Python Functions: Arguments, Parameters, and Debugging: Takeaways

Syntax

- Write a single function to generate the frequency tables for any column we want:

```
def freq_table(index):
    frequency_table = {}
    for row in apps_data[1:]:
        value = row[index]
        if value in frequency_table:
            frequency_table[value] += 1
        else:
            frequency_table[value] = 1
    return frequency_table
ratings_ft = freq_table(7)
```

- Define a function with multiple parameters:

```
def add(a, b):
    a_sum = a + b
    return a_sum
print(add(a=9, b=11))
```

- Use named arguments and positional arguments:

```
def subtract(a, b):
    return a - b
print(subtract(a=10, b=7))
print(subtract(b=7, a=10))
print(subtract(10,7))
```

- Reuse functions inside other functions:

```
def find_sum(a_list):
    a_sum = 0
    for element in a_list:
        a_sum += float(element)
    return a_sum

def find_length(a_list):
    length = 0
    for element in a_list:
        length += 1
    return length

def mean(a_list_of_numbers):
    return find_sum(a_list_of_numbers) / find_length(a_list_of_numbers)
```

Concepts

- Python allows us to use multiple parameters for the functions we create.
- We call arguments that we pass by name are called **keyword arguments** (the parameters yield the name). When we use multiple keyword arguments, the order we use doesn't make any practical difference.
- We call arguments that we pass by position **positional arguments**. When we use multiple positional arguments, the order we use matters.
- Positional arguments are often advantageous, because they involve less typing and can speed up our workflow. However, we need to pay extra attention to the order we use to avoid incorrect mappings that can lead to logical errors.
- Reusing functions inside other functions enables us to build complex functions by *abstracting away* function definitions.
- In programming, we call errors bugs. We call the process of fixing an error **debugging**.
- Debugging more complex functions can be more of a challenge, but we can find the **bugs** by reading the **traceback**.

Resources

- Functions in Python