[220 / 319] Using Functions

Meena Syamkumar
Andy Kuemmel

Please read Ch 3
of Think Python

Learning Objectives Today

How to call functions

- input/output
- terminology: call / invoke, parameter, argument, return value
- control flow

Function usage examples

- input(),
- type cast functions: int(), bool(), float(), str()

Using functions from built-in or user-created module:

- keywords: import, as
- attribute operator: "."
- help: inspect a module

make a battleship game!

we'll learn about how to give functions input by passing arguments (e.g., 2) to parameters (e.g., moves)

Main Code:

1. Put 2 in the "moves" box
2. Perform the steps under "Move Code", then continue to step 3
3. Rotate the robot 90 degrees to the right (so arrow points to right)
4. Put 3 in the "moves" box
5. Perform the steps under "Move Code", then continue to step 6
6. Whatever symbol the robot is sitting on, write that symbol in the "resut" box

today we'll learn how to use functions in Python

we'll also learn how to ask functions questions and get answers called return values

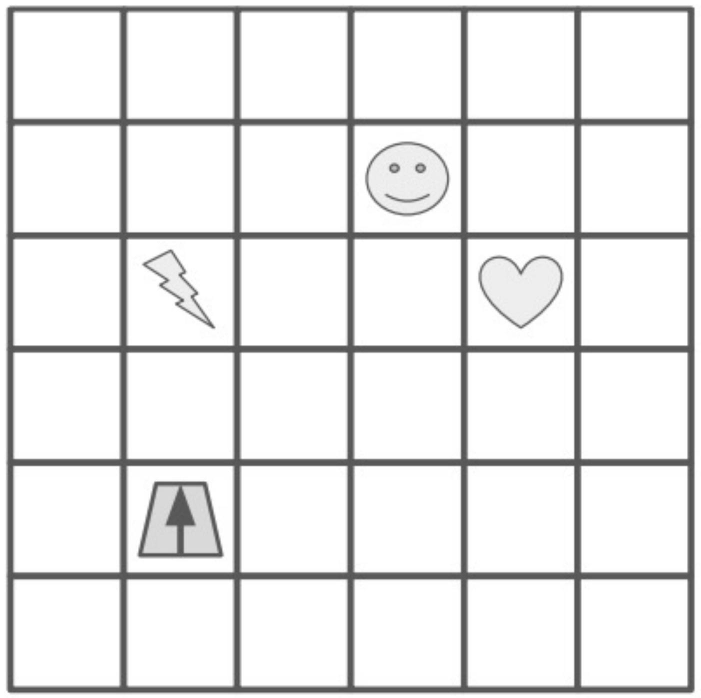
Move Code:

A. If "moves" is 0, stop performing these steps in "Move Code", and go back to where you last were in "Main Code" to complete more steps
B. Move the robot forward one square, in the direction the arrow is pointing
C. Decrease the value in "moves" by one
D. Go back to step A

