Homework/Lab 3, DSE5002

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 [2]:
```
!pip install openai dotenv
```

```
Requirement already satisfied: openai in c:\users\water\anaconda3\envs\dse5002\lib\s
ite-packages (1.70.0)
Requirement already satisfied: dotenv in c:\users\water\anaconda3\envs\dse5002\lib\s
ite-packages (0.9.9)
Requirement already satisfied: anyio<5,>=3.5.0 in c:\users\water\anaconda3\envs\dse5
002\lib\site-packages (from openai) (4.6.2)
Requirement already satisfied: distro<2,>=1.7.0 in c:\users\water\anaconda3\envs\dse
5002\lib\site-packages (from openai) (1.9.0)
Requirement already satisfied: httpx<1,>=0.23.0 in c:\users\water\anaconda3\envs\dse
5002\lib\site-packages (from openai) (0.27.0)
Requirement already satisfied: jiter<1,>=0.4.0 in c:\users\water\anaconda3\envs\dse5
002\lib\site-packages (from openai) (0.9.0)
Requirement already satisfied: pydantic<3,>=1.9.0 in c:\users\water\anaconda3\envs\d
se5002\lib\site-packages (from openai) (2.11.1)
Requirement already satisfied: sniffio in c:\users\water\anaconda3\envs\dse5002\lib
\site-packages (from openai) (1.3.0)
Requirement already satisfied: tqdm>4 in c:\users\water\anaconda3\envs\dse5002\lib\s
ite-packages (from openai) (4.67.1)
Requirement already satisfied: typing-extensions<5,>=4.11 in c:\users\water\anaconda
3\envs\dse5002\lib\site-packages (from openai) (4.12.2)
Requirement already satisfied: python-dotenv in c:\users\water\anaconda3\envs\dse500
2\lib\site-packages (from dotenv) (1.1.0)
Requirement already satisfied: idna>=2.8 in c:\users\water\anaconda3\envs\dse5002\li
b\site-packages (from anyio<5,>=3.5.0->openai) (3.7)
Requirement already satisfied: certifi in c:\users\water\anaconda3\envs\dse5002\lib
\site-packages (from httpx<1,>=0.23.0->openai) (2025.1.31)
Requirement already satisfied: httpcore==1.* in c:\users\water\anaconda3\envs\dse500
2\lib\site-packages (from httpx<1,>=0.23.0->openai) (1.0.2)
Requirement already satisfied: h11<0.15,>=0.13 in c:\users\water\anaconda3\envs\dse5
002\lib\site-packages (from httpcore==1.*->httpx<1,>=0.23.0->openai) (0.14.0)
Requirement already satisfied: annotated-types>=0.6.0 in c:\users\water\anaconda3\en
vs\dse5002\lib\site-packages (from pydantic<3,>=1.9.0->openai) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.0 in c:\users\water\anaconda3\env
s\dse5002\lib\site-packages (from pydantic<3,>=1.9.0->openai) (2.33.0)
Requirement already satisfied: typing-inspection>=0.4.0 in c:\users\water\anaconda3
\envs\dse5002\lib\site-packages (from pydantic<3,>=1.9.0->openai) (0.4.0)
Requirement already satisfied: colorama in c:\users\water\anaconda3\envs\dse5002\lib
\site-packages (from tqdm>4->openai) (0.4.6)
```

In [3]:
```python
import os
from dotenv import find_dotenv, dotenv_values

keys = list(dotenv_values(find_dotenv('.env')).items())
OPENAI_API_KEY = os.environ['OPENAI_API_KEY'] = keys[0][1]
```

In [4]:
```python
from openai import OpenAI

client = OpenAI()
client.api_key=OPENAI_API_KEY
```

In [5]:
```python
response = client.chat.completions.create(
    model="gpt-4",
    messages=[{"role" : "user", "content" : "What are some uses of the modulus oper
)
```

In [6]:
```
from IPython.display import display, Markdown
display(Markdown(response.choices[0].message.content))
```

1. Calculating Remainder: The most basic use of the modulus operator is to calculate the remainder of a division operation.

2. Implementing Circular Arrays or Cyclic operations: For example, to implement operations like a clock where after 12 hours, it reverts back to 1, modulus operator can be used.

3. Checking Odd/Even Number: In many programming languages, the modulus operator is frequently used to check if a number is even or odd. If the number modulus 2 equals 0, then it is even; otherwise, it is odd.

