Functions, Libraries, and Practical Applications

What is a Function?

- A **function** is a named block of code that performs a specific task.
- Functions help organize your code and avoid repetition.
- You give input, it does work, and gives back output.
- **Analogy:** Like a vending machine—put in choice (input), get a snack (output), magic happens inside (hidden).

Defining and Calling Functions

```
def greet():
    print("Hello, robots!")

greet()
```

Parameters

```
def greet(name):
    print(f"Hello, {name}!")

greet("Maria")
greet("Alex")
```

Note: What is an f-string?

An **f-string** lets you easily put variable values into a string in Python. Start your string with `f""`, and put variables inside curly braces {}.

```
name = "Bob"
print(f"Welcome, {name}!") # Prints "
    Welcome, Bob!"
```

Return Values (Example)

```
def f2c(fahrenheit):
    celsius = (fahrenheit - 32) * 5/9
    return celsius

temp_c = f2c(77)
print(temp_c) # Outputs 25.0
```

Avoiding Copy-Paste Code

Bad example:

```
print("Robot moves forward")
print("Robot moves forward")
# ... (10 times)
```

Better with Functions

```
def move_forward():
    print("Robot moves forward")

for i in range(10):
    move_forward()
```

Functions with Parameters

```
def move(direction):
    print(f"Robot moves {direction}")

move("forward")
move("backward")
```

Mini Project Idea

- Write functions like `move_forward()`, `turn_left()`.
- Use loops and timing to simulate a simple robot routine.
- Just print out what the robot should be doing for now.

Useful Libraries

math: extra math functions
time: timing and delays

Example: math Library

```
import math
print(math.sqrt(9))
print(math.sin(math.pi))
```

Example: time Library

```
import time
print("Start")
time.sleep(1)
print("Stop")

# What is time.time()?
current = time.time()
print(current) # Prints the current time in
     seconds since Jan 1, 1970 (the "epoch
    ")
```

Note: `time.time()` returns the current time in seconds as a floating point number. You can use it to:

- Measure elapsed time in your program.
- Build non-blocking loops, timers, or periodic actions for your robot.

Non-blocking Code Example

```
import time
last_check = time.time()
interval = 0.5
while True:
    if time.time() - last_check >= interval
        :
        print("Check sensor")
        last_check = time.time()
    # Other robot tasks here
```

Exercises

- Write a function that prints your name three times.
- Write a function that implements this function: $x^3 + 5x^2 - 7$. Use input to get x from the user.
- Use a loop and a function to print numbers from 1 to 10.
- Make a function that converts Celsius to Fahrenheit.
- Use a non-blocking loop to print "Hello!" every 2 seconds, without using `time.sleep()`.

Further Study Ideas

- Explore more math and time library functions.
- Try writing functions for robot tasks, like turning or reading sensors. Just print output for now.
- Learn how to make your own Python library.
- Look up more on non-blocking code. It is important to understand this concept.

Reflection & Reminders

- Functions = organized, reusable code (like a vending machine).
- Use parameters and return values for flexibility.
- Libraries provide handy tools!
- Non-blocking code keeps your robot responsive.