.
- That is tells if an eliminate is present in the tuple or not and not in does the opposite.

053 MEMBERSHIP OPERATOR IN & NOT IN

```
tuple = (1, 2, 3, 4, 5)
print(5 in tuple)
print(3 not in tuple)
print(6 in tuple)
```

053 MEMBERSHIP OPERATOR IN & NOT IN

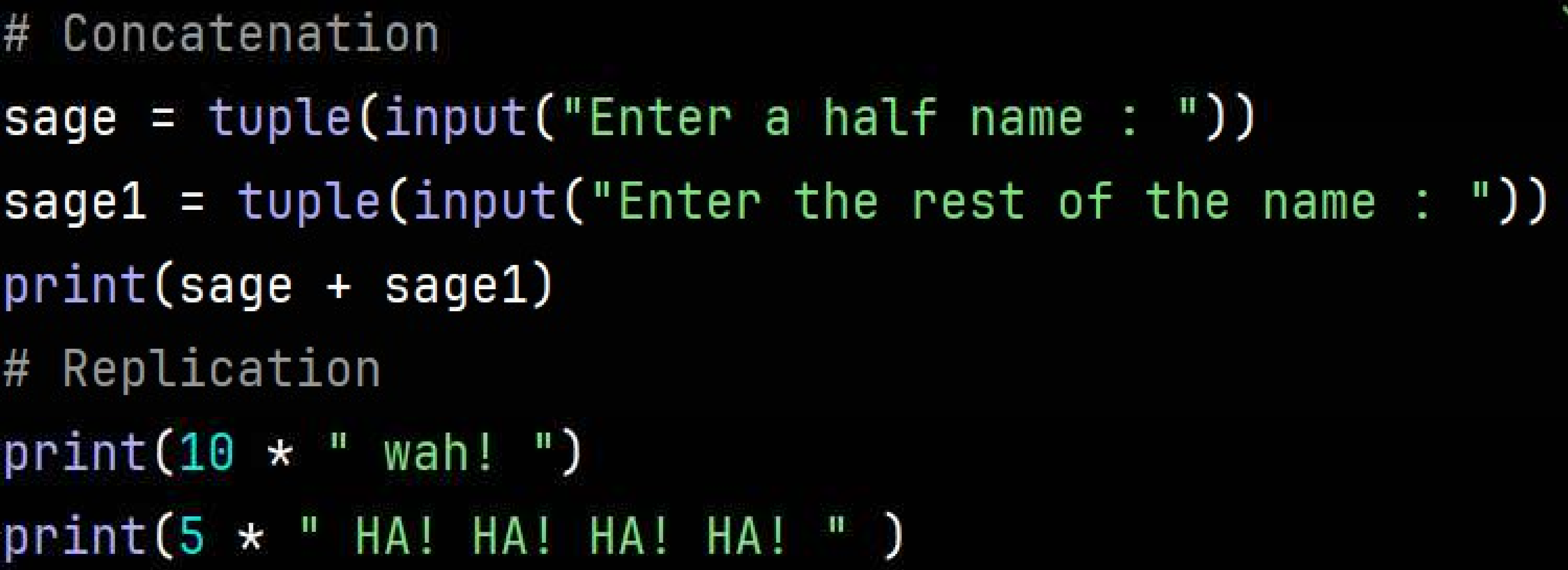


```python
vowels = ('a', 'e', 'i', 'o', 'u')
print('a' in vowels)
print('u' not in vowels)
print('g' in vowels)
```

Tuples Vs. Lists

- **Concatenation and replication operators + and ***
  - The + operator adds one tuple to the end of another.
  - The * operator repeats a tuple.

