Functions, parameters & references

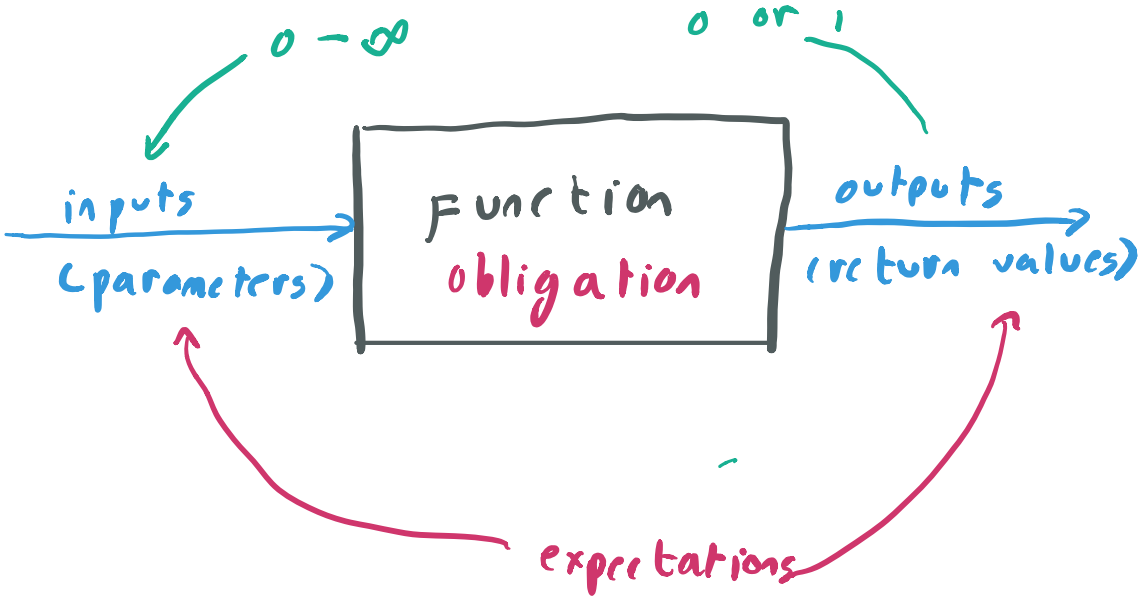
- A function is a **contract** in your program

obligations

what you'll be doing

expectations

what you need & what you'll produce



- when we define a function, 2 promises:

these are summarized in function comment

1. we'll provide specified inputs
2. we'll get back a particular output

```
def add_3_nums(a, b, c):
    .
    .
    .
    return sum
```

promise #1: fn will be called with 3 parameters

promise #2: returning this sum

Abstraction

↳ if you're outside the function, you don't care about the inside & vice versa

```
def main():
    a = 1
    b = 2
    c = 3
    n = add_3_nums(a, b, c)
```

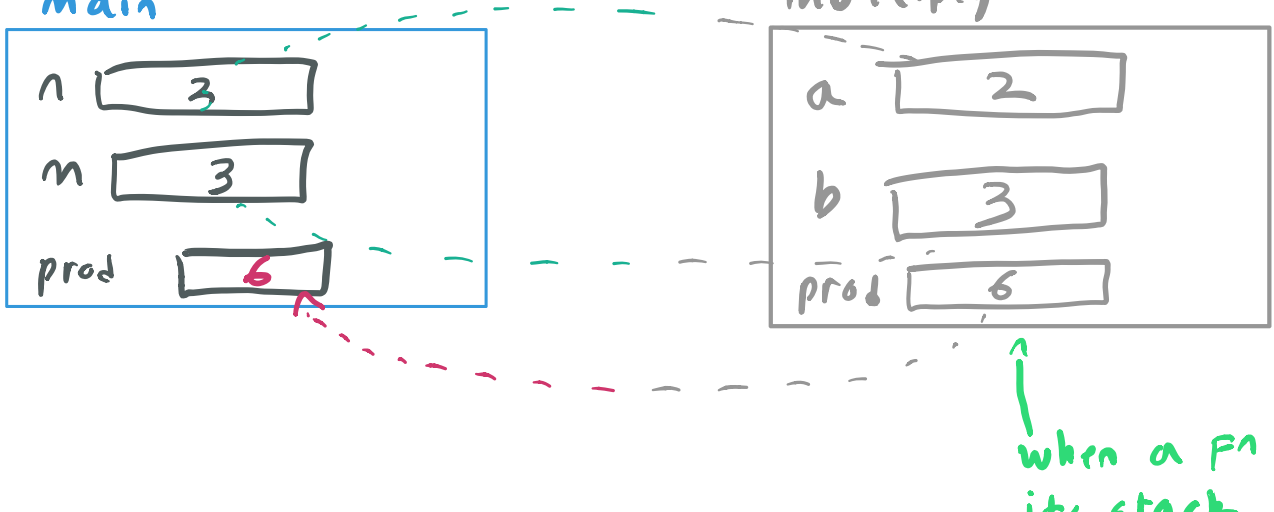
```
def add_3_nums(a, b, c):
    :
    :
    return sum
```

