Abstraction

Functional Abstractions

- Square takes one argument.
- Square has the intrinsic name square.
- Square computes the square of a number.
- Square computes the square by calling mul.

```python
def square(x):
    return pow(x, 2)
```

- If the name "square" were bound to a built-in function, `sum_squares` would still work identically.

```python
def square(x):
    return mul(x, x-1) + x
```

Which Values Deserve a Name

Reasons to add a new name

- Repeated compound expressions:
  ```python
  if sqrt(square(a) + square(b)) > 1:
      x = x + sqrt(square(a) + square(b))
  ```

- Meaningful parts of complex expressions:
  ```python
  x = (-b + sqrt(square(b) - 4 * a * c)) / (2 * a)
  hypotenuse = sqrt(square(a) + square(b))
  if hypotenuse > 1:
      x = x + hypotenuse
  ```

More Naming Tips

- Names can be long if they help document your code:
  ```python
  average_age = average(age, students)
  ```

- Names can be short if they represent generic quantities: counts, arbitrary functions, arguments to mathematical operations, etc.
  ```python
  n, k, i - Usually integers
  x, y, z - Usually real numbers
  f, g, h - Usually functions
  ```

Testing

Test-Driven Development

Write the test of a function before you write the function.

A test will clarify the domain, range, & behavior of a function.

Tests can help identify tricky edge cases.

Develop incrementally and test each piece before moving on.

You can’t depend upon code that hasn’t been tested.

Run your old tests again after you make new changes.

Bonus idea: Run your code interactively.

Don’t be afraid to experiment with a function after you write it.

Interactive sessions can become doctests. Just copy and paste.
**Function Currying**

```python
def make_adder(n):
    return lambda k: n + k

>>> make_adder(2)(3)
5
>>> add(2, 3)
5
```

Curry: Transform a multi-argument function into a single-argument, higher-order function

**Function Decorators**

```python
@trace1
def triple(x):
    return 3 * x

def square(x):
    return x * x
```

A function that takes any argument and returns a function that returns that arg

```python
print(print(5))
None
```

```python
delay(delay)()(6)
5
```

```python
print(delay(print)()(4))
```

Interactive Output

```text
5
6
5
None
5
None
5
None
```

**What Would Python Print?**

The print function returns None. It also displays its arguments (separated by spaces) when it is called.

```python
from operator import add, mul
```