Homework/Lab 3, DSE5002

Created 2/11/2025

Martin Omasta 10 Nov 25

See

Think Python https://allendowney.github.io/ThinkPython/chap05.html

5.14.1. Ask a virtual assistant

Ask a virtual assistant, “What are some uses of the modulus operator?”

Answer) Modulus (%) operator's primary use is in determining if a number is wholly divisible by another number, or if there is a remainder - and returning that remainder. This base function can then be extended to perform cyclical behaviors using it as a pointer of sorts in a list that needs to be read through multiple times. There are also uses in formatting output and finding out what the last digit of an integer is, among many others.

In this chapter, we saw two ways to write an if statement with three branches, using a chained conditional or a nested conditional. You can use a virtual assistant to convert from one to the other. For example, ask a VA, “Convert this statement to a chained conditional.”

if x == y: print('x and y are equal') else: if x < y: print('x is less than y') else: print('x is greater than y')

Copy and paste the code from the VA and figure out if it works. If it doesn't, fix it

In [3]:
```
def chained_conditional_test(x, y):
    if x == y:
        print('x and y are equal')
    elif x < y:
        print('x is less than y')
    else:
        print('x is greater than y')

chained_conditional_test(5, 5)
chained_conditional_test(4, 5)
chained_conditional_test(6, 5)
```

```
x and y are equal
x is less than y
x is greater than y
```

Ask a VA, “Rewrite this statement with a single conditional.”

if 0 < x: if x < 10: print('x is a positive single-digit number.')

If this doesn't work, fix it

In [4]:
```
tx = 5
if 0 < tx < 10:
    print('tx is a positive single-digit number.')
```

```
tx is a positive single-digit number.
```

Here's an attempt at a recursive function that counts down by two.

def countdown_by_two(n): if n == 0: print('Blastoff!') else: print(n) countdown_by_two(n-2)

In [6]:
```
def countdown_by_two(n):
    if n == 0:
        print('Blastoff!')
    else:
        print(n)
        countdown_by_two(n-2)

countdown_by_two(8)
```

```
8
6
4
2
Blastoff!
```

5.14.3. Exercise

If you are given three sticks, you may or may not be able to arrange them in a triangle. For example, if one of the sticks is 12 inches long and the other two are one inch long, you will not be able to get the short sticks to meet in the middle. For any three lengths, there is a test to see if it is possible to form a triangle:

If any of the three lengths is greater than the sum of the other two, then you cannot form a triangle. Otherwise, you can. (If the sum of two lengths equals the third, they form what is called a "degenerate" triangle.)

Write a function named is_triangle that takes three integers as arguments, and that prints either "Yes" or "No", depending on whether you can or cannot form a triangle from sticks with the given lengths. Hint: Use a chained conditional.

In [3]:
```
#function to determine if 3 integer sides form a triangle, if that triangle has are
#or if it is "degenerate" with 0 area

def is_triangle(side1, side2, side3):
    candidate_sides = [side1, side2, side3]
    candidate_sides.sort()
    if (candidate_sides[0] + candidate_sides[1]) == candidate_sides[2]:
```

```
        print("Yes: You've got a degenerate triangle, area == 0")
    elif (candidate_sides[0] + candidate_sides[1]) > candidate_sides[2]:
        print("Yes: That there is a triangle with area > 0")
    else:
        print("No:  Can't make a triangle from those 3 lengths of sides")

is_triangle(2, 2, 4)
is_triangle(2, 5, 4)
is_triangle(1, 1, 6)
is_triangle(6, 1, 1)
```

```
Yes: You've got a degenerate triangle, area == 0
Yes: That there is a triangle with area > 0
No:  Can't make a triangle from those 3 lengths of sides
No:  Can't make a triangle from those 3 lengths of sides
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: