Functions: Fundamentals: Takeaways 🖻

by Dataquest Labs, Inc. - All rights reserved © 2019

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 does something to that input.
 - It gives back an output.
- In Python, we have **built-in functions** like <code>sum()</code> , <code>max()</code> , <code>min()</code> , <code>len()</code> , and <code>print()</code> , and functions that we create ourselves.
- Structurally, a function is composed of a header (which contains the def statement), a body, and a return statement.
- Input variables are called **parameters**, and the various values that parameters take are called **arguments**. In **def square(number)**, the **number** variable is a parameter. In **square(number=6)**, the value **6** is an argument that is passed to the parameter **number**.

- Arguments that are passed by name are called **keyword arguments** (the parameters give the name). When we use multiple keyword arguments, the order we use doesn't make any practical difference.
- Arguments that are passed by position are called **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



Takeaways by Dataquest Labs, Inc. - All rights reserved © 2019