054 CONCATENATION AND REPLICATION OPERATORS + AND MULTIPLICATION



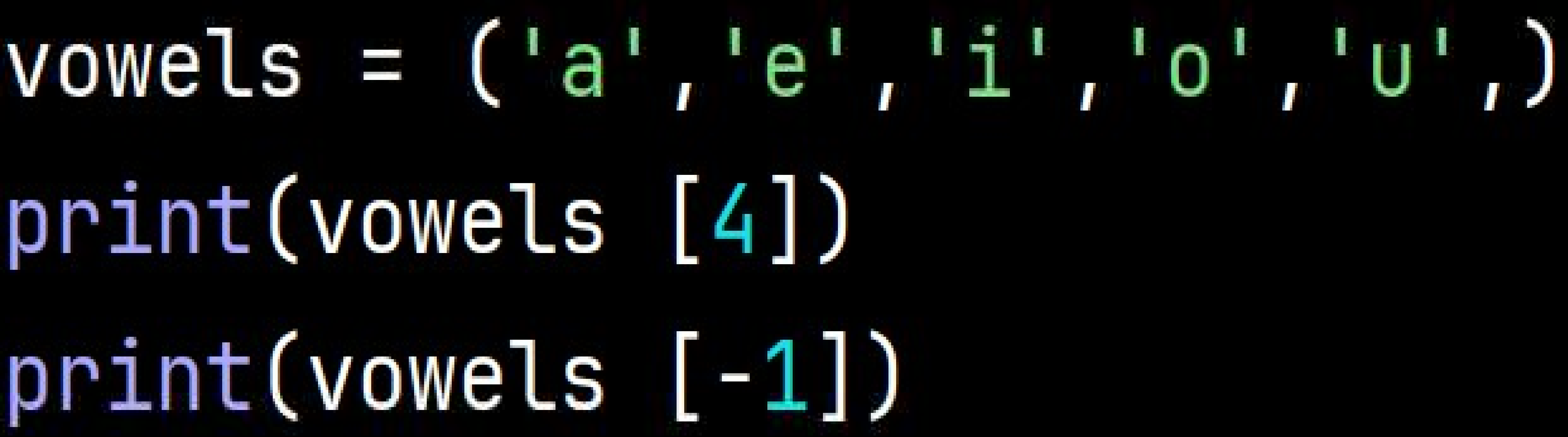
```python
# Concatenation
sage = tuple(input("Enter a half name : "))
sage1 = tuple(input("Enter the rest of the name : "))
print(sage + sage1)
# Replication
print(10 * " wah! ")
print(5 * " HA! HA! HA! HA! " )
```

Tuples Vs. Lists

- **Assessing individual elements**
- The individual elements of a couples are accessed through they are indexing given in square brackets.
- Consider the following examples:

```
>>>Vowels=('a', 'e', 'I', 'o', 'u')
>>>Vowels [4]
'u'
>>>Vowels[-1]
'u'
```

055A ACCESING INDIVIDUAL ELEMENTS



```
vowels = ('a','e','i','o','u',)
print(vowels [4])
print(vowels [-1])
```

Tuple Operations

- **Traversing a tuple**
- Traversing a tuple means assigning and processing each element of it.
- The for loop mix it easy to draw or loop over the items in a double as per following syntax:

055B TRAVERSING A TUPLE

```
T = ('P', 'u', 'r', 'e')
for a in T:
    print(a)
```

Tuple Operations

- **Repeating or Replicating Tuples**
- Like strings and list you can use * operator to replicate double specified number of times e.g.,

```
>>>TPL 1*3
(1, 3, 5, 1, 3, 5, 1, 3, 5)
```

- Like strings and list you can only use and integer with a operator when typing to replicate a tuples

056 REPEATING OR REPLICATING TUPLES

```
a = (1,2,3,4,5)
print(a*int(input(" Enter the numbers for the tuples: ")))


b = ('v','i','v','a')
print(b*int(input(" Enter the numbers for the tuples: ")))
```

Tuple Operations

- **Slicing The Tuples**
- Double slices like list slices or string slices are not sub part of the tuple extracted out.
- You can use in texting of couples elements to create apple slices as per following format:

  Sep=T[start : stop]
- The call that index on last limit is not included in the trouble slice.

