

Important! There are three ways for values to get into **parameters**:

1. positional **argument**
2. keyword/named **argument**
3. default **argument**

For these puzzles, the best way to practice is to write down what you think, then actually run the code in Python (in PythonTutor!) to check your mental model. Try thinking of your own examples to try too (this is active learning).

```
def add(x, y):
    print(x) # for debugging
    print(y) # for debugging
    return x+y
```

How many parameters? Two (x, y)

Do the parameters have default values? No

What will be in the parameters for each call? [some crash; mark them]

Call	Crashed?	Parameters	Return value?
1. add(3)	✓	x=3, y=?	
2. add(3, 4)	NO	x=3, y=4	7
3. add(4, 3)	NO	x=4, y=3	7
4. add(3, 4, 5)	✓	x=3, y=4	
5. add(6, y=7)	NO	x=6, y=7	13
6. add(7, x=6)	✓	x=7, y=?	
7. add(7, 8, x=9)	✓	x=7, y=8	
8. add(x=10, y=11)	NO	x=10, y=11	21
9. add(y=10, x=11)	NO	x=11, y=10	21

How many arguments? 1, 2, 3, 2, 2, 3, 2, 2, 2

3rd argument (only 2 parameters)

```
def say(word, t=1, end=""):
    # add prints here to debug
    print(word * t + end)
```

How many parameters? Three (word, t, end)

How many arguments?

Call	Crashed?	Parameters	Output
10. say()	✓	word=?, t=1, end=""	
11. say("ha")	NO	word="ha", t=1, end=""	ha
12. say("ha", t=2)	NO	word="ha", t=2, end=""	haha
13. say("ha", end="!")	NO	word="ha", t=1, end="!"	ha!
14. say(t=2, word="yo")	NO	word="yo", t=2, end=""	yoyo
15. say("W", 3, "eb")	NO	word="W", t=3, end="eb"	WWWeb
16. say(t=3)	✓	word=?, t=3, end=""	
17. say("W", end=".", 3)	✓	word=?, t=?, end="."	
18. say("huh", 1, "?", "!!")	✓	word="huh", t=1, end="?"	

Reasons for crash:

1. 2 arguments for x
2. no argument for y

Reason: positional argument (3) passed after keyword argument (end=".")

How many parameters have default values? Two (t, end)

4th argument (only 3 parameters)

19. What letters are printed, and in what order?

```
print("A")
```

```
def foo():
```

```
    print("B")
```

```
print("C")
```

```
foo() → FUNCTION CALL
```

```
print("D")
```

```
foo() → FUNCTION CALL
```

OUTPUT :

A

C

B

D

B

20. Repeat question 19, but imagine a tab before print("C")

21. What is printed, and in what order?

```
def func_c():
```

```
    print("C")
```

```
def func_b():
```

```
    print("B1")
```

```
    func_c()
```

```
    print("B2")
```

```
def func_a():
```

```
    print("A1")
```

```
    func_b()
```

```
    print("A2")
```

```
func_a()
```

OUTPUT :

A1

B1

C

B2

A2

22. What is printed, and in what order?

```
def f():
```

```
    print("A")
```

```
    return("B")
```

```
    print("C") → this line will
```

```
print("D")
```

```
x = f()
```

```
print("E")
```

```
print(x)
```

OUTPUT :

D

A

E

B

never get
executed, because

return statement will always be the final line of
code executed in a function call.