Homework/Lab 3, DSE5002

Peter Gyorda April 6, 2025

Created 2/11/2025

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?”

In [ ]:
```
The modulus operator (`%`) in Python is a versatile arithmetic operator that return

## Common Uses of the Modulus Operator

1. **Finding remainders**: The most basic use is to find the remainder after divisi

   print(10 % 3)  # Returns 1 (10 divided by 3 equals 3 with remainder 1)




2. **Checking if a number is even or odd**: A number is even if `number % 2 == 0`.

3. **Checking divisibility**: To check if a number is divisible by another, use `nu

4. **Wrapping around values** (circular behavior):
   - Array/list index wrapping
   - Clock arithmetic (12-hour or 24-hour time cycles)
   - Rotating through a fixed set of options

5. **Generating periodic sequences**: Creating patterns that repeat after a certain

6. **Hash functions**: Used in some hashing algorithms to map data to fixed-size va

7. **Random number generation**: Some random number generators use modulus to const

8. **Cryptography**: Used in various encryption algorithms.

## Python Packages That Use Modulus

If you're working with mathematical operations involving modulus, these packages mi

- **NumPy**: For efficient modulus operations on arrays
- **SymPy**: For symbolic mathematics including modular arithmetic
```

```
- **math**: Python's built-in math module has functions like `math.fmod()` for floa
- **random**: Uses modulus internally for generating random numbers within ranges

The modulus operator is a fundamental tool in programming that helps solve many com
```

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 [2]:
```
x=5
y=6

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')
```

```
x is less 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 ti

In [3]:
```
X=7

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

```
x 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 [4]:
```
def countdown_by_two(n): if n == 0: print('Blastoff!') else: print(n) countdown_by_
```

```
  Cell In[4], line 1
    def countdown_by_two(n): if n == 0: print('Blastoff!') else: print(n) countdown_
by_two(n-2)
                             ^
SyntaxError: invalid syntax
```

In [6]:
```
""" There was an error in the code above so I fixed it"""

def countdown_by_two(n):
```

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

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 [12]:
```
a=12
b=3
c=5

def is_triangle(a, b, c):
  """
  Checks if three given integer lengths can form a triangle.

  Args:
    a: The length of the first stick.
    b: The length of the second stick.
    c: The length of the third stick.

  Prints:
    "Yes" if the sticks can form a triangle, "No" otherwise.
  """
  if a > b + c or b > a + c or c > a + b:
    print("No")
  else:
    print("Yes")

# Call the function with the defined variables
is_triangle(a, b, c)
```

No

In [ ]:
```

```

In [ ]:
```

```

In [ ]:

In [ ]: