global variables
Global variables

Start of transcript. Skip to the end.

>> In this section, we'll be looking at global variables,

and how they can be used within your programs.

Something important to note is that if

you have a value for a global variable,

that value will be expressed in

all of the functions within your program.

So, they should be used with

0:00 / 4:25 1.0x

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Concepts

A global variable is assigned outside all functions and lives within the global scope of the module. It exists from the time of its assignment until the program ends. Global variables are visible to all functions within the module and can be used within their different local scopes. Additionally, global variables can be used by expressions in the global scope. The value of a global variable can be changed from the global scope, and can also be modified from a local scope using the `global` statement (i.e. `global x = 4`). If (`global`) was not used, a local variable would be defined instead, and any changes to this new variable will not affect the global variable that bears the same name.

Global variables are highly discouraged because they make your code hard to understand and follow, imagine that 20 functions written by different developers are all changing one global variable. Tracking the functionality of the program will be very challenging. Global variables are covered in this lesson because some developers use them for very specialized applications when there are no alternatives. But please know, that you can write very complicated Python scripts without the need to use global variables. It is OK, however, to use constant global variables that are accessible from all functions but will not (and cannot) be changed.

Generally speaking:

- Global variables are accessible from local scopes
- Global variables can be changed from the global scope
- Global variables can be changed from a local scope using the `global` statement
- If a local variable shares the same name with a global variable, changes in the local will not affect the global

---

Examples

In the following examples, you will see how to define, read, and modify global variables.

Global variables are accessible from local scopes

Numerical variable

```
# Global variable
PI = 3.14

# Compute the area of a circle
def circle_area (radius):
    # PI is accessible from this local scope
    area = PI * radius ** 2
    return area

# Global scope
a = circle_area(4)
print("circle area =", a)
```

String variable

```
# Global variable
vowels = 'AaEeIiOoUiYy'

# count the number of vowels in a sentence
def count_vowels(sentence):
    # vowels is accessible from this local scope
    count = 0
    for c in sentence:
        if c in vowels:
            count = count + 1
    return count

# Global scope
s = 'Monty Python'
print('Number of vowels in "{:s}" = {:d}'.format(s, count_vowels(s
```

Global variables can be changed from a local scope

When the value of a global variable is changed from a local scope, it stays changed even after the local scope has been destroyed

Numerical Variable

```
# Global variable
PI = 3.14

# Compute the area of a circle
def circle_area (radius):
    # Define PI as a global variable in this scope
    global PI
    PI = 3.14159265359 # more accurate pi
    area = PI * radius ** 2
    return area

# Global scope
print("PI =", PI)
a = circle_area(4)
print("More accurate circle area =", a)
print("Updated PI =", PI) # Global PI changed in circle_area
```

String variable

```
# String global variable
planet = 'Mercury'

# function to change the current planet
def planet_change(new_planet):
    # Define planet as a global variable in this scope
    global planet
    planet = new_planet

# Global scope
print("Planet =", planet)
planet_change('Mars')
print("Planet =", planet)
```

Assigning a value to a global variable in a local scope without `global`

Changes to a local variable sharing the same name as a global variable is not reflected in the global variable

Numerical variables

```
# Global variable
PI = 3.14

# Compute the area of a circle
def circle_area (radius):
    # Assigning a value to PI without (global) makes it a local va
    PI = 3.14159265359 # more accurate pi
    area = PI * radius ** 2
    return area

# Global scope
print("PI =", PI)
a = circle_area(4)
print("More accurate circle area =", a)
print("Unchanged PI =", PI) # Global PI didn't change
```

String variable

```
# String global variable
planet = 'Mercury'

# function to change the current planet
def planet_change(new_planet):
    planet = new_planet # planet is a local variable


# Global scope
print("Planet = ", planet)
planet_change('Mars')
print("Planet = ", planet)
```

Global variables can be changed from the global scope

Numeric variable

```
# Global variable
PI = 3.14

# Compute the area of a circle
def circle_area (radius):
    # PI is accessible from this local scope
    area = PI * radius ** 2
    return area

# Global scope
# PI is changed before it is used in circle_area
PI = 0
a = circle_area(4)
print("PI =", PI)
print("Wrong circle area =", a)
```

String variable

```
# String global variable
planet = 'Mercury'

# function to change the current planet
def planet_change(new_planet):
    planet = new_planet # planet is a local variable


print("Planet = ", planet)
planet_change('Mars')
print("Planet = ", planet) # global variable (planet) did not chan
planet = "Earth" # changing global variable (planet)
print("Planet = ", planet)
```

Task 3

Global variables

Currency Converter

```
# [ ] The program below converts amount from US Dollars to Indian
# complete the function USD2INR so it performs the conversion

XCHANGE_RATE = 63.6856 # = 1 USD

def USD2INR(amount):
    """
    Convert amount from US Dollars to Indian Rupees.

    Use XCHANGE_RATE

    args:
        amount: US dollar amount (float)

    returns:
        value: the equivalent of amount in Indian Rupees (float)
    """
    #TODO: Your code goes here
    return value

print("Current exchange rate $1 USD = {} INR".format(XCHANGE_RATE)
amount = 220 #USD
inr = USD2INR(amount)
print("${} = {}".format(amount, inr))
```

```
# [ ] The following program calculate the equivalent of $220 USD i
# and perform the conversion again
# Complete the functions USD2INR and change_rate so they function


XCHANGE_RATE = 63.6856 # = 1 USD

def USD2INR(amount):
    """
    Convert amount from US Dollars to Indian Rupees.

    Use XCHANGE_RATE

    args:
        amount: US dollar amount (float)

    returns:
        value: the equivalent of amount in Indian Rupees (float)
    """
    #TODO: Your code goes here
    return value

def change_rate():
    """
    Change the exchange rate to 63.6782

    args:
        None

    returns:
        None
    """
    #TODO: Your code goes here

print("Current exchange rate $1 USD = {} INR".format(XCHANGE_RATE)
amount = 220 #USD
inr = USD2INR(amount)
print("${} = {}".format(amount, inr))

print()
change_rate()
print("After changing the exchange rate $1 USD = {} INR".format(XC
inr = USD2INR(amount)
print("${} = {}".format(amount, inr))
```

Flip

```
# [ ] In the following program, the function `flip()` is designed
# Fix the `UnboundLocalError` exception without changing the expre

NUMBERS = [1, 2, 3, 4, 5, 6]

def flip():
    NUMBERS = NUMBERS[-1::-1]


print("Before flipping, NUMBERS =", NUMBERS)
flip()
print("After flipping, NUMBERS =", NUMBERS)
```

Learn About Verified Certificates