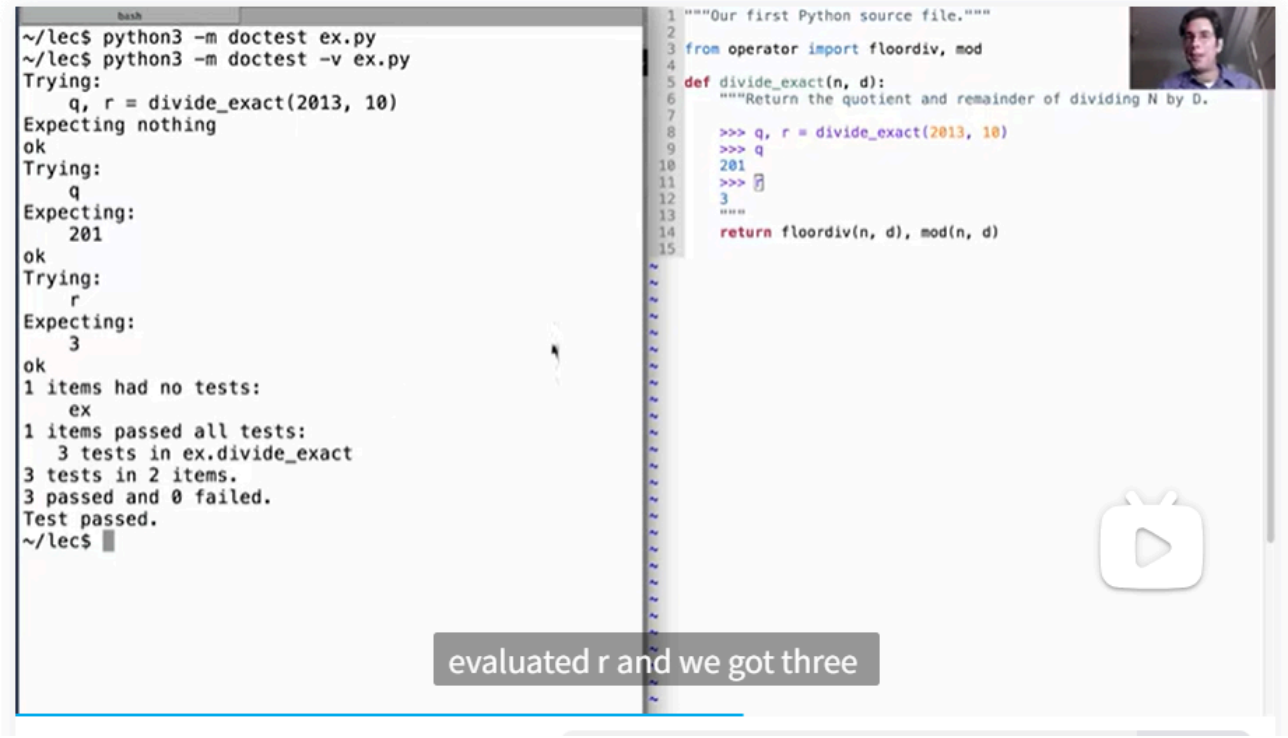
Lecture 3.control

RETURN & EFFECTS

miscellaneous python features

-
    -
        -
            - /(true div)
- //(floor dive)
- %(mod)
- return multiple values from function
    - return n // d, n % d
- docstring
    - documentation about what a function does
- besides describing what it does , we could also add some dottest



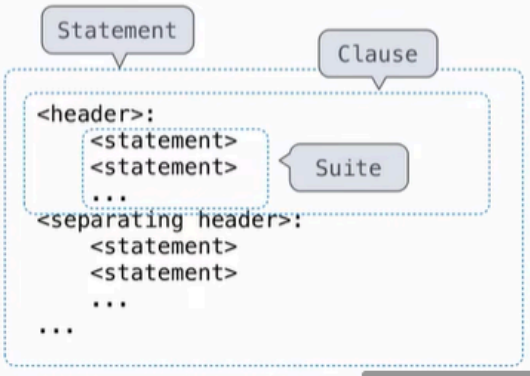
- default values of parameters
    - already learnt, so no more explaination.