Function calling mechanics

```
def main():
    n = 2
    m = 3
    prod = multiply(n, m)
    print(prod)
```

```
def multiply(a, b):
    prod = a * b
    return prod
```

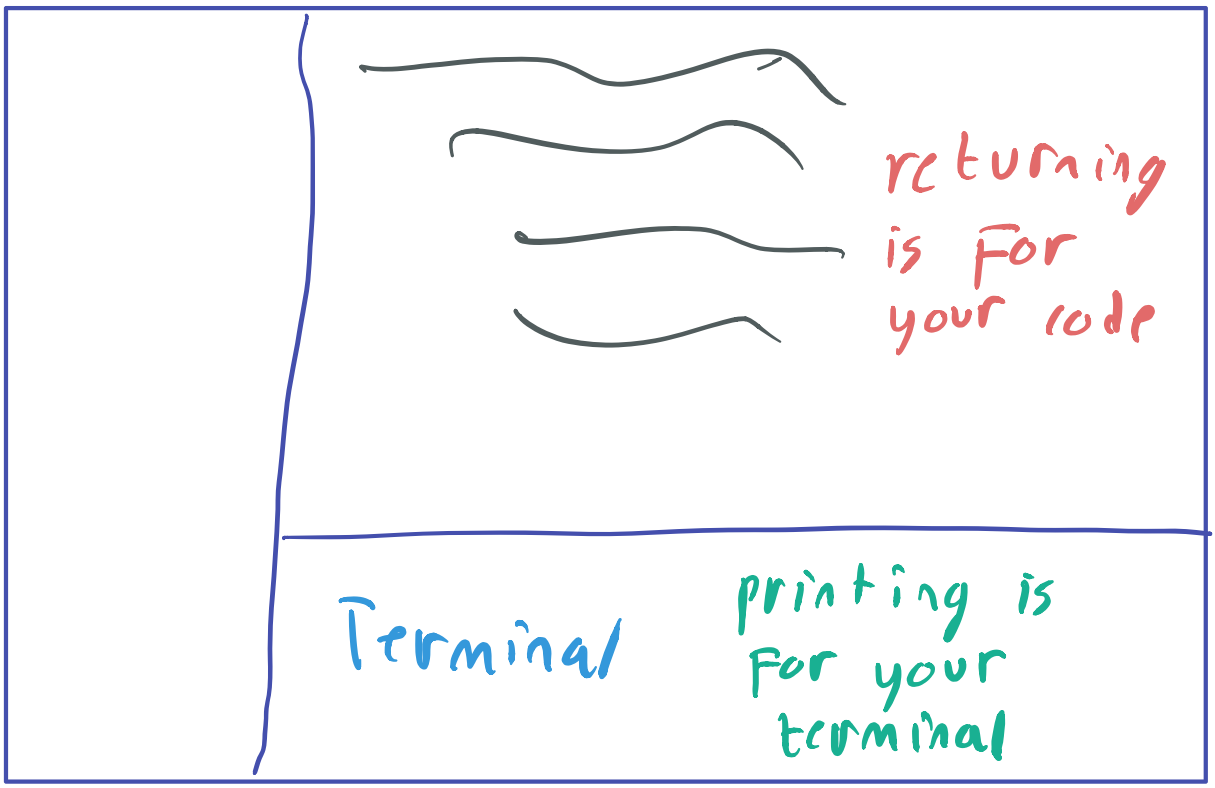
'stack frame'



is deleted

printing & returning

Pycharm:



