

Course > Durabl... > Variabl... > isolate...

isolated local scopes Isolated local scopes

Start of transcript. Skip to the end.

>> In this section,

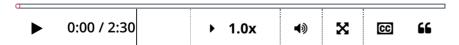
we're going to be using a function

to call a sub-function and exploring how that impacts local

variable versus global variable scope.

>> Let's take a further look at

variable scope as it is in regards to functions.



Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Concepts

When a function calls a subfunction, the current variables within the function scope are stored in memory, and another temporary local scope is created to accommodate the new subfunction variables. The temporary local scope is destroyed when the subfunction returns, at that point the original local scope becomes active again. This concept is demonstrated in the following example

Examples

In this example, the function area diff computes the area difference between a rectangle and a square. The function calls square area and rectangle area, and all three functions use a local variable called area without any clash. As you can see, choosing the same variable name in all three functions does not create any issue.

When area diff calls square area, the current local variables within area diff are stored in a location in memory called the stack, then a new local scope is created with new variables for square area. The local scope of area diff is still alive; however, it's inaccessible until square area returns. Both area diff and square area use the variable area; however, the two variables live in two different local scopes and cannot affect each other. After square area returns, the local scope of area diff becomes active again until calling rectangle_area, and the cycle repeats.

In summary, a variable called area is used in all three functions without any clash. The content of the three variables are kept separate because they belong to three different local scopes.

```
# 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
    return 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
    return area
# Compute the area difference between a square and a rectangle
def area diff (side, length, width):
    # square area
    areal = square area(side) # defines area in its local scope
    # rectangle area
    area2 = rectangle area(length, width) # defines area in its lo
    # area difference
    area = area2 - area1 # area is local in area diff local scope
    return area
# Call the area diff function
print("Area difference = ", area_diff(2, 2, 3))
```

Task 2

Isolated local scopes

Currency Converter

The same variable name can be used across different functions, even when the functions call each other.

```
# [ ] The program below converts US Dollars to British Pounds. How
# Complete the functions USD2EUR, EUR2GBP, and USD2GBP so they all
# You should use USD2EUR and EUR2GBP in USD2GBP, do not try to fin
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 EUR2GBP(amount):
    Convert amount from Euros to British Pounds.
    Use 1 EUR = 0.889358 GBP
    args:
        amount: Euros amount (float)
    returns:
        value: the equivalent of amount in GBP (float)
    .....
    #TODO: Your code goes here
    return value
def USD2GBP(amount):
    Convert amount from US Dollars to British Pounds.
    The conversion rate is unknown, you have to use USD2EUR and EU
    args:
        amount: US dollar amount (float)
    returns:
        value: the equivalent of amount in British Pounds (float)
```

```
.....
    #TODO: Your code goes here
    return value
def main():
    amount = float(input("Enter amount in USD: $"))
    # In British Pounds
    gbp = USD2GBP(amount)
    print("${:.2f} USD = {:.2f} GBP".format(amount, gbp))
if __name__ == '__main__':
    main()
```

Learn About Verified Certificates

© All Rights Reserved