Functions: Fundamentals: Takeaways

Syntax

- Creating a function with a single parameter:

```
def square(number):
    return number**2
```

- Creating a function with more than one parameter:

```
def add(x, y):
    return x + y
```

- Reusing a function within another function's definition:

```
def add_to_square(x):
    return square(x) + 1000  # we defined square() above
```

Concepts

- Generally, a function displays this pattern:
    - It takes in an input.
    - It processes that input.
    - It returns output.
- In Python, we have **built-in functions** like `sum()`, `max()`, `min()`, `len()`, and `print()`, and functions that we create ourselves.
- Structurally, a function contains a header (which contains the `def` statement), a body, and a `return` statement.
- We call input variables **parameters**, and we call the various values that parameters take **arguments**. In `def square(number)`, the `number` variable is a parameter. In `square(number=6)`, the value `6` is an argument that passes to the parameter `number`.
- 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.
- **Debugging** more complex functions can be a bit more challenging, but we can find the **bugs** by reading the **traceback**.

Resources

- Functions in Python