Sometimes we copy and paste our code

```
import datetime
# print timestamps to see how long sections
# take to run

first_name = 'Susan'
print('task completed')
print(datetime.datetime.now())
print()

for x in range(0,10):
      print(x)
print('task completed')
print(datetime.datetime.now())
print()
```

```
task completed
2019-05-30 16:55:01.815327

0
1
2
3
4
5
6
7
8
9
task completed
2019-05-30 16:55:01.817263
```

Use functions instead of repeating code

```
import datetime
# Print the current time
def print_time():
      print('task completed')
      print(datetime.datetime.now())
      print()

first_name = 'Susan'
print_time()

for x in range(0,10):
      print(x)
print_time()
```

```
task completed
2019-05-30 16:55:45.397319

0
1
2
3
4
5
6
7
8
9
task completed
2019-05-30 16:55:45.399314
```

By moving the code to a function, you reduce rework and the chance of introducing bugs when you change the code you had copied

```
# Import the datetime class from datetime library
from datetime import datetime
# Print the current time
def print_time():
    print('task completed')
    # Now I don't need the extra datetime prefix
    print(datetime.now())
    print()
```

What if I want a different message displayed?

```
from datetime import datetime
# print timestamps to see how long sections
# take to run

first_name = 'Susan'
print('first name assigned')
print(datetime.now())
print()

for x in range(0,10):
      print(x)
print('loop completed')
print(datetime.now())
print()
```

```
first name assigned
2019-05-31 10:18:53.419754

0
1
2
3
4
5
6
7
8
9
loop completed
2019-05-31 10:18:53.422748
```

Pass the task name as a parameter

```
from datetime import datetime

# Print the current time and task name
def print_time(task_name):
      print(task_name)
      print(datetime.now())
      print()

first_name = 'Susan'
print_time('first name assigned')

for x in range(0,10):
    print(x)
print_time('loop completed')
```

```
first name assigned
2019-05-31 10:18:53.419754

0
1
2
3
4
5
6
7
8
9
loop completed
2019-05-31 10:18:53.422748
```

Here's another example where the code looks different but we are doing the same logic over and over

```
first_name = input('Enter your first name: ')
first_name_initial = first_name[0:1]
last_name = input('Enter your last name: ')
last_name_initial = last_name[0:1]

print('Your initials are: ' + first_name_initial \
      + last_name_initial)
```

```
# output
Enter your first name: Susan
Enter your last name: Ibach
Your initials are: SI
```

I can still use a function, but this time my function returns a value

```
def get_initial(name):
      initial = name[0:1]
      return initial

first_name = input('Enter your first name: ')
first_name_initial = get_initial(first_name)

last_name = input('Enter your last name: ')
last_name_initial = get_initial(last_name)
```

```
# output
Enter your first name: susan
Enter your last name: ibach
Your initials are: si
```

If you need to change something you only have to change it in one place!

```
def get_initial(name):
      initial = name[0:1].upper()
      return initial

first_name = input('Enter your first name: ')
first_name_initial = get_initial(first_name)

last_name = input('Enter your last name: ')
last_name_initial = get_initial(last_name)
```

```
# output
Enter your first name: susan
Enter your last name: ibach
Your initials are: SI
```

Functions that return values allow clever code, but you might trade readability for less code

```
def get_initial(name):
      initial = name[0:1].upper()
      return initial

first_name = input('Enter your first name: ')
last_name = input('Enter your last name: ')

print('Your initials are: ' \
      + get_initial(first_name) \
      + get_initial(last_name))
```

```
# output
Enter your first name: susan
Enter your last name: ibach
Your initials are: SI
```

Functions make your code more readable and easier to maintain

Always add comments to explain the purpose of your functions

Functions must be declared before the line of code where the function is called