"Move Code" is a function

next lecture, we'll learn how to write our own new functions

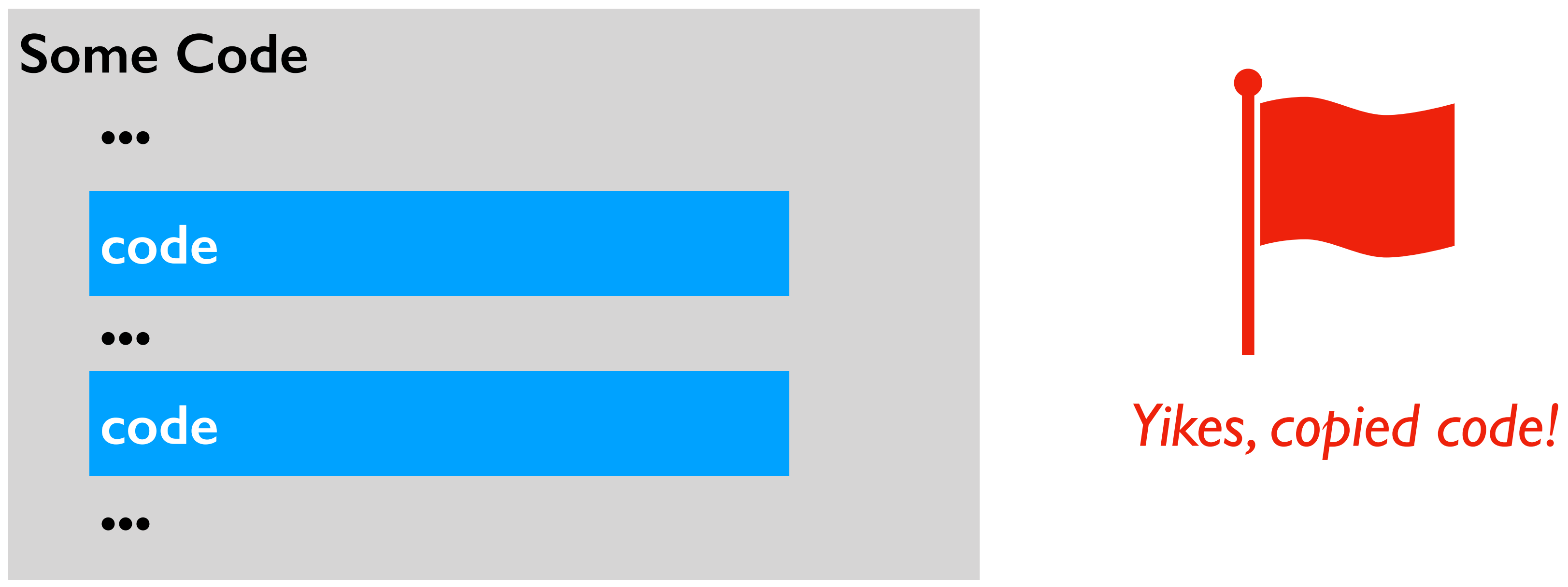
