FUNCTIONAL DECOMPOSITION

An Introduction to Computer Science

Let's learn about Functional Decomposition

Functional Decomposition

Going to the store		
Get to car	Drive car to store	Exit car into store

When writing out a solution, you can use a powerful tool called Functional Decomposition. The idea is to break a complex problem into easier parts.

Consider the task of going to the store.

You could break this long algorithm into three shorter subtasks, and focus on solving each in turn.

Split Up Complex Functions!

```python
def is_user_input_freezing(text):
    celsius = int(text[-2:])
    fahrenheit = (celsius*9/5) + 32
    return fahrenheit < 32
```

```python
def capture_temperature(text):
    return int(text[-2:])

def to_fahrenheit(celsius):
    return (celsius*8/5) + 32

def is_freezing(temperature):
    return temperature < 32
```

```python
def is_user_input_freezing(text):
    celsius = capture_temperature(text)
    fahrenheit = to_fahrenheit(celsius)
    return is_freezing(fahrenheit)
```

Let's take a concrete example.
On the left, we see some code that needs to do a series of tasks.
On the right, we have split that one function into three separate pieces, each of which could be tested independently.
Then, we rewrite the original function to use our new functions.

3

Functional Decomposition is Powerful!

```python
from cisc108 import assert_equal

def to_fahrenheit(celsius):
    return (celsius*8/5) + 32

assert_equal(to_fahrenheit(40), 104.0)
```

FAILURE, predicted answer was 104, computed answer was 96.

Ah, so this part of the function is the problem!

It may seem like Functional Decomposition is just giving yourself more work.
Now you have to unit test and document extra functions?
Why not just do all the work in a single function?
The answer is that decomposition makes it easier to debug functions.
As our programs get more complex, you are more likely to make mistakes.
Breaking your functions into manageable parts is scientifically shown to increase your likelihood of success.