Pt 1

For loops have no control, data flows automatically

Pt 2

Because its an iterator, data only flows when next() is called

Iterator remembers its place so the next call will return the next value

Pt 3

Yield pauses the function, doesn't destroy it like return

Remembers its place like next()

Pt 4

Return destroys stack frame and therefore function

Pt 5

x does not reset

Pt 6

next() and yield together allows for precise control

Data will keep flowing from one next() call until yield succeeds the check

Pt 7

Dunder methods already exist in Python and run in the background

Defining them allows programmers to adjust functionality for specific purpose

Pt 8

All methods with "__name__" formatting are dunder methods and happen automatically in the background

1.1

x is stored in the heap because its stored in the function object

x still exists because its in the heap, which continues to exist after stack frame is destroyed

Execution doesn't reset because the stack frame is destroyed, but Python remembers the data stored in x

1.2

count is preserved in heap so even after a new stack is created, Python remembers the value

It's still in the heap

1.3

The stack frame is gone. The value is being stored inside the function object

2.1

my_decorator runs first

greet() was not modified, the decorator features were added onto it

my_decorator added the behavior

2.2

when the function is defined as a decorator, it runs, showing that the wrapper doesn't run immediately when the decorator is called

the wrapper is modification of the function, so the function runs the decorator every time its called

the modifications often change how the function runs in some way, which means the wrapper should call the original function when it is needed

2.3

the wrapper overrides the original function, so it now refers to the wrapper class
the original function is there so the wrapper can access the original functionality

3.1

logging is added once the original function is called
no, the decorator only adds on to the function
we don't know how many numbers we need to pass to the function so we use args

4.1

stores data, no wrapper, no behavior interception

4.2

extra behavior added, wrapper layer present, original function untouched