Functions are like "mini programs", as in our robot worksheet problem



Vocabulary

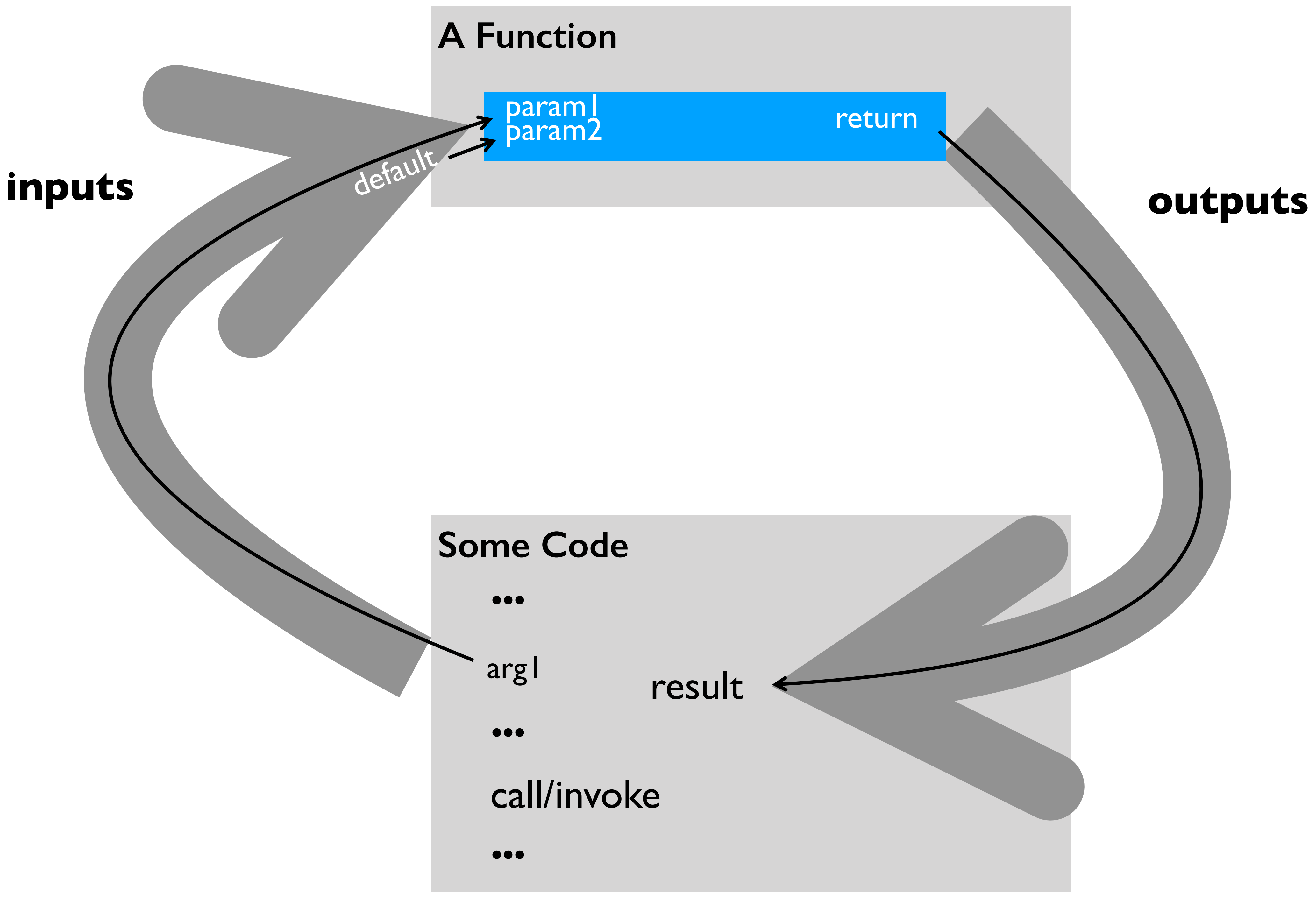
- ...

