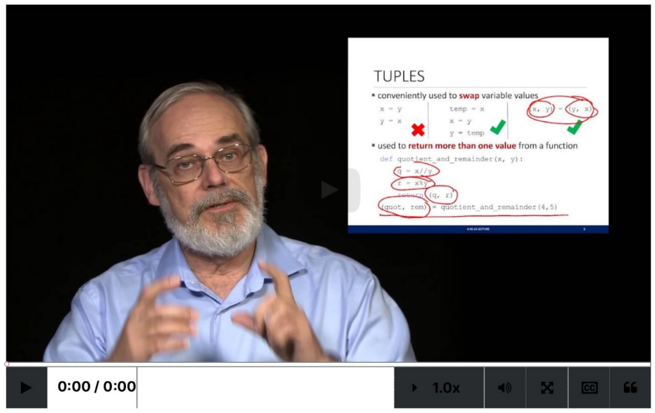
Video: Tuples



TUPLES

- conveniently used to **swap** variable values

```
x = y          temp = x          (x, y) = (y, x)
y = x          x = y
               y = temp
```

- used to **return more than one value** from a function

```
def quotient_and_remainder(x, y):
    q = x//y
    r = x%y
    return (q, r)
(quot, rem) = quotient_and_remainder(4,5)
```

Film
Pobierz plik filmowy

Transkrypcje
Pobierz SubRip (.srt) file
Pobierz Text (.txt) file

Notatki
Pobierz notatkę

Video: Tuples

Ukryj dyskusję

Temat Lecture 5 / Video: Tuples

Dodaj wpis

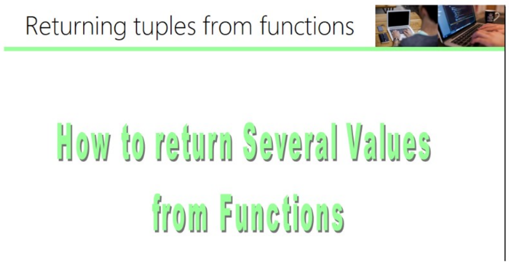
‹ Wszystkie wpisy

[Tutorial] How to return several values from a function: Behind the Scenes

discussion posted 15 days ago by **Kiara-Elizabeth** (Community TA)

How to return several values from functions

Hi! Welcome to this new session where we will learn **HOW TO RETURN SEVERAL VALUES FROM A FUNCTION!** Let's get started! :)

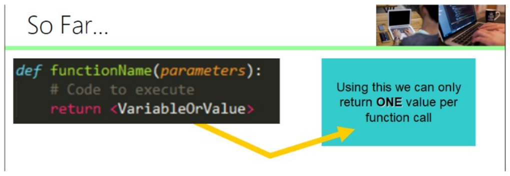


Returning tuples from functions

How to return Several Values from Functions

What we've learned so far is that we can return a value from a function to the scope that called the function and then save it to a variable for later use.

But... what if we want to return more than one value?



So Far...

```
def functionName(parameters):
    # Code to execute
    return <VariableOrValue>
```

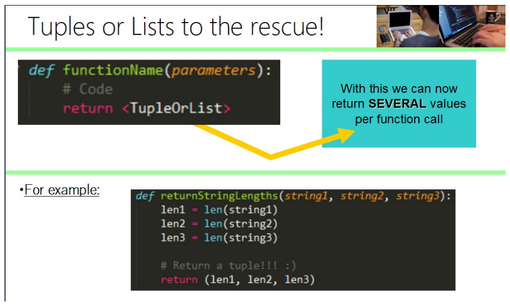
Using this we can only return **ONE** value per function call

This is where our newly discovered friends come into play, **TUPLES AND LISTS!**

If we return a tuple or a list from a function, we can return several values and assign them individually to their corresponding variables

Hide Notes

(We will see this in detail in the next diagrams).


