Making use of a function of a friend.

Objective:

This task is designed to help you to **make the connection between a real-world scenario and the world of coding.**

The code you will write will be *syntactically* correct but is *not designed to work*.

Tips:

- Open this document twice so you can refer to the previous story as you complete the code for the following story.
- Do not use a code editor! Write all of your code in the given code blocks.

Marvin's Mugs

Marvin was the proud owner of a box of nine plain mugs. But there was a problem—Marvin found plain mugs terribly boring. He dreamed of bright colours, fun patterns, and stylish designs. However, Marvin was hopeless with a paintbrush and had no idea how to decorate them himself.

Thankfully, Marvin's friend, **Paula** the Painter, was a true artist. She could transform even the dullest mug into a masterpiece.

Marvin and Paula came up with a plan:

1. Marvin would take each plain mug from the box and hand it to Paula.
2. Paula would work her magic, decorating the mug with beautiful designs.
3. Once the mug was painted to perfection, Paula would hand it back to Marvin, who would carefully place it in a box of completed mugs, admiring the vibrant collection as it grew.

Here's Marvin's plan in code:

```python
# Define a box of mugs
box = [mug1, mug2, mug3, mug4, mug5, mug6, mug7, mug8, mug9]
# Create a box to put the completed mugs into
completed_box = []
# Take each mug in turn
for mug in box:
      # Give a plain mug to Paula and get back the painted version
      painted_mug = paula(mug)
      # Add the painted mug to the completed box
      completed_box.append(painted_mug)
```

By the end of the day, Paula had turned Marvin's boring mugs into a dazzling collection of painted treasures. Each mug was unique and so beautifully decorated that Marvin couldn't help but beam with pride as he placed them into the completed box.

Marvin and Paula had proven once again that great teamwork—and a little bit of creativity—could brighten up even the plainest of mugs.

Workshop of Wonders

Once upon a time, in the bustling Workshop of Wonders, **there lived a pile of six plain** old broom **handles**. They dreamed of the day they'd become proper magical brooms and help sweep the kingdom clean.

In the centre of the workshop was Michelle, the head organiser, and her talented partner, **Charlie** The Constructor, a genius inventor with a knack for transforming broom handles into full-fledged brooms by adding magical brush heads.

Here's how it worked:

1. Michelle would pick up one broom **handle** at a time and hand it to Charlie.
2. **Charlie**, with a flick of his wand (or perhaps just some clever tools), would attach a magical brush head.
3. Once the broom was complete, it was handed back to Michelle, who carefully added it to the growing pile of **finished brooms**, ready to be sent out into the world.

Write code for the story, using the same pattern of code as the previous story to show how Michelle made use of Charlie's help.

Python

```
#Complete this code
broom_handles = [handle1, handle2, handle3
```

Caesar's Love Message

Caesar was madly in love and wanted to send sweet messages to his girlfriend. But there was a problem—he didn't want anyone who might glance upon the messages to understand them. How could he keep his love notes a secret?

Enter Wingman **Dan**, Caesar's clever and loyal friend. Dan came up with a brilliant idea:

"Give me your message one letter at a time," Dan said with a grin, "and I'll shift each letter one step forward in the alphabet. That way, even if someone sees the message, they won't understand it!"

Caesar loved the idea! Here's how their plan worked:

1. Caesar handed each character of his message to Dan.
2. Dan shifted the letter one step forward in the alphabet (e.g., **'a'** became **'b'**, **'b'** became **'c'**, and so on).
3. Caesar took the new letters and stitched them together into a **secret message**, ready to send safely.

Here's how Caesar and Dan's teamwork looked in code:

Python

```
# Define the message
message = "i luv u"
# Create a secret message, ready to accept new characters
secret_msg = ""
# Take each character in the plain text in turn
for character in message:
      # Give the character to Dan and get back the secret character
      secret_character = dan(character)
      # Add the secret character to the secret message
      secret_msg = secret_msg + secret_character
# Show the secret message on the screen
print(secret_msg)
```

By the time they were done, Caesar held a perfectly encoded love note in his hands. His girlfriend would understand the message in an instant, but no prying eyes would have a clue what it meant.

And so, thanks to Wingman Dan's cunning plan, Caesar's love messages remained safe and private. Who knew coding could make you such a great wingman? What secrets will your code keep?

Trafficville

Once upon a time, in the bustling town of Trafficville, a grand traffic survey was conducted. Surveyors carefully observed every vehicle passing by and recorded their findings as a list: Car, Bus, Car, Car, Car, Taxi, Motorbike.

But the surveyors faced a dilemma—this long list was too bulky to send to the King of Trafficville! They needed it condensed into a code that could fit on a single scroll.

Enter **Edith** The Encoder, a brilliant inventor who could transform any vehicle into a single letter: Car became C, Bus became B, Taxi became T, Motorbike became M.

Working alongside Edith The Encoder was Oscar, the town organiser. Together, they devised a plan:

1. Oscar would take each vehicle from the survey one at a time and hand it to Edith.
2. Edith would work her magic and turn the vehicle into a single letter.
3. The letter was returned to Oscar, who carefully added it to a growing string of letters.

Their teamwork was flawless!

Using ideas from the first two stories, write code to show how Oscar made use of Edith's help.
HINT: The survey is stored as a [list]. The output is a scroll that is a "single string".

```python
#Complete this code
survey = [
scroll = "
```

By the end of the day, Oscar and Edith had turned the entire traffic survey into a neat little code. The King of Trafficville was overjoyed—it was short, simple, and exactly what he needed to understand the traffic patterns in his kingdom. And so, thanks to teamwork and a clever bit of coding, Trafficville's roads became smoother, its scrolls lighter, and its people happier.

Writing Functions

Dan the Letter Turner

Dan the Letter Turner loved puzzles but had one big challenge: he needed to shift the letter he was given, but didn't know how to do it. Luckily, he had two brilliant robot helpers: Numeric **Nick** and Alpha **Annie**.

Numeric Nick was amazing at turning letters into numbers (like turning 'A' into 1, 'B' into 2, and so on). Alpha Annie could take a number and turn it back into a letter. Together, they came up with this plan:

1. He'd give a letter to Numeric Nick to get its number.
2. He'd adjust the number by adding one.
3. Finally, he'd give the new number to Alpha Annie to turn it back into a letter.

Now Dan could shift letters with ease, and return each letter to whoever wanted the shifted letter.

Here's how Dan's plan looks in code:

Python

```python
def shift_letter(character):
    # Step 1: Use Numeric Nick to turn the letter into a number
    number = numeric_nick(character)
    # Step 2: Add one to shift to the next number
    shifted_number = number + 1
    # Step 3: Use Alpha Annie to turn the number back into a letter
    new_character = alpha_annie(shifted_number)
    # Return the shifted letter
    return new_character
```

Sam the Sound Maestro

Sam the Sound Maestro had a big problem. She needed to transpose the pitch of notes she was given, but didn't know how to do it.

Thankfully, Sam had two trusty gadgets to help her: **Wavey Wanda** and **Note Norman**.

Wavey Wanda could take a note and turn it into a wave number (a number that represented the note).

Note Norman could take a wave number and turn it back into a new note, before returning it to anyone who wanted a transposed note.

Sam's plan was simple:

1. She'd give a note to Wavey Wanda to get its wave number.
2. She'd adjust the wave number by adding or subtracting the transpose amount.
3. Finally, she'd give the new wave number to Note Normal to turn it back into a note.

Write code to show how Sam took a note from anyone, and how she worked with Wavey Wanda and Note Norman to change any note before giving it back.

Python

```
def adjust_note(note, change):
      #Replace this with your code
```