057 SLICING TUPLES

```
T = (1,2,3,4,5,6,7,8,9,10)
Seq = T[0:10:2]
print(Seq)
```

Tuple Operations

- **Unpacking Tuples**
- Creating article from a set of values is called packing and its reserve,
- i.e., creating individual value from a tuple's elements is called and unpacking
- Unpacking is done as per syntax:

  <Variable 1>,<variable 2>,<variable 3>,...=t

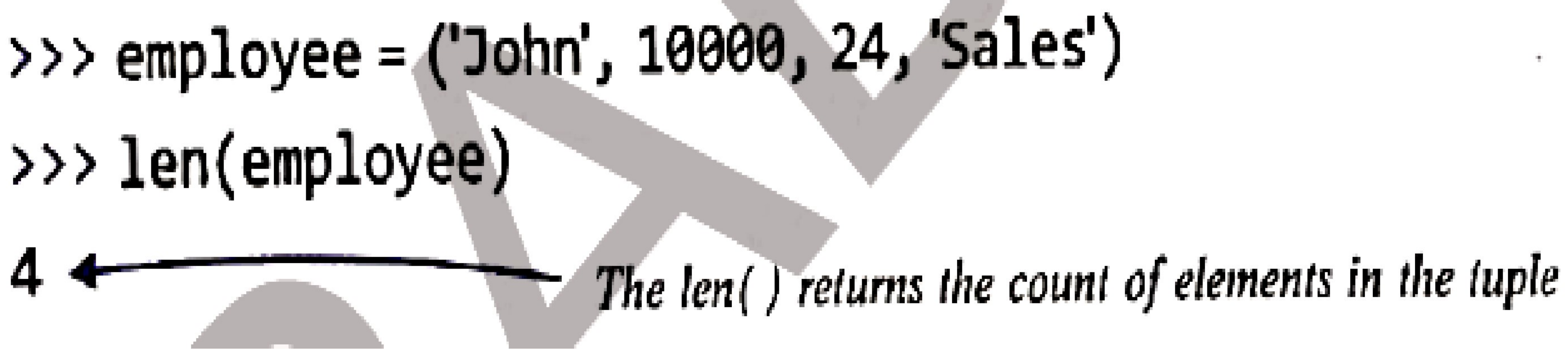
058 UNPACKING TUPLES

```
a = (1,2,'A','B')

w,x,y,z = a

print(w,"-",x,"-",z)
```

Tuple Functions and Methods

1. The len ( ) Method

- This method returns length of the tuple,
- i.e., The count of elements in the tuple.
- Syntax : len (<tuple>)



```
>>> employee = ('John', 10000, 24, 'Sales')
>>> len(employee)
4
```

The len( ) returns the count of elements in the tuple

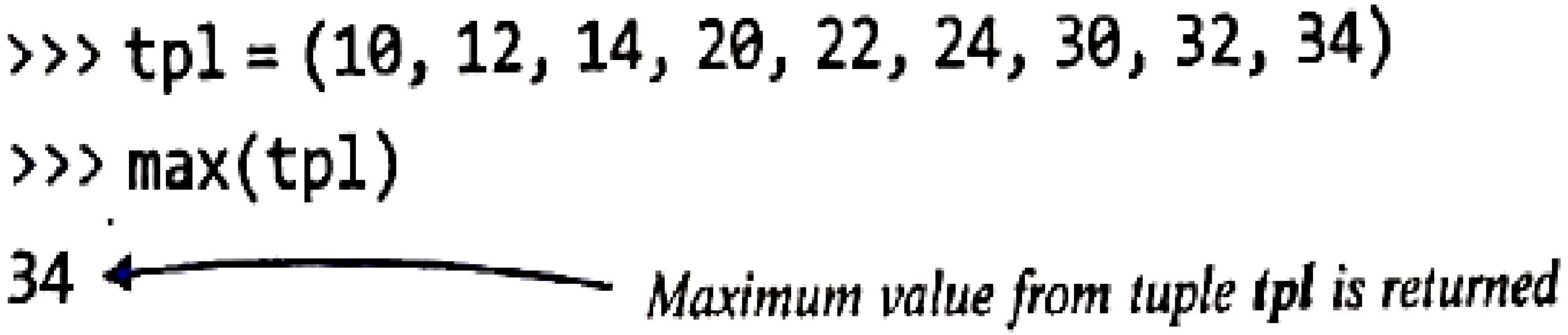
059 THE LEN () METHOD

```
a=('John',10000,24,'Sales')
len(a)
print(a)
```

Tuple Functions and Methods

2. The max ( ) Method

- This method returns the element from the tuple having maximum value.
- Syntax :   max (<tuple>)



```
>>> tpl = (10, 12, 14, 20, 22, 24, 30, 32, 34)
>>> max(tpl)
34
```

Maximum value from tuple tpl is returned

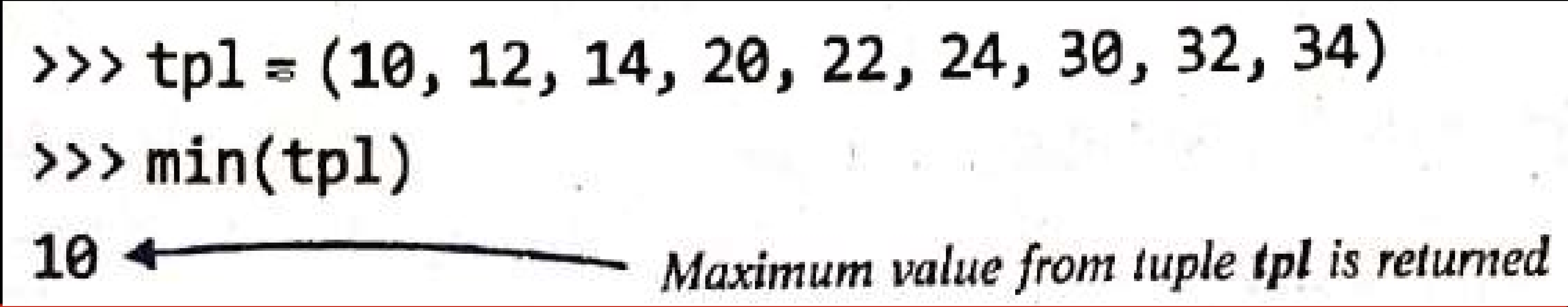
060 THE MAX () METHOD

```
a = (2,6,5,7,8,5)
max(a)
print(max(a))
```

Tuple Functions and Methods

3. The min ( ) Method

- This method returns the element from the tuple having Minimum value.
- Syntax : min (<tuple>)



```
>>> tpl = (10, 12, 14, 20, 22, 24, 30, 32, 34)
>>> min(tpl)
10
```

Note : like Max ( ), for men ( ) to work, the elements of couples should be of same type.