General Function Concepts



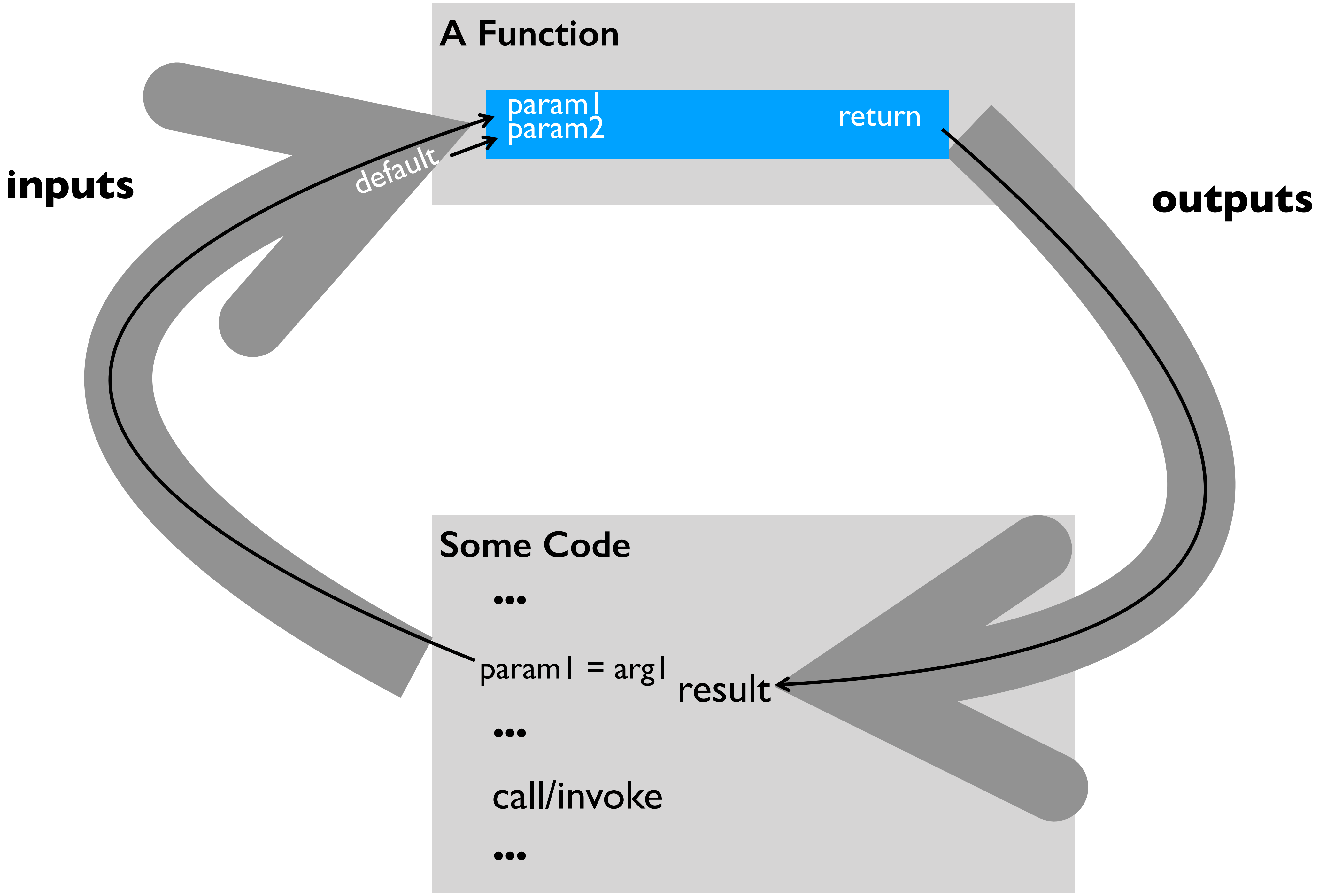
Vocabulary

- **refactor**: change organization of code (e.g., to avoid repetition)
- **parameter**: variable that receives input to function
- **argument**: value sent to a function (lines up with parameter)
- **return value (or result)**: function output sent back to calling code
- **default argument**: value put in parameter if argument not passed



Vocabulary

- **refactor**: change organization of code (e.g., to avoid repetition)
- **parameter**: variable that receives input to function
- **argument**: value sent to a function (lines up with parameter)
- **return value (or result)**: function output sent back to calling code
- **default argument**: value put in parameter if argument not passed
- **named/keyword argument**: argument explicitly tied to a parameter



Calling/Invoking a Function in Python

```
print(“hello”)
result = f(x)
```

return value

ALWAYS: function’s name

ALWAYS: followed by parentheses

SOMETIMES: with one or more arguments

SOMETIMES: producing a result

Calling/Invoking a Function in Python

```
print(“hello”, “world”)
x = input()
```

