

The background is a dark chalkboard with various light-colored chalk sketches. On the left, there's a large 'V' and a telescope. In the top left, a globe is sketched. At the bottom, there are sketches of a book, a plus sign, a percentage sign, and a less-than sign.

Functions In Python

Creating and Using Functions

Functions in Python

- Functions are blocks of code that are given a name so we can use them over and over again.

```
def say_hello():  
    print("Hello! I'm a function!")
```

- Functions start with keyword def, then the name, a pair of parenthesis (), and a colon
- Code inside a function must be indented

Functions in Python

```
def say_hello():  
    print("Hello! I'm a function!")
```

To call a function like the one above from within the same module you wrote the function, simply type the name of the function – including the ()

```
say_hello()  
say_hello()  
say_hello()
```

Output:

```
Hello! I'm a function!  
Hello! I'm a function!  
Hello! I'm a function!
```

Functions in Python

To call a function from a different module we must import the function.

```
writing_functions.py ×  
1 def say_hello():  
2     print("Hello! I'm a function!")
```

Here the say_hello function has been written in module named writing_functions

```
using_functions.py ×  
1 from writing_functions import say_hello  
2 say_hello()  
3 say_hello()  
4 say_hello()
```

As long as the module using_functions is saved in the same directory as writing_functions, we can import and use the say_hello function.

Functions in Python

To call a function from a different module we must import the function.

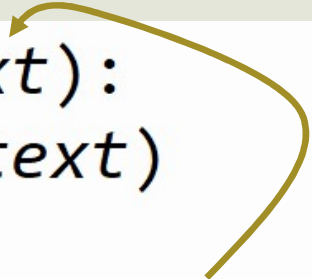
```
writing_functions.py ×  
1 def say_hello():  
2     print("Hello! I'm a function!")
```

Instead of importing specific functions we can import an entire module. But we have to say the module name 'dot' before the function each time.

```
import writing_functions  
writing_functions.say_hello()  
writing_functions.say_hello()  
writing_functions.say_hello()
```

Functions in Python with Parameters

```
def say_three_times(text):  
    print(text, text, text)
```



```
say_three_times('Hello CS Students!')  
say_three_times('Hello CS Teacher!')
```

- Functions can use parameters to allow customizing how they behave!
- The variables in the () of the function def block are called parameters.
- The values send are called arguments.
- The parameters are like parking spaces waiting to be filled with values of the arguments.

Functions in Python using the return keyword

- Functions often make use of the *return* keyword.
- *return* will send the information back to the spot that called the function in the program.
- The value returned must either be stored in a variable or printed immediately.

```
def triple_it(x):  
    return x * 3  
  
result = triple_it(6.2)  
print(result)  
print(triple_it(5))
```

Output:

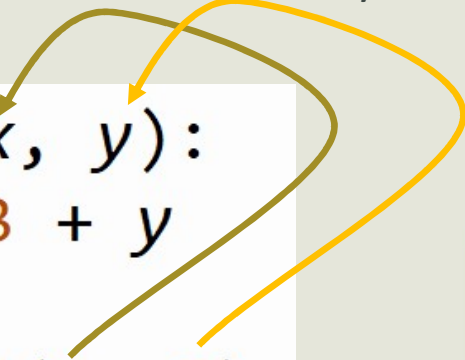
18.6

15

Functions in Python using Multiple Parameters

- Functions can have as many parameters as necessary to get the work done.
- Arguments fill the parameters in the order they are listed in the call statement.

```
def triple_add(x, y):  
    return x * 3 + y  
  
ans = triple_add(5, 2)  
print(ans)
```

A diagram with two yellow arrows and one brown arrow. One yellow arrow points from the '5' in the function call to the 'x' parameter in the function definition. Another yellow arrow points from the '2' in the function call to the 'y' parameter in the function definition. A brown arrow points from the function call line to the return value '17' in the output box.

Output:
17