061 THE MIN () METHOD

```
tpl = (-2,-6,-5,-7,-8)
min(tpl)
print(min(tpl))
```

Tuple Functions and Methods

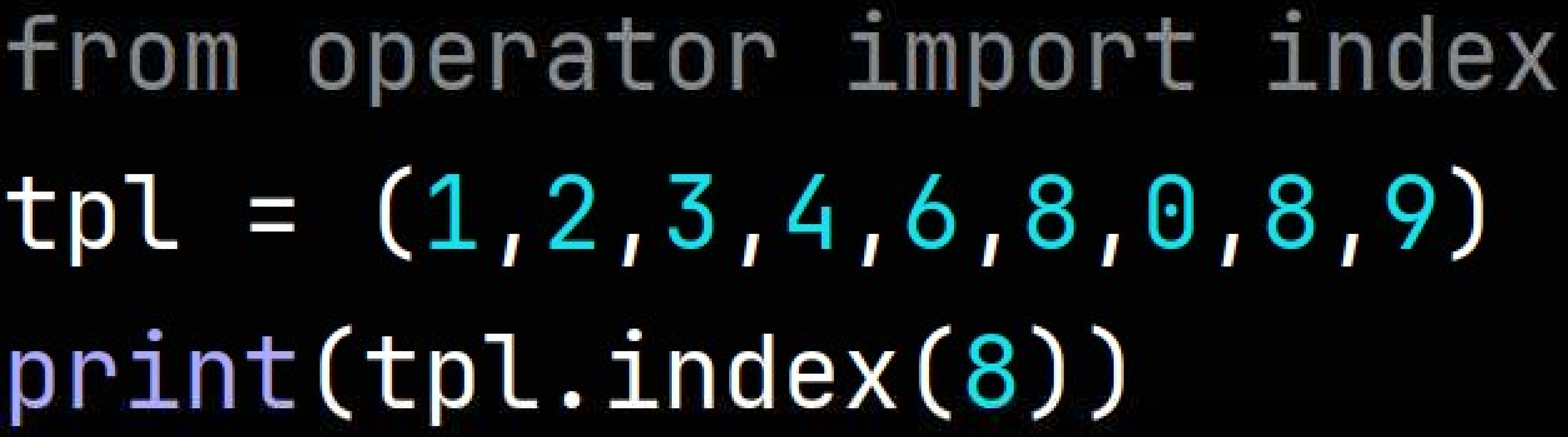
4. The index ( ) Method

- it returns the index of an existing element of a tuple.
- syntax: <tuplename> . Index (<item>)

```
>>> t1=[3456.0]
>>>T1 point index (5)
2
```

- But if the given item does not exist in trouble, it raises value error expectation2m

062 THE INDEX () METHOD

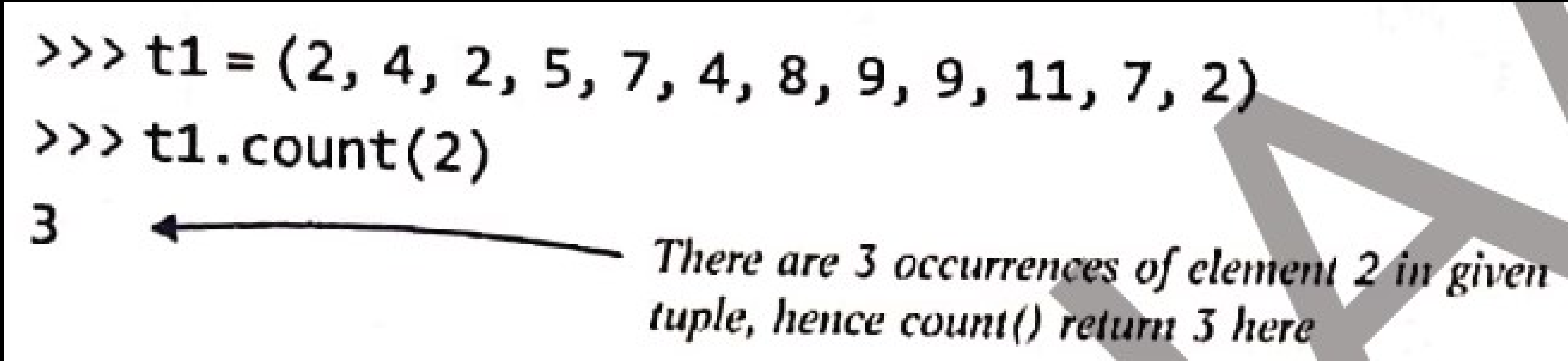


```
from operator import index
tpl = (1,2,3,4,6,8,0,8,9)
print(tpl.index(8))
```

Tuple Functions and Methods

5. The count ( ) Function

- The count ( ) method written the count of a number element/object in a given sequence (list/tuple).
- Syntax: <sequence name>.count (<object>)



```
>>> t1 = (2, 4, 2, 5, 7, 4, 8, 9, 9, 11, 7, 2)
>>> t1.count(2)
3
```

