

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

How many arguments?

1. add(3)
2. add(3, 4)
3. add(4, 3) *3rd argument*
4. add(3, 4, 5) *(only 2 parameters)*
5. add(6, y=7)
6. add(7, x=6)
7. add(7, 8, x=9)
8. add(x=10, y=11)
9. add(y=10, x=11)

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

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

How many arguments?

10. say()
11. say("ha")
12. say("ha", t=2)
13. say("ha", end="!")
14. say(t=2, word="yo")
15. say("W", 3, "eb")
16. say(t=3)
17. say("W", end=".", 3)
18. say("huh", 1, "?", "!")

How many parameters have default values?

Two (t, end)

- crashed? ✓ OR x= 3, y= ?
crashed? NO OR x= 3, y= 4 → 7
crashed? NO OR x= 4, y= 3 → 7
crashed? ✓ OR x= 3, y= 4
crashed? NO OR x= 6, y= 7 → 13
crashed? ✓ OR x= 7, y= ?
crashed? ✓ OR x= 7, y= 8
crashed? NO OR x= 10, y= 11 → 21
crashed? NO OR x= 11, y= 10 → 21

return value?

6. REASONS FOR CRASH:

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

OUTPUT:

- crashed? ✓ OR word= ?, t= ?, end=?
crashed? NO OR word= "ha", t= 1, end= "?" ha
crashed? NO OR word= "ha", t= 2, end= "" haha
crashed? NO OR word= "ha", t= 1, end= "!" ha!
crashed? NO OR word= "yo", t= 2, end= "" yoyo
crashed? NO OR word= "W", t= 3, end= "eb" WwWeb
crashed? ✓ OR word= ?, t= 3, end= ""
crashed? ✓ OR word= ___, t= ___, end= ___.
crashed? ✓ OR word= "huh", t= 1, end= "?"

1 of 2

REASON: positional argument (3) passed after keyword argument (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.