

# PythonTip 01 - Functions (pre-class-version)

February 7, 2022

## 1 Python Tip #1: Functions

Functions are separately defined code snippets that you can then use in your main code.

```
[ ]: def double(number): # arguments = input
      new_number = 2*number
      return new_number
```

```
[ ]: double(5)
```

```
[ ]: def print_hello(x):
      print(x+1)
```

```
[ ]: print_hello(5)
```

```
[ ]:
```

```
[ ]: def double(number=7): # number will default to 7 if you don't specify it
      new_number = 2*number
      return new_number
```

```
[ ]: double()
```

```
[ ]: double(5)
```

```
[ ]: double(number=5)
```

```
[ ]: def double(number): # arguments = input
      new_number = 2*number
      return new_number
```

“Lambda Functions” sound very fancy, but they are just a quicker way to define very simple functions.

```
double = lambda x : 2*x
```

```
[name] = lambda [inputs] : [outputs]
```

```
[ ]: new_double = lambda number : 2*number
new_double(5)
```

```
[ ]: combine = lambda x, y: 2*x + 3 * y**2
```

```
[ ]: combine(5,2)
```

```
[ ]:
```

They are often useful (as we'll see later) for extracting one component of a tuple or list.

```
[ ]: second_component = lambda r : r[1]
```

```
[ ]: second_component([5, -8, 1])
```

This is totally equivalent to:

```
def second_component(r):
    return r[1]
```

This is mostly useful when you just want to use the function in one spot, and not define it forever.

When sorting a list, you can give it a “key” function to tell it what to sort by.

```
[ ]: L = [-5, 1, 0, 7, -10]
print(L)
L.sort()
```

```
[ ]: L.sort(key=lambda x : abs(x))
```

```
[ ]: L
```

```
[ ]:
```

```
[ ]: L = [(0, 3), (-1, 7), (2, 5)]
```

```
[ ]: sorted(L)
```

```
[ ]: L
```

```
[ ]: sorted(L, key=lambda x : x[1])
```

```
[ ]:
```