There are 3 occurrences of element 2 in given tuple, hence count() return 3 here

063 THE COUNT () METHOD

```
tpl = (1,2,3,4,6,8,0,8,9)
print(tpl.count(8))
```

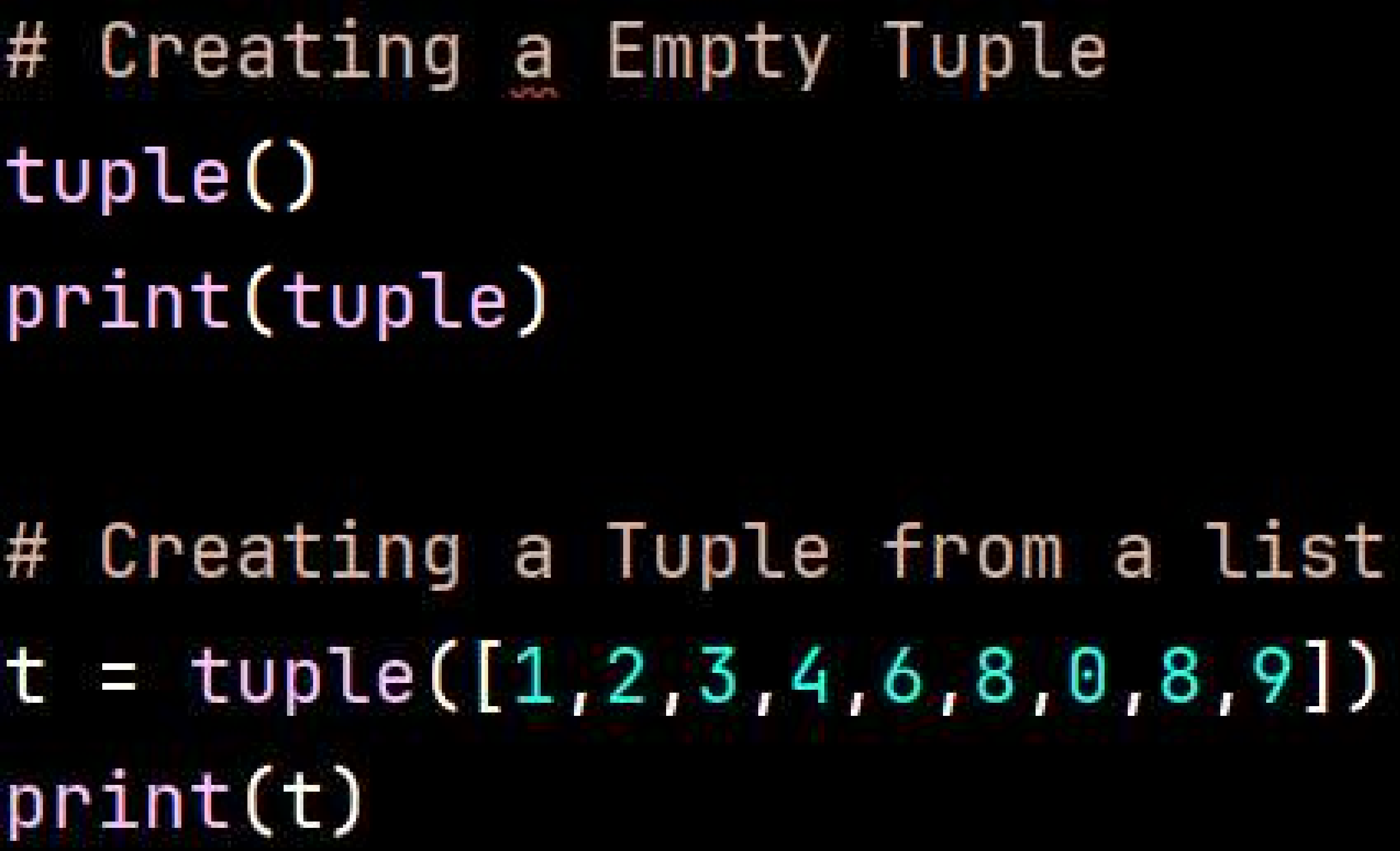
Tuple Functions and Methods

6. The tuple ( ) Function

- This method is actually constructor method that can be used to create tuples from different types of values.
- Syntax: tuple (<sequence>)

