

```
$ whatsafunction.sh
```

Intro to Programming

```
// Year 10s
```

— Dr. Hughes

Instructions to make tea



- 1 Step 1: boil water
- 2 Step 2: get cup
- 3 Step 3: get a teabag
- 4 Step 4: put teabag in cup
- 5 Step 5: add boiled water to cup
- 6 Step 6: wait 1 minute
- 7 Step 7: use spoon to remove teabag

NORMAL

following 7
steps...

// seven steps. TeaPT is going to follow these, in order, every time.

How to make four cups of tea



BUSY
repeat after
me...

Cup 1

- 1 Step 1: boil water
- 2 Step 2: get cup
- 3 Step 3: get a teabag
- 4 Step 4: put teabag in cup
- 5 Step 5: add boiled water to cup
- 6 Step 6: wait 1 minute
- 7 Step 7: use spoon to remove teabag

Cup 2

- 1 Step 1: boil water
- 2 Step 2: get cup
- 3 Step 3: get a teabag
- 4 Step 4: put teabag in cup
- 5 Step 5: add boiled water to cup
- 6 Step 6: wait 1 minute
- 7 Step 7: use spoon to remove teabag

Cup 3

- 1 Step 1: boil water
- 2 Step 2: get cup
- 3 Step 3: get a teabag
- 4 Step 4: put teabag in cup
- 5 Step 5: add boiled water to cup
- 6 Step 6: wait 1 minute
- 7 Step 7: use spoon to remove teabag

Cup 4

- 1 Step 1: boil water
- 2 Step 2: get cup
- 3 Step 3: get a teabag
- 4 Step 4: put teabag in cup
- 5 Step 5: add boiled water to cup
- 6 Step 6: wait 1 minute
- 7 Step 7: use spoon to remove teabag

// 28 lines. Four cups, one recipe, four copies.

Why can't Ctrl+C, Ctrl+V do all the work?



CONFUSED

uh... what?

```
1 Step 1: boil water
2 Step 2: get cup
3 Step 3: get a teabag
4 Step 4: put teabag in cup
5 Step 5: add boiled water to cup
6 Step 6: wait 1 minute
7 Step 7: use spoon to remove teabag copy 1 – pasted
8 Step 8: wait 1 minute
9 Step 9: use spoon to remove teabag copy 2 – pasted
10 Step 10: wait 1 minute
11 Step 11: use spoon to remove teabag copy 3 – pasted
12 Step 12: wait 1 minute
13 Step 13: use spoon to remove teabag copy 4 – pasted
14 Step 14: wait 1 minute
15 Step 15: use spoon to remove teabag copy 5 – pasted
```

Spot the problem?

// copy-paste struck again: the same 2 lines, pasted 5 times over. recipe again.

It's fine, we'll just be careful



OVERWHELMED
four drinks?

Customer 1



"Tea, please"

Customer 2



"Tea with
milk, please"

Customer 3



"Tea with **oat**
milk, please"

Customer 4



"**Coffee**,
please"

- 1 Step 1: boil water
- 2 Step 2: get cup
- 3 Step 3: get a **coffee bag**
- 4 Step 4: put bag in cup
- 5 Step 5: add boiled water
- 6 Step 6: wait **30s**
- 7 Step 7: remove bag
- 8 Step 8: add milk

Now we're managing 4 things 

If only...

If only there was a better way...

We wrap the steps in a **function**:

```
make_tea(with_milk)
```



THINKING

function idea!

- 1 Step 1: boil water
- 2 Step 2: get cup
- 3 Step 3: get a teabag
- 4 Step 4: put teabag in cup
- 5 Step 5: add boiled water to cup
- 6 Step 6: wait 1 minute
- 7 Step 7: use spoon to remove teabag
- if with_milk:
- 8 Step 8: add milk

// one name. One recipe. Call it with or without milk.

More functions for your functions



DECIDING

tea or coffee?

```
make_hotdrink(with_milk, ??? )
```

// What could go in that blank? Let's build it up one line at a time.

```
1 Step 1: boil water # same for tea, coffee, hot chocolate...
2 Step 2: get cup # still the same, whatever's going in the cup
3 ... ? ? ? ... # teabag? coffee scoop? hot chocolate powder?
4 ... ? ? ? ... # whatever it is, it still goes "in the cup"
5 ... ? ? ? ...
```

// the first two steps are the same for any hot drink. The rest depend on which one.

More functions for your functions



HAPPY
got it!

```
make_hotdrink(with_milk, drink)
```

```
1 Step 1: boil water
2 Step 2: get cup
3 if drink == "tea":
    Step 3: get a teabag
4 elif drink == "coffee":
    Step 3: get a spoonful of coffee
5 Step 4: put drink in cup
6 Step 5: add boiled water to cup
7 Step 6: wait 1 minute
8 Step 7: use spoon to remove bag
    if with_milk:
9 Step 8: add milk
```

// one recipe, many drinks — and with_milk still works exactly like before.