Tuples or Lists to the rescue!

```
def functionName(parameters):
    # Code
    return <TupleOrList>
```

With this we can now return **SEVERAL** values per function call

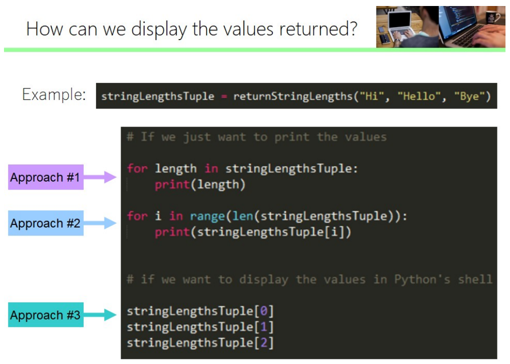
•For example:

```
def returnStringLengths(string1, string2, string3):
    len1 = len(string1)
    len2 = len(string2)
    len3 = len(string3)

    # Return a tuple!!! :)
    return (len1, len2, len3)
```

We've learned that we can display the values returned by a function using print().

Since lists and tuples are iterables, we can iterate over their elements using a for/in loop like the examples below. We can also access each individual element by its corresponding index.


How can we display the values returned?

Example:
```
stringLengthsTuple = returnStringLengths("Hi", "Hello", "Bye")
```

```
# If we just want to print the values

for length in stringLengthsTuple:
    print(length)

for i in range(len(stringLengthsTuple)):
    print(stringLengthsTuple[i])


# if we want to display the values in Python's shell

stringLengthsTuple[0]
stringLengthsTuple[1]
stringLengthsTuple[2]
```

USE VARIABLES TO STORE VALUES RETURNED

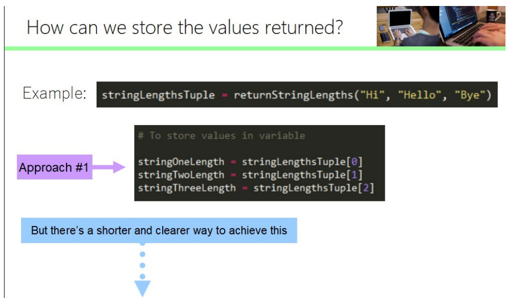
But… wait a minute! What if we want to store the values returned by the function call? Can we apply the principles we've learned? Yes, you can. But there is a new syntax that will allow you to store each element in the tuple into its corresponding variable.

Let's see why we need this…

In this example, we are using the function defined above, that returns a tuple with the length of each argument. In this case, (2, 5, 3).

To store them in separate variables, we would first need to create a new variable to store the entire tuple and then index individually and assign them to their variable. This is one way to achieve this but…

Keep reading to find out how you can do this in ONE LINE!


How can we store the values returned?

Example:
```
stringLengthsTuple = returnStringLengths("Hi", "Hello", "Bye")
```

```
# To store values in variable

stringOneLength = stringLengthsTuple[0]
stringTwoLength = stringLengthsTuple[1]
stringThreeLength = stringLengthsTuple[2]
```

But there's a shorter and clearer way to achieve this

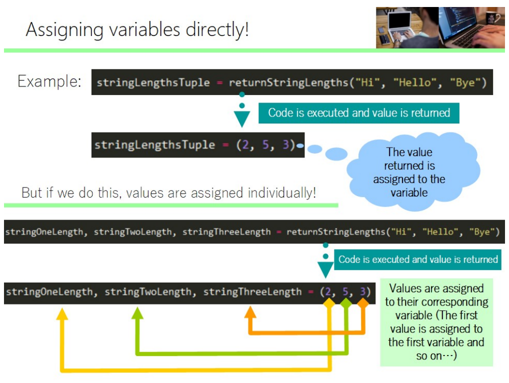
To assign the variables individually in ONE LINE:

- On the left hand side we create as many variables as the number of values contained in the tuple returned by the function call. (In this case, 3 variables for a tuple that contains 3 values)
- On the right hand side we call the function

On the diagram below you can see this process broken down into steps with a "Behind the scenes" look of what happens when the value is

Hide Notes

returned.

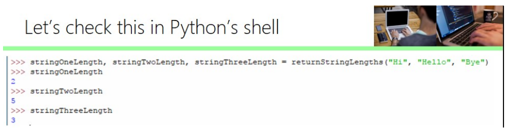


Assigning variables directly!