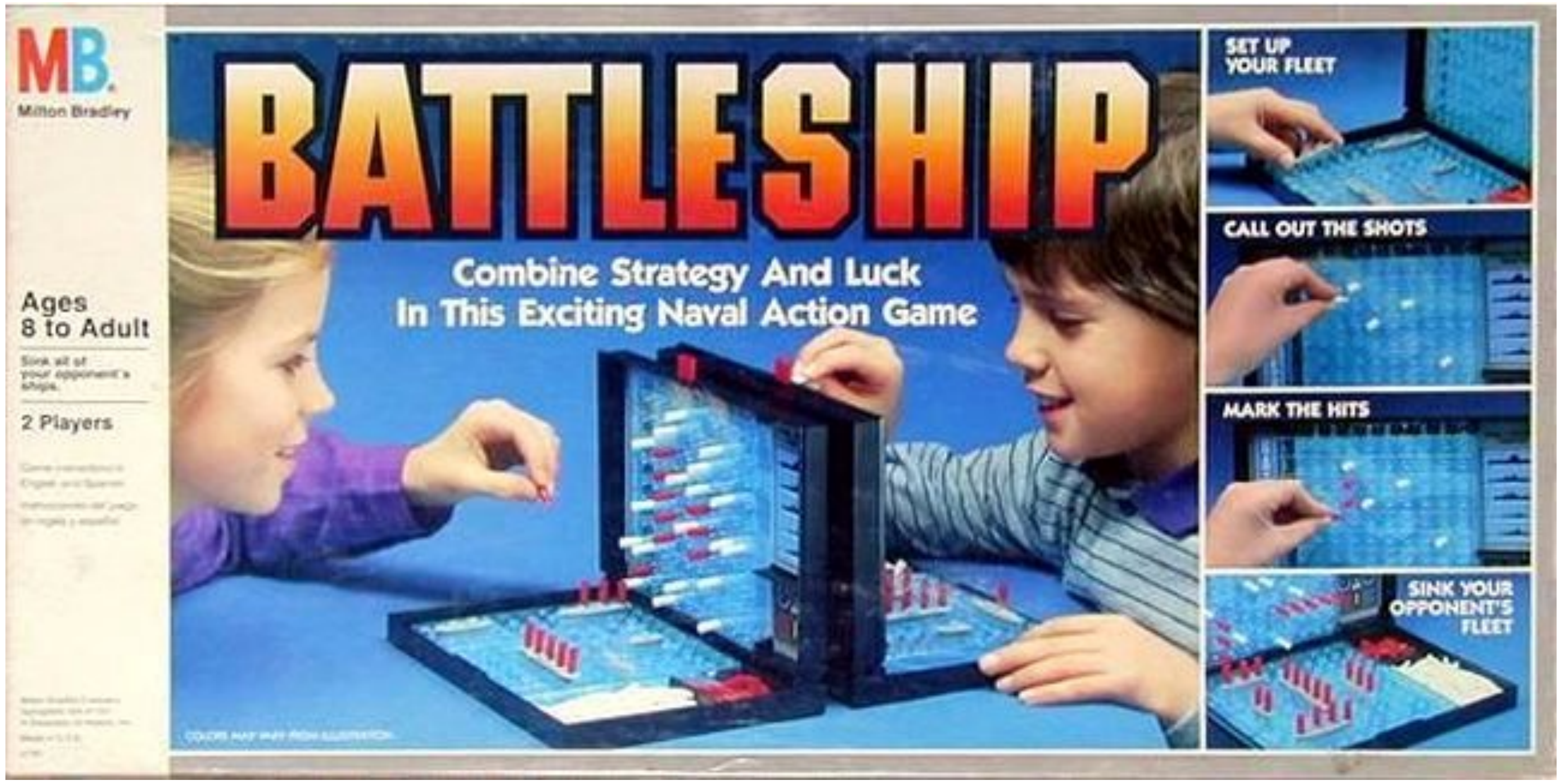
ALWAYS: function’s name

ALWAYS: followed by parentheses

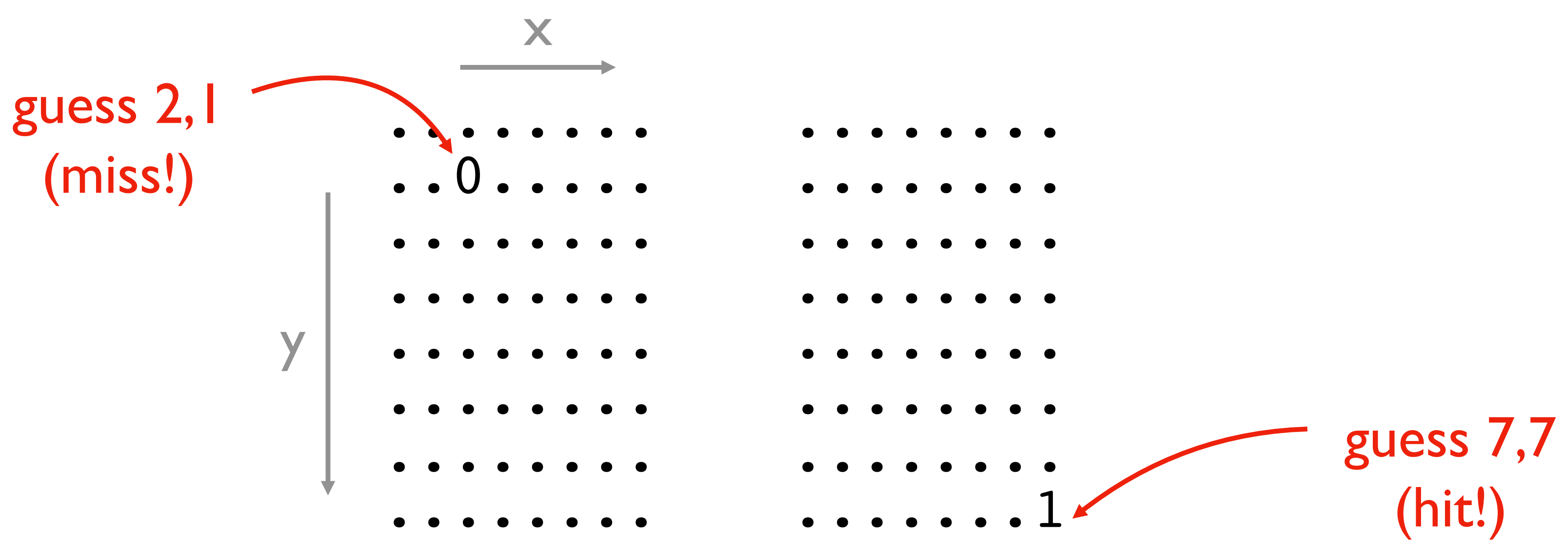
SOMETIMES: with one or more arguments

SOMETIMES: producing a result

Example: Battleship Demo (Version 1)



