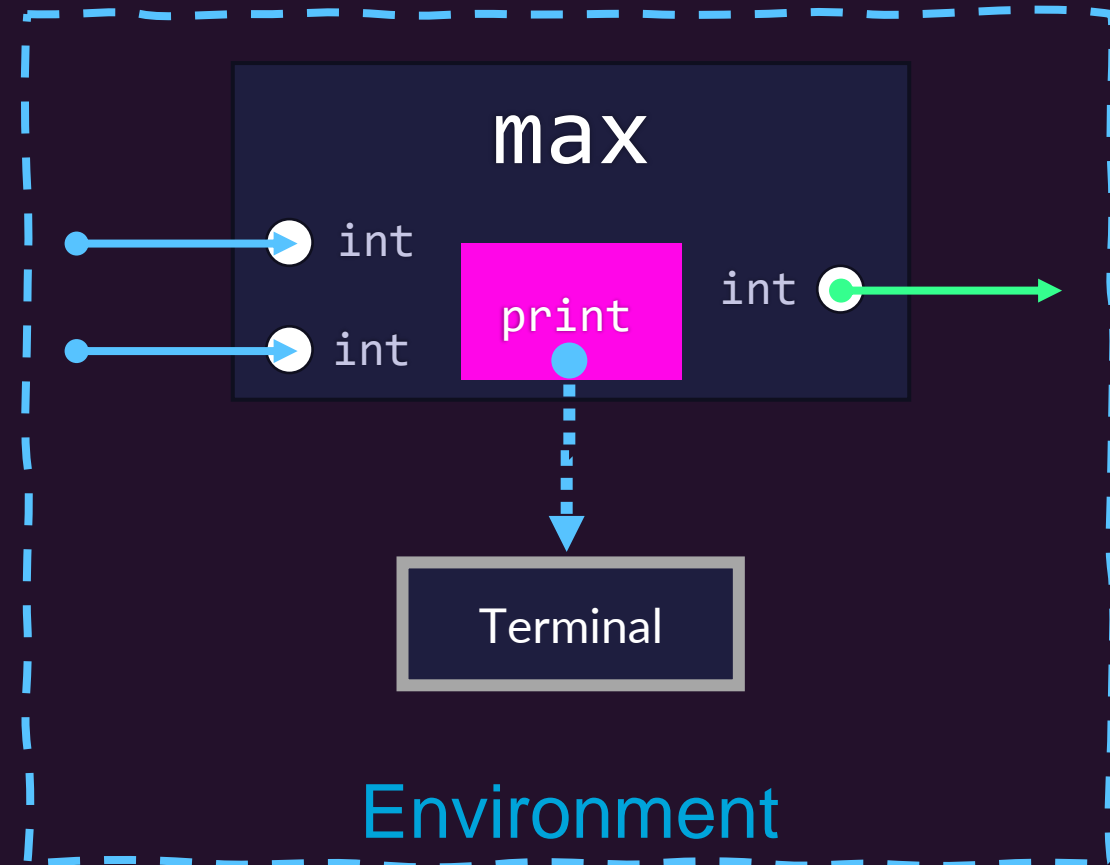


return  
vs print!

# Return vs. "Side-effects"

- A function definition has a list of parameter "inputs" and a single returned "result".
- What happens if you call *print* from within a function definition?
  - Consider the modified max function.
  - This breaks our model!
- Our *Fundamental Pattern* also has an *Environment*
  - An algorithm may have "side-effects" on its "environment"
- The terminal your program is running in is part of its environment
- When you call the built-in print function, it has a "side-effect" that produces output in the terminal

```
def max(a: int, b: int) -> int:  
    """Return the largest of two numbers."""  
    print(str(a) + ", " + str(b))  
    if a > b:  
        return a  
    else:  
        return b
```



# The **return** Statement vs. Printing

- **The return statement is *for the computer* to use as the evaluated result of the originating function call expression**
  - A bookmark is dropped when you *call* a function. When that function's body reaches a *return statement*, the returned value is "substituted" for the function call expression and the program continues.
- **Printing is for getting information out of the program.**
  - Typically, it is *for humans to see*.
  - To present data to the user of a program you must *output* it in some way.
- **With a function that returns a value and has no other side-effects, you can print the value it returns by:**
  1. Printing the function call expression directly, such as `print(a_func())`, or
  2. By storing its return value in a variable and later printing the variable.

# Procedures are functions that return *nothing*...

- ...their whole purpose is to produce a side-effect on the environment.
- The purpose of procedures is to produce side-effects such as printing output.
  - The `print` function is a procedure!
- Use a *return type* of `None` to declare a procedure.

```
def print_repeatedly(line: str, repeats: int) -> None:  
    """Print a line a specific number of repetitions."""  
    i: int = 0  
    while i < repeats:  
        print(line)  
        i = i + 1
```

```
print_repeatedly("hello", 5)  
print_repeatedly("world", 5)
```