Example:
```
stringLengthsTuple = returnStringLengths("Hi", "Hello", "Bye")
```
Code is executed and value is returned
```
stringLengthsTuple = (2, 5, 3)
```
The value returned is assigned to the variable

But if we do this, values are assigned individually!
```
stringOneLength, stringTwoLength, stringThreeLength = returnStringLengths("Hi", "Hello", "Bye")
```
Code is executed and value is returned
```
stringOneLength, stringTwoLength, stringThreeLength = (2, 5, 3)
```
Values are assigned to their corresponding variable (The first value is assigned to the first variable and so on…)

Let's check these values in Python's shell. As you can see, the values are assigned individually and you can now use these variables later in your code. Yes! : D

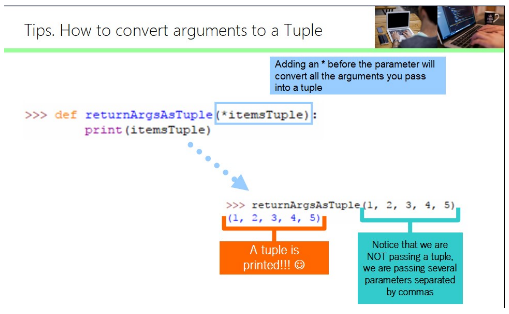


Let's check this in Python's shell

```
>>> stringOneLength, stringTwoLength, stringThreeLength = returnStringLengths("Hi", "Hello", "Bye")
>>> stringOneLength
2
>>> stringTwoLength
5
>>> stringThreeLength
3
```

TIPS

You can convert the arguments you pass into a function into a tuple by adding an asterisk before the function's parameter. This will convert all the arguments you pass in separated by commas into a single tuple that you can use inside your function.



Tips. How to convert arguments to a Tuple

Adding an * before the parameter will convert all the arguments you pass into a tuple

```
>>> def returnArgsAsTuple(*itemsTuple):
        print(itemsTuple)
```

```
>>> returnArgsAsTuple(1, 2, 3, 4, 5)
(1, 2, 3, 4, 5)
```

A tuple is printed!!! ☺

Notice that we are NOT passing a tuple, we are passing several parameters separated by commas

Hope this helps!

If you have any questions, please do not hesitate to post them in forums or right below this post. Community TAs and your fellow classmates will be there to help you :)

Estefania.

This post is visible to everyone.

Add a Response

2 odpowiedzi

Untangled1351
7 days ago

How can I see all similar parameters such as *itemsTuple? I imagine there must be on for lists as well?

In

`def f(*vars):`

vars is a tuple. But the elements of vars can include any objects including lists.

`f(1,2,[1,2],'me too',3)`

Remember that a tuple is effectively just an immutable list.

posted 7 days ago by __kiwi__ (Community TA)

Hide Notes

Dodaj komentarz

ElsonB
5 days ago

While messing with this code:

```
def returnStringLengths(s1, s2, s3):
  len1= len(s1)
  len2= len(s2)
  len3= len(s3)

  return (len1, len2, len3)

returnStringLengths('blue', '23', '5')
stringLengthsTuple= returnStringLengths('Hi', 'Hello', 'Bye')

for i in stringLengthsTuple:
  print(length)
```

as you can see before the print, i accidentally typed in **i** instead of **length** and the result was **3 3 3** instead of an error. I ran it again in a clear environment in colab notes and still got the same. I'm confused.

I get an "NameError: name 'length' is not defined"

posted 3 days ago by **Slaski907**

Yes. `length` doesn't seem to be bound to anything. Perhaps a line like `length = returnStringLengths('blue', '23', '5')` was intended in your code ElsonB. Or something similar?

posted 3 days ago by **__kiwi__** (Community TA)

Dodaj komentarz

Wyświe‹ Previous ‹ Next ›

Add a response:

edX

About
Affiliates
edX for Business
Open edX
Careers
News

Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap

Connect

Blog
Contact Us
Help Center
Media Kit

Hide Notes

Hide Notes