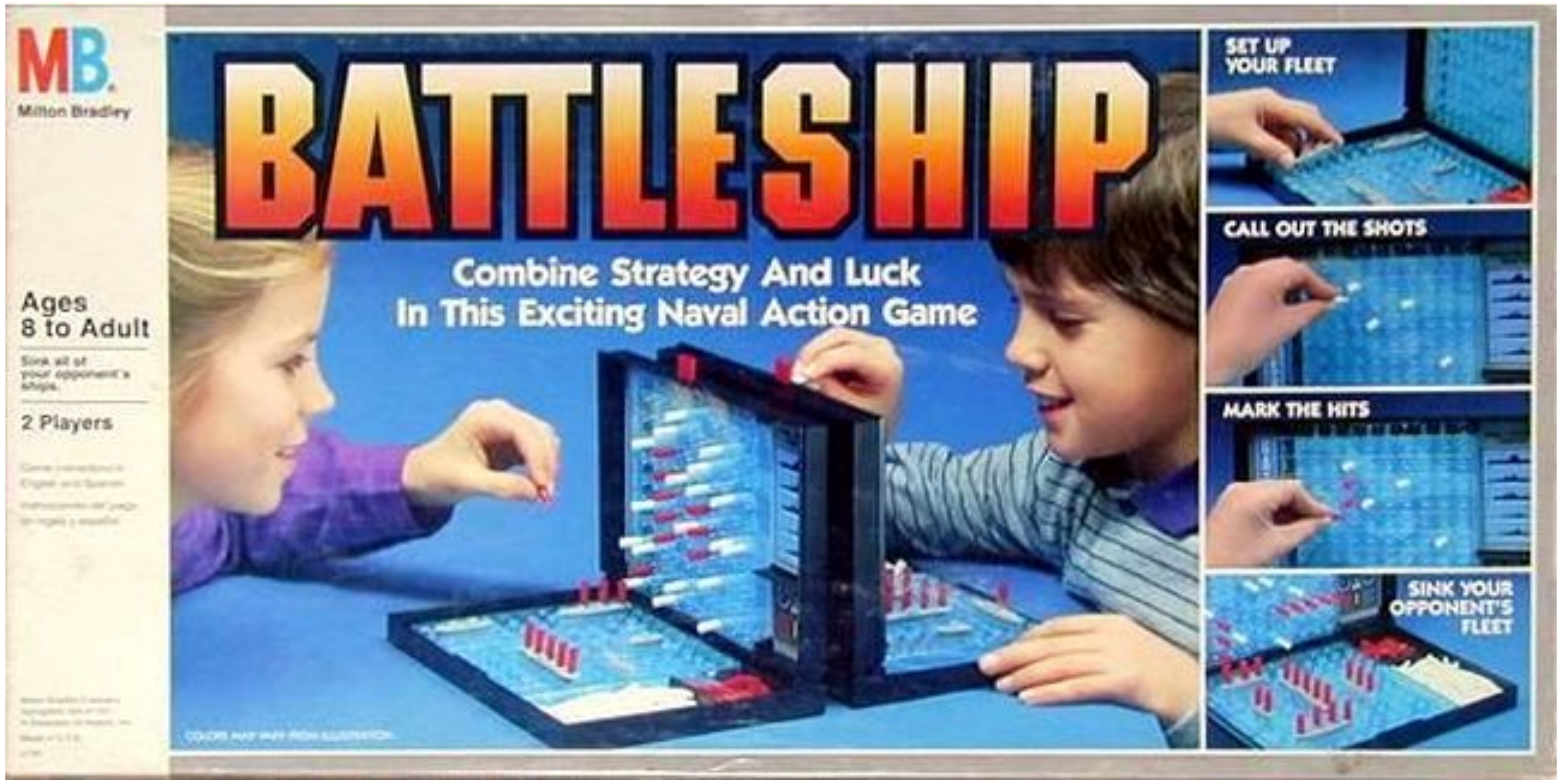
https://boardgamegeek.com/image/288374/battleship



Version 1 (MVP)

- 1 ship, 1 guess
- ship is 1 space
- fixed position
- top/left is 0,0
- horrible graphics

Practice: Battleship Demo (Version 2)



https://boardgamegeek.com/image/288374/battleship

Version 1 (MVP)
- 1 ship, 1 guess
- ship is 1 space
- fixed position
- top/left is 0,0
- horrible graphics

Version 2
- larger ship
- multiple ships
- random locations

guess 2,1
(miss!)