↳ printing outputs info to the terminal's text area

↳ returning outputs info to the program's information flow

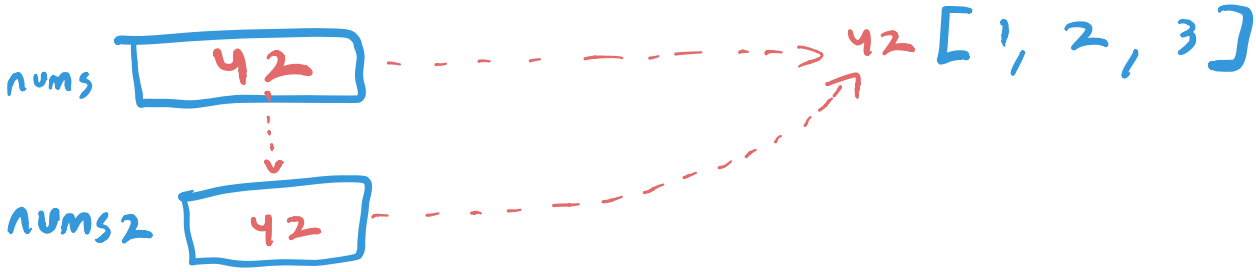
Passing by reference

nums = [1, 2, 3]

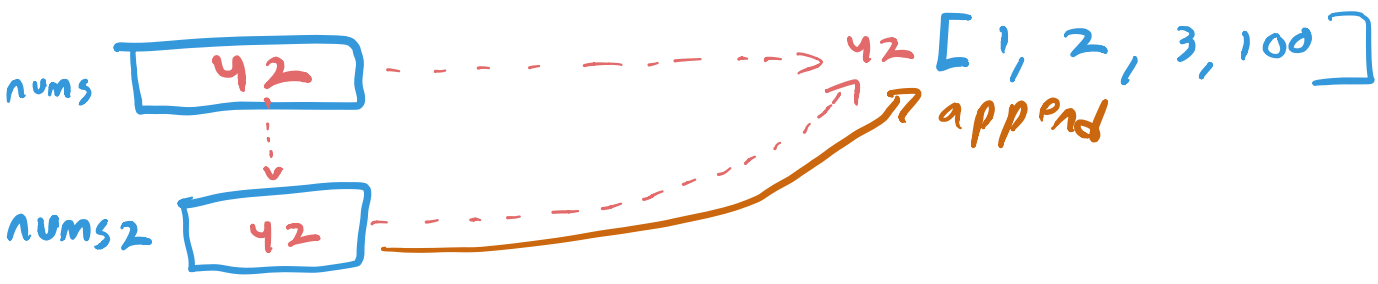
memory:

nums2 = nums

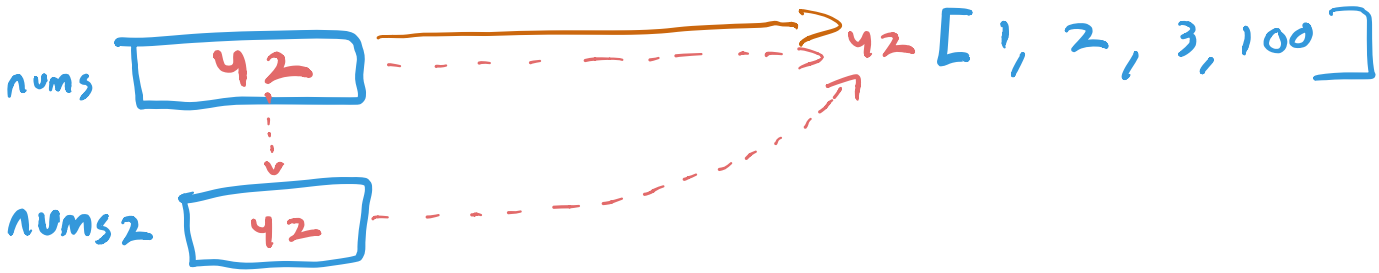
memory:



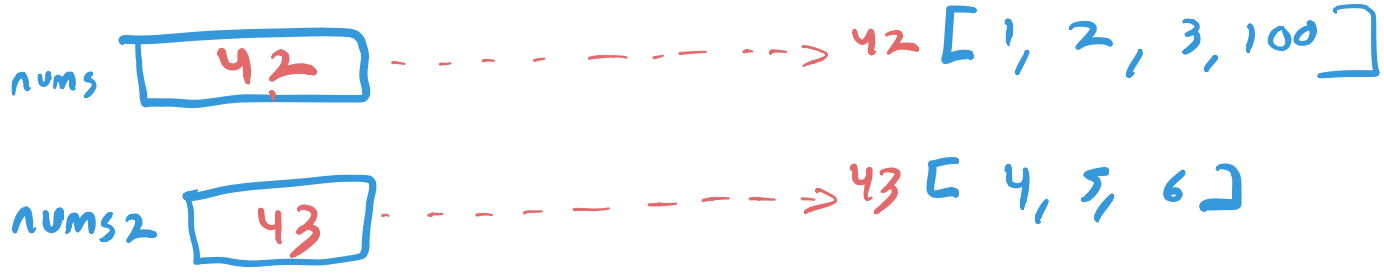
nums2.append(100)



print(nums)

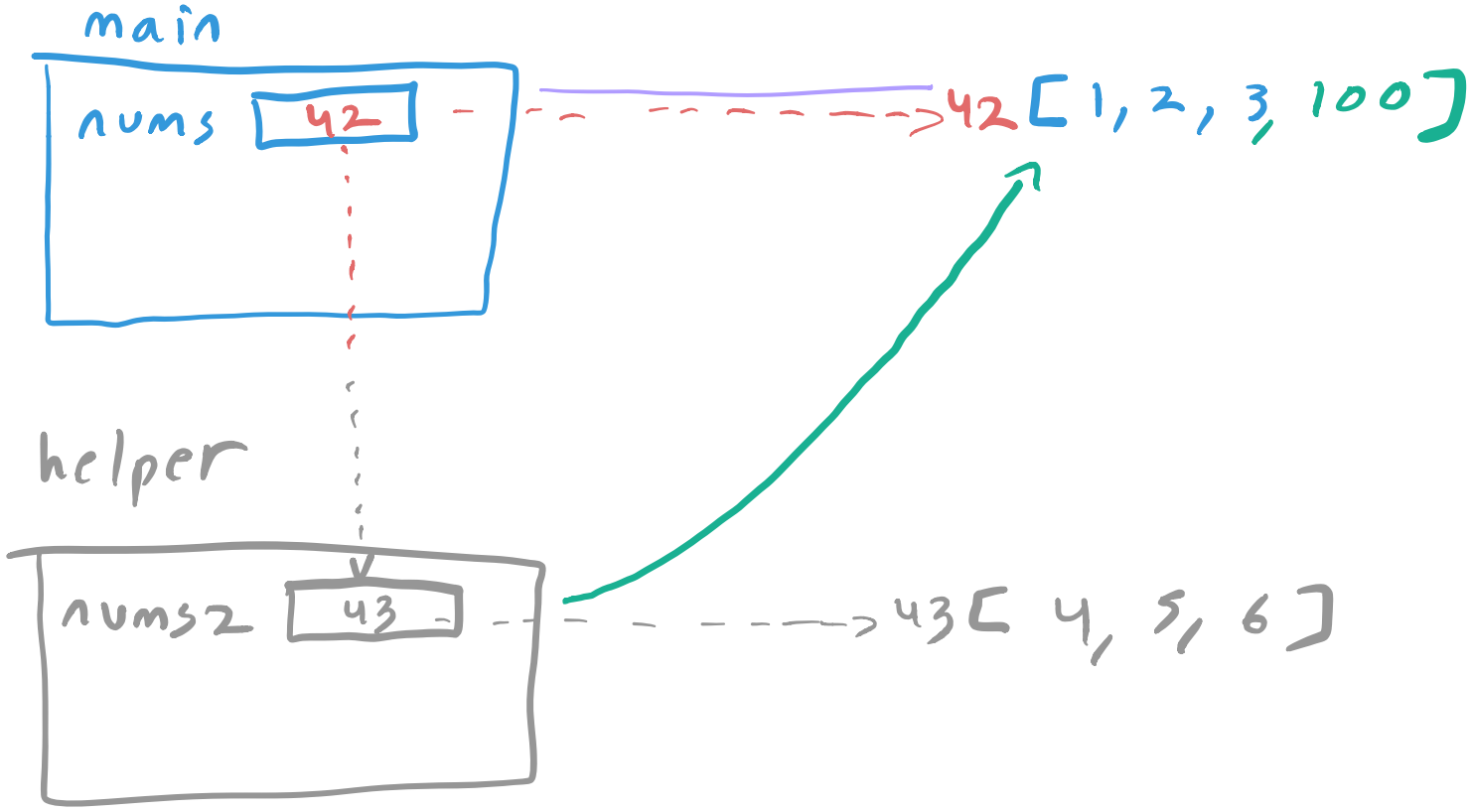


```
nums2 = [4, 5, 6]  # making a new list
```




```
def main():
    nums = [1, 2, 3]
    helper(nums)
    print(nums)
```

```
def helper(nums2):
    nums2.append(100)
    nums2 = [4, 5, 6]
```



→ passing by reference applies to lists, dictionaries, images & pixels, canvases