local variables
Local variables

Start of transcript. Skip to the end.

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

variable scope within Python

and what it means to be a local versus a global variable.

>> Let's look at the scope of

variables when they are local,

inside of a function,

and global, out in the general area of a program.

Here, we have a function

0:00 / 3:32 1.0x

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Concepts

A variable in Python lives within a scope, the scope determines how the variable is accessed and when it is deleted. A variable scope is determined by the place where it is initially assigned; there are two types of scopes: local and global. Parameters passed to a function and variables assigned within it are said to be within the local scope of the function and they are called local variables; on the other hand, variables assigned outside all functions in a module are said to be within the global scope and they are called global variables.

Local scopes are created when a function is called and they get destroyed when the function return to its caller. This means, that you might have several local scopes that have different local variables within them. These local variables can be accessed and changed within their own local scope; however, they cannot interact with variables from other local scopes and they cannot be accessed from the global scope. This is important because it allows you to use the same variable name in different functions without worrying about name clashes or collision of variables.

The concept of a local scope guides you to think about functions as black boxes that can only interact with your code through arguments and returned values. When developing a function, you do not have to worry about a variable name being used in another function, because you know they will be local within their own function and can only be accessed from within it.

Generally speaking:

- Local variables cannot be read or modified from the global scope
- Local variables cannot be read or modified from other local scopes
- The same variable name can be used in different functions without causing a clash

---

Examples

In the following examples, you will see functions to compute the areas and volumes of different geometric figures. The variable name `area` will be used in all functions to demonstrate the concepts of local scope and global scope. The demonstrated concepts apply to other data types as well (i.e. lists and strings)

Local variables cannot be read or modified from the global scope

```
# Compute the area of a square
def square_area (side):
    # area is a local variable in square_area
    area = side ** 2
    return area


# Global scope
square_area(2)

# area is not within scope anymore and cannot be
# accessed from this global scope
print(area)
```

Local variables cannot be read or modified from other local scopes

A local variable in one function cannot be accessed from another function

```
# Compute the area of a square
def square_area (side):
    # area is a local variable in square_area
    area = side ** 2
    return area

# Compute the volume of a cube
def cube_volume (side):
    # cube volume = area of base X side
    volume = area * side # area is not defined within this scope
    return volume

# Global scope
s = 2
square_area(s)
# area was deleted when the local scope of square_area was destroy
cube_volume(s)
```

The same variable name can be used in different functions without causing a clash

```
# Compute the area of a square
def square_area (side):
    # area is a local variable in square_area
    # area does not conflict with the variable area in rectangle_a
    area = side ** 2
    print("square area =", area)

# Compute the area of a rectangle
def rectangle_area (length, width):
    # area is a local variable in rectangle_area
    # area does not conflict with the variable area in square_area
    area = length * width
    print("rectangle area =", area)

# Global scope
square_area(2)
rectangle_area(2, 3)
```

Task 1

Local variables

Fix the errors

Return to the examples that generated errors and fix them so they function as expected.

```
# [ ] Fix the program below so it displays the area of a square wi

# Compute the area of a square
def square_area (side):
    # area is a local variable in square_area
    area = side ** 2
    return area


# Global scope
square_area(2)

# area is not within scope anymore and cannot be
# accessed from this global scope
print(area)
```

```
# [ ] Fix the program below so it displays the area of a square wi
# and the volume of a cube with side = 2

# Compute the area of a square
def square_area (side):
    # area is a local variable in square_area
    area = side ** 2
    return area

# Compute the volume of a cube
def cube_volume (side):
    # cube volume = area of base X side
    volume = area * side # area is not defined within this scope
    return volume

# Global scope
s = 2
square_area(s)
# area was deleted when the local scope of square_area was destroy
cube_volume(s)
```

Currency Converter

The same name for arguments and local variables can be used across different functions

```
# [ ] The program below converts US Dollars to Euros, British Poun
# Complete the functions USD2EUR, USD2GBP, USD2JPY so they all ret

def USD2EUR(amount):
    """
    Convert amount from US Dollars to Euros.

    Use 1 USD = 0.831467 EUR

    args:
        amount: US dollar amount (float)

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

def USD2GBP(amount):
    """
    Convert amount from US Dollars to British Pounds.

    Use 1 USD = 0.739472 GBP

    args:
        amount: US dollar amount (float)

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

def USD2JPY(amount):
    """
    Convert amount from US Dollars to Japanese Yens.

    Use 1 USD = 112.656 JPY

    args:
        amount: US dollar amount (float)

    returns:
        value: the equivalent of amount in Japanese Yens (float)
    """
```

```
    #TODO: Your code goes here
    return value

def main():
    amount = float(input("Enter amount in USD: $"))

    # In Euros
    eur = USD2EUR(amount)

    # In British Pounds
    gbp = USD2GBP(amount)

    # In Japanese Yens
    jpy = USD2JPY(amount)

    print("${:.2f} USD = {:.2f} EUR, {:.2f} GBP, {:.2f} JPY".forma

if __name__ == '__main__':
    main()
```

Learn About Verified Certificates