4. Implementing Wrap Around connections: In some network algorithms, like in P2P networks, IP addresses or identifiers are represented in a circular way using Modulus.

5. Generating Alternating Sequences: The modulus operator can be used to create alternating sequences. For example, to print an alternating sequence of 0 and 1, we can use the number modulus 2.

6. Calculating Hash Codes: Modulus, in conjunction with other mathematical operations, can be used to calculate hash values or bucket places in hashing related algorithms.

7. Cryptography Algorithms: Modulus operator has high significance in cryptography algorithms like RSA.

8. Implementing Algorithm Checks: An example of this is calculating check digits in ISBN and credit card numbers.

9. Converting between different units of measurement: A common use case involves converting seconds to minutes and remaining seconds.

10. Implementing Pac-Man style screen wraps: When a game character hits one edge of the screen, it reappears on the opposite edge. The character's coordinates can be calculated using a modulus operation.

11. Enforcing limits: If you want a number to remain within a certain range (like 0-255 for color channels in a graphic), the modulus operator can be used.

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 [12]:
```
chained_conditional = client.chat.completions.create(
    model="gpt-4",
    messages=[{"role" : "user", "content" : """ Convert the following python code b

    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')

    """}]
)

display(Markdown(chained_conditional.choices[0].message.content))
```

Sure, a chained conditional format looks slightly different. Here is the chained conditional format of your Python code:

```
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')
```

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 [16]:
```
single_conditional = client.chat.completions.create(
    model="gpt-4",
    messages=[{"role" : "user", "content" : """ Convert the following python code b

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

    """}]
)
```

```
display(Markdown(single_conditional.choices[0].message.content))
```

```
if 0 < x < 10:
    print('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 [17]:
```
recursive = client.chat.completions.create(
    model="gpt-4",
    messages=[{"role" : "user", "content" : """Please ensure the recursive python f

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

    """}]
)

display(Markdown(recursive.choices[0].message.content))
```

The python function `countdown_by_two(n)` appears to be written correctly. It will count down by 2 starting from a given number `n`, and print 'Blastoff!' when it reaches 0. However, it will not handle odd numbers as you might expect. If this function is called with an odd number argument, it will countdown by twos until it gets to 1, and then it would stuck in an infinite recursion, because '1' will never be equal to '0'.

Hence, a slight modification is necessary to make it work properly even with odd numbers:

```
def countdown_by_two(n):
    # If n is less or equal to 0, print 'Blastoff!'
    if n <= 0:
        print('Blastoff!')
    else:
        print(n)
        countdown_by_two(n-2)
```

In the fix, I changed `if n == 0:` to `if n <= 0:`. This way, the function will stop recursing not only when `n` becomes 0 (after calling the function with an even number), but also as soon as `n` becomes a negative number (after calling the function with an odd number). The function will thus behave correctly regardless of whether it is called with an even or odd number.

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 [40]:
```
def is_triangle(side_list:list):
    longest_side = max(side_list)
    side_list.remove(longest_side)
    other_sides_sum = sum(side_list)

    if other_sides_sum==longest_side:
        output = "Degenerate"
    else:
        output = "No" if longest_side > other_sides_sum else "Yes"
```

```
    return output

from random import randrange

for i in range(15):
    side_list = [randrange(0,15,1) for i in range(3)]
    print(f'Sides: {side_list}\tTriangle?: {is_triangle(side_list)}')
```

```
Sides: [13, 2, 14]      Triangle?: Yes
Sides: [3, 10, 7]       Triangle?: Degenerate
Sides: [6, 6, 8]        Triangle?: Yes
Sides: [2, 11, 8]       Triangle?: No
Sides: [6, 9, 11]       Triangle?: Yes
Sides: [14, 2, 13]      Triangle?: Yes
Sides: [6, 14, 1]       Triangle?: No
Sides: [9, 11, 2]       Triangle?: Degenerate
Sides: [5, 5, 1]        Triangle?: Yes
Sides: [12, 13, 5]      Triangle?: Yes
Sides: [11, 4, 2]       Triangle?: No
Sides: [5, 9, 4]        Triangle?: Degenerate
Sides: [6, 0, 9]        Triangle?: No
Sides: [9, 10, 5]       Triangle?: Yes
Sides: [9, 6, 14]       Triangle?: Yes
```

In [ ]: