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

These are summarized in function comment

def add_3_nums(a, b, c):
    return sum
**Abstraction**

If you’re outside the function, you don’t care about the inside & vice versa.

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

**Function calling mechanics**

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

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

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

```
stack frame
```

```
when a fn completes its stack frame
```
**printing & returning**

Pycharm:

1. Printing outputs info to the terminal's text area
2. Returning outputs info to the program's information flow
Passing by reference

```python
nums = [1, 2, 3]
memory:

nums = 42
```

```python
nums2 = nums
memory:

nums = 42
nums2 = 42
```

```python
nums2.append(100)
memory:

nums = 42
nums2 = 42
```

```python
print(nums)
memory:

nums = 42
nums2 = 42
```
nums2 = [4, 5, 6] # making a new list
nums  
\[42\]  
\[\rightarrow \]  
nums \[\rightarrow \]  
nums2 \[43\]  
\[\rightarrow \]  
nums2 \[43\]  
\[\rightarrow \]  

```python
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, arrays, images & pixels, canvas'es