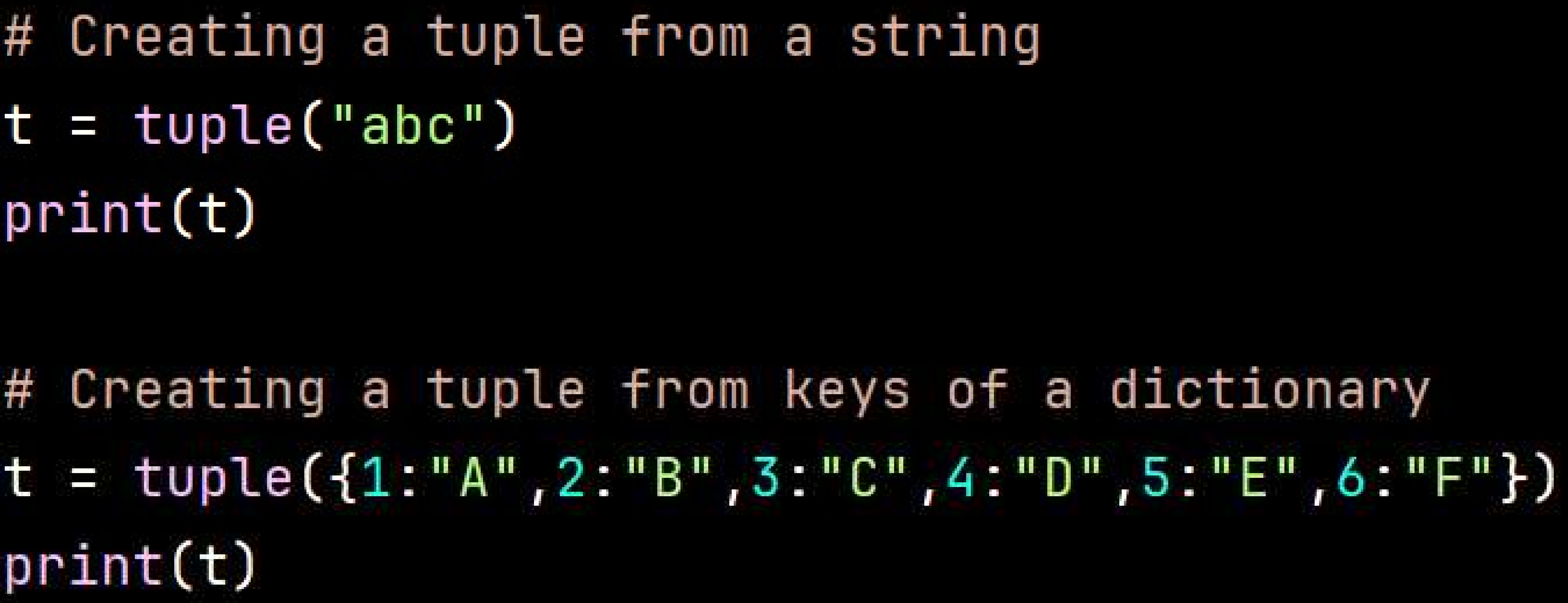
Note : With tuple ( ), are argument must be a sequence type i.e., a string or a list or a dictionary,

064 THE TUPLE () FUNCTION



```python
# Creating a Empty Tuple
tuple()
print(tuple)


# Creating a Tuple from a list
t = tuple([1,2,3,4,6,8,0,8,9])
print(t)
```

064 THE TUPLE () FUNCTION



```python
# Creating a tuple from a string
t = tuple("abc")
print(t)

# Creating a tuple from keys of a dictionary
t = tuple({1:"A",2:"B",3:"C",4:"D",5:"E",6:"F"})
print(t)
```