But... what *is* a function?

It's the same idea, everywhere



// in maths – in code

$$y = m \cdot x + b$$



```
1 def f(x):  
2     return m * x + b
```

Ultimately: a function is just a series of repeatable instructions — sometimes based on input.

THINKING

same idea,
everywhere

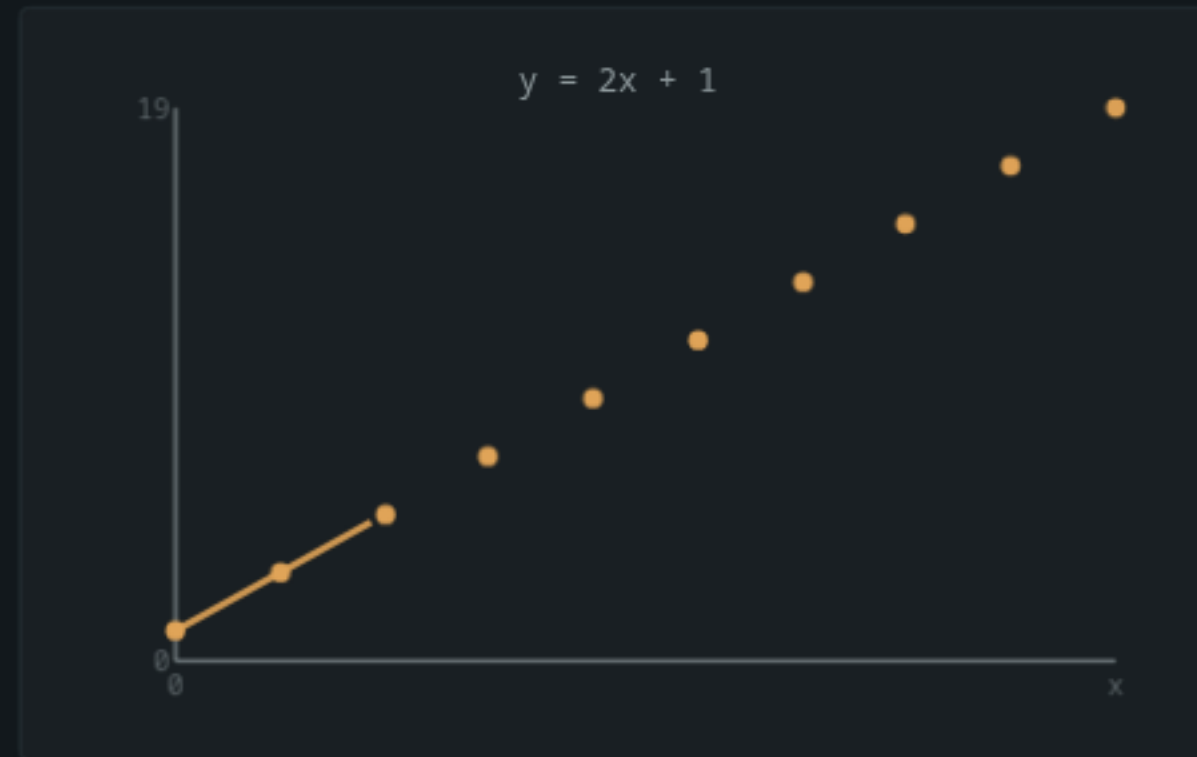
// same shape, whether you write it with symbols or with code.

Code examples with functions



NORMAL

one point per
click...



Plot $y = 2x + 1$ — one call
at a time. Click through:

```
1 plot(0, f(0)) # (0, 1)
2 plot(1, f(1)) # (1, 3)
3 plot(2, f(2)) # (2, 5)
4 plot(3, f(3)) # (3, 7)
5 plot(4, f(4)) # (4, 9)
6 plot(5, f(5)) # (5, 11)
7 plot(6, f(6)) # (6, 13)
8 plot(7, f(7)) # (7, 15)
9 plot(8, f(8)) # (8, 17)
10 plot(9, f(9)) # (9, 19)
```

// ten points, one straight line. And "f(x)" got typed out ten times to get there.

Code examples with functions



HAPPY

the loop does
it for me!

// `for x in range(10):` repeats the line below it 10 times, once for each value of x from 0 to 9. More on loops next time.



For our line, **m = 2** and
b = 1:

```
1 m = 2
2 b = 1

3 for x in range(10):
4     plot(x, f(x))
```

Code examples with functions



NORMAL

any line, one
call

// same ten points, same line — but `plot_straight_line` works for any `m` and `b` you hand it, not just this one.



`f(x)` only ever draws **this** line. Let's write one that draws **any** line:

```
1 def plot_straight_line(m, b):  
2     for x in range(10):  
3         plot(x, m * x + b)  
  
4 plot_straight_line(2, 1)
```