Conditional Statements

Statements

A *statement* is executed by the interpreter to perform an action

Compound statements:



The first header determines a statement's type

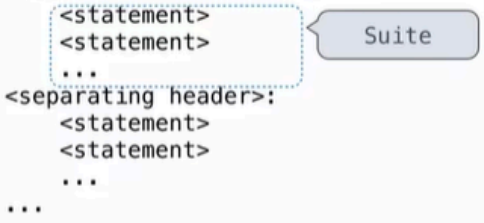
The header of a clause "controls" the suite that follows

def statements are compound statements

deaf statements as we saw

Compound Statements

Compound statements:



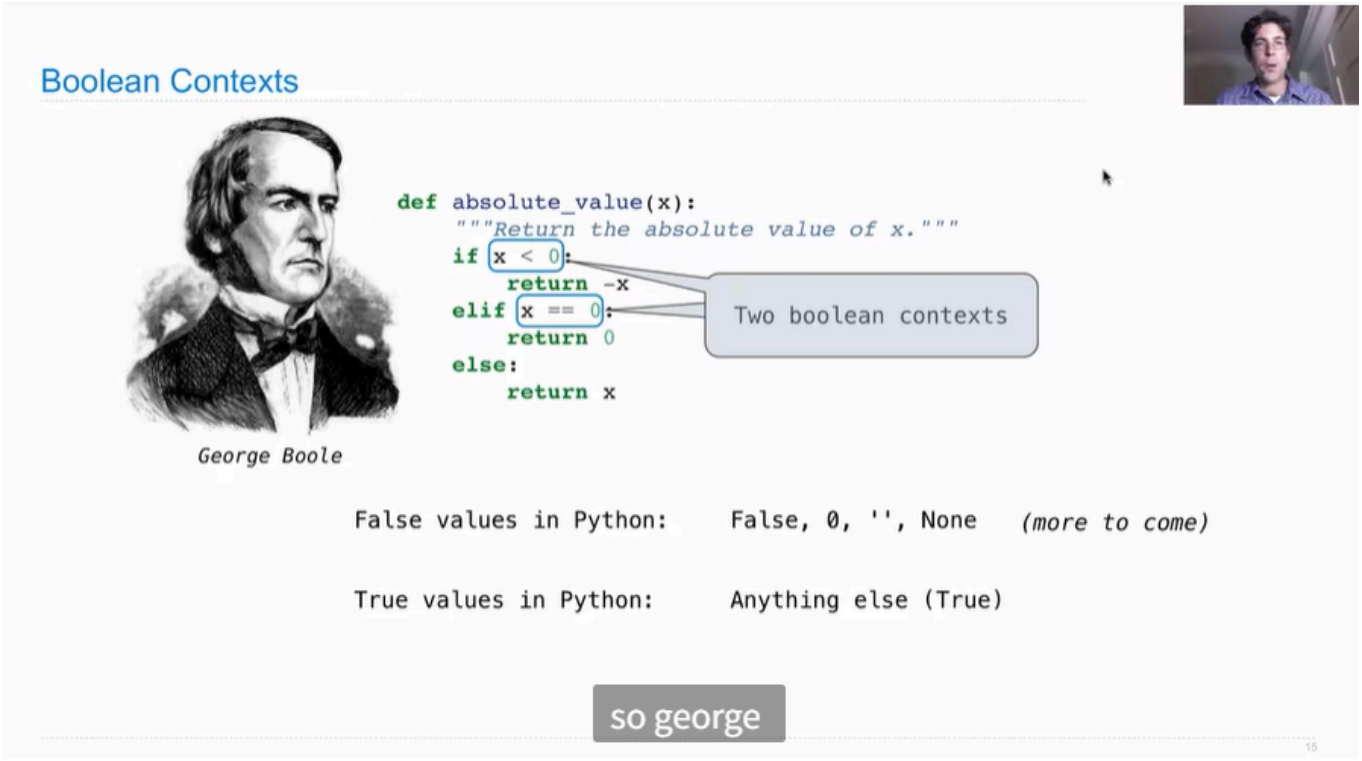
A suite is a sequence of statements

To "execute" a suite means to execute its sequence of statements, in order

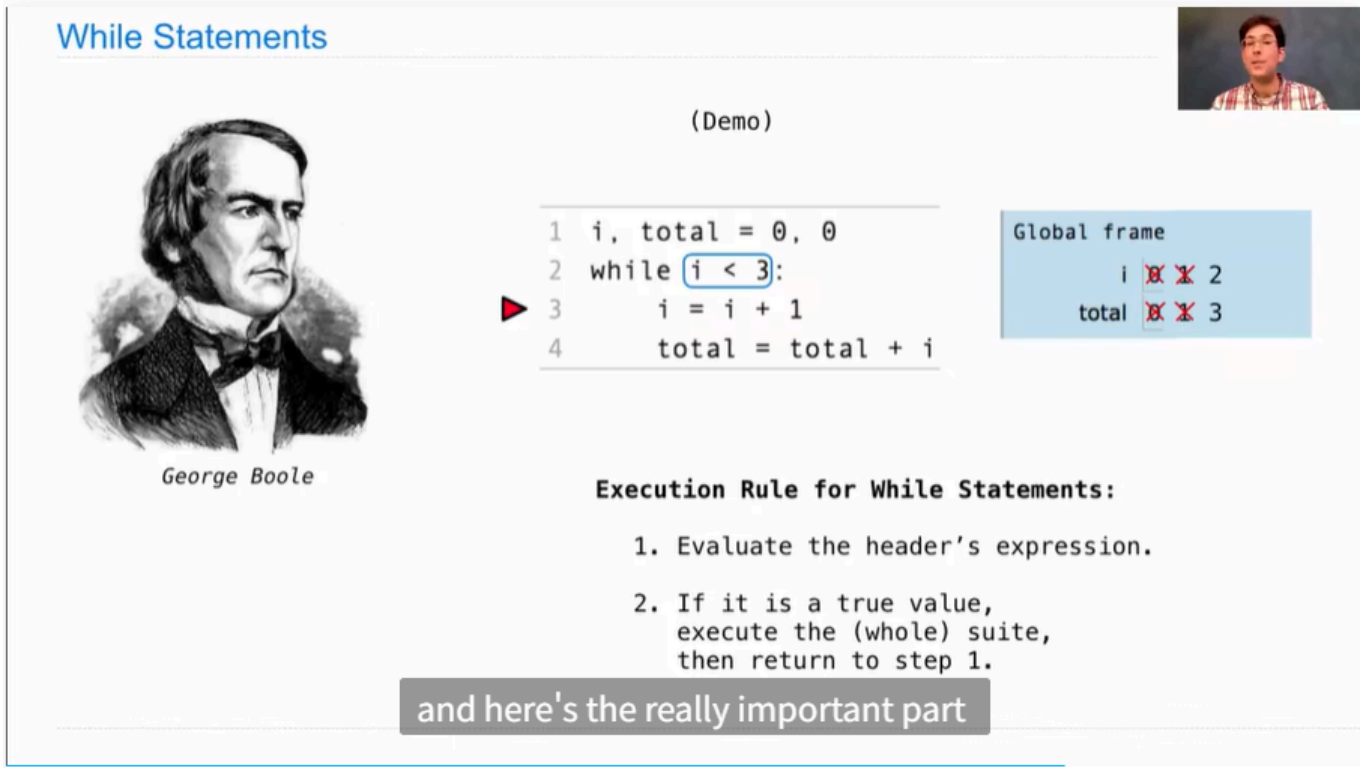
Execution Rule for a sequence of statements:

- Execute the first statement
- Unless directed otherwise, execute the rest

it's execute the first statement



iteration



example: find the prime factorization

```
    """Print the prime factors of n in non-decreasing order.

    >>> prime_factors(8)
    2
    2
    2
    >>> prime_factors(9)
    3
    3
    >>> prime_factors(10)
    2
    5
    >>> prime_factors(11)
    11
    >>> prime_factors(12)
    2
    2
    3
    >>> prime_factors(858)
    2
    3
    11
    13
    """
    while n > 1:
        k = smallest_prime_factor(n)
        n = n // k
        print(k)

def smallest_prime_factor(n):
    """Return the smallest k > 1 that evenly divides n."""
    k = 2
    while n % k != 0:
        k = k + 1
    return k
```

> what we learn from this code?
> at first, think of the problems in the simpliest way , ignore all the complex structures, use the simpliest sentences to organize your main document.
> then, write the structures of the functions you used in your main document, but in the same way, so in this way, you don't need to organize your program at the very begining. in stead, you solve it in the middle.

> more things, try to make your names easy to understand, like later prof changed k into smallest_prime, this is not necessary, but it makes your program a better one, at least easier to be interpreted by human.

> this is a way of abstraction when we are thinking about things, however, after we finished our program, we might realize that we don't need the functions, we might think we don't need it, cause obviously we can just fit it into the main document. however, when we fit it together, it does seems shorter, but more unlikely to be understand straightly.

```
but often using functions and their names makes everything clearer than trying
to fit everthing into a long body of a program.
```