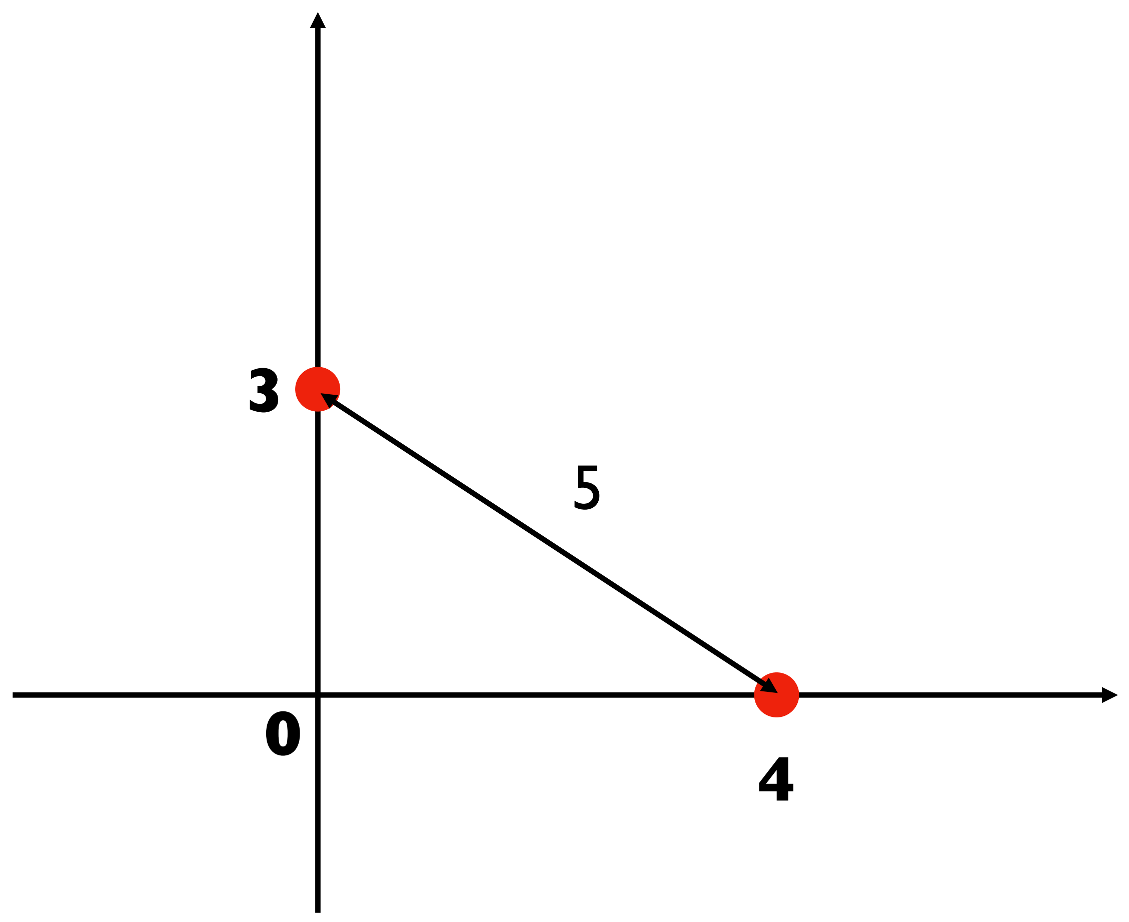
```
. . . . . . . .
. . 0 . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
```

```
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . 1
```

guess 7,7
(hit!)

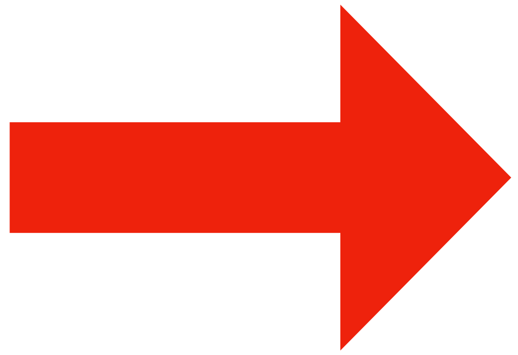
time permitting

Challenge: Polar Coords Distance



point 1: distance 3 at angle 90°

point 2: distance 4 at angle 0°



distance: 5