Design
Announcements
Abstraction
Functional Abstractions
def square(x):
    return mul(x, x)
Functional Abstractions

```python
def square(x):
    return mul(x, x)
def sum_squares(x, y):
    return square(x) + square(y)
```
Functional Abstractions

```python
def square(x):
    return mul(x, x)

def sum_squares(x, y):
    return square(x) + square(y)
```

What does `sum_squares` need to know about `square`?
**Functional Abstractions**

```python
def square(x):
    return mul(x, x)

def sum_squares(x, y):
    return square(x) + square(y)
```

What does `sum_squares` need to know about `square`?

- Square takes one argument.
Functional Abstractions

```python
def square(x):
    return mul(x, x)

def sum_squares(x, y):
    return square(x) + square(y)
```

What does `sum_squares` need to know about `square`?

- Square takes one argument.  Yes
Functional Abstractions

```python
def square(x):
    return mul(x, x)

def sum_squares(x, y):
    return square(x) + square(y)
```

What does `sum_squares` need to know about `square`?

• Square takes one argument.  
  
  Yes

• Square has the intrinsic name `square`. 
  

Functional Abstractions

```python
def square(x):
    return mul(x, x)

def sum_squares(x, y):
    return square(x) + square(y)
```

What does `sum_squares` need to know about `square`?

• Square takes one argument. Yes

• Square has the intrinsic name `square`. No
Functional Abstractions

```python
def square(x):
    return mul(x, x)
def sum_squares(x, y):
    return square(x) + square(y)
```

What does `sum_squares` need to know about `square`?

- Square takes one argument.  
  
  Yes

- Square has the intrinsic name `square`.  
  
  No

- Square computes the square of a number.
Functional Abstractions

```python
def square(x):
    return mul(x, x)

def sum_squares(x, y):
    return square(x) + square(y)
```

What does `sum_squares` need to know about `square`?

- Square takes one argument.  
  - Yes
- Square has the intrinsic name `square`.  
  - No
- Square computes the square of a number.  
  - Yes
Functional Abstractions

```python
def square(x):
    return mul(x, x)

def sum_squares(x, y):
    return square(x) + square(y)
```

What does `sum_squares` need to know about `square`?

- Square takes one argument.  
  - Yes
- Square has the intrinsic name `square`.  
  - No
- Square computes the square of a number.  
  - Yes
- Square computes the square by calling `mul`.  
  -
Functional Abstractions

```
def square(x):
    return mul(x, x)
def sum_squares(x, y):
    return square(x) + square(y)
```

What does `sum_squares` need to know about `square`?

- Square takes one argument.  
  Yes

- Square has the intrinsic name square.  
  No

- Square computes the square of a number.  
  Yes

- Square computes the square by calling `mul`.  
  No
**Functional Abstractions**

```python
def square(x):
    return mul(x, x)
```

```python
def sum_squares(x, y):
    return square(x) + square(y)
```

What does `sum_squares` need to know about `square`?

- Square takes one argument.  
  - Yes

- Square has the intrinsic name `square`.  
  - No

- Square computes the square of a number.  
  - Yes

- Square computes the square by calling `mul`.  
  - No

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

```python
def square(x):
    return mul(x, x)

def sum_squares(x, y):
    return square(x) + square(y)
```

What does sum_squares need to know about square?

- Square takes one argument.  
  - Yes
- Square has the intrinsic name square.  
  - No
- Square computes the square of a number.  
  - Yes
- Square computes the square by calling mul.  
  - No

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

def square(x):
    return mul(x, x-1) + x
```
**Functional Abstractions**

```python
def square(x):
    return mul(x, x)

def sum_squares(x, y):
    return square(x) + square(y)
```

What does `sum_squares` need to know about `square`?

- Square takes one argument.  
  - Yes
- Square has the intrinsic name `square`.  
  - No
- Square computes the square of a number.  
  - Yes
- Square computes the square by calling `mul`.  
  - No

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

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

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

Names typically don’t matter for correctness

**but**

they matter a lot for composition
Choosing Names

Names typically don’t matter for correctness

but

they matter a lot for composition

Names should convey the meaning or purpose
of the values to which they are bound.
Choosing Names

Names typically don’t matter for correctness

but

they matter a lot for composition

Names should convey the meaning or purpose of the values to which they are bound.

The type of value bound to the name is best documented in a function's docstring.
Choosing Names

Names typically don’t matter for correctness

but

they matter a lot for composition

Names should convey the meaning or purpose of the values to which they are bound.

The type of value bound to the name is best documented in a function's docstring.

Function names typically convey their effect (print), their behavior (triple), or the value returned (abs).
Choosing Names

Names typically don’t matter for correctness

**but**

eyes matter a lot for composition

Names should convey the meaning or purpose of the values to which they are bound.

The type of value bound to the name is best documented in a function's docstring.

Function names typically convey their effect (**print**), their behavior (**triple**), or the value returned (**abs**).
Choosing Names

Names typically don’t matter for correctness

but

they matter a lot for composition

<table>
<thead>
<tr>
<th>From:</th>
<th>To:</th>
</tr>
</thead>
<tbody>
<tr>
<td>true_false</td>
<td>rolled_a_one</td>
</tr>
</tbody>
</table>

Names should convey the meaning or purpose of the values to which they are bound.

The type of value bound to the name is best documented in a function's docstring.

Function names typically convey their effect (print), their behavior (triple), or the value returned (abs).
## Choosing Names

Names typically don’t matter for correctness

**but**

they matter a lot for composition

<table>
<thead>
<tr>
<th>From:</th>
<th>To:</th>
</tr>
</thead>
<tbody>
<tr>
<td>true_false</td>
<td>rolled_a_one</td>
</tr>
<tr>
<td>d</td>
<td>dice</td>
</tr>
</tbody>
</table>

Names should convey the meaning or purpose of the values to which they are bound.

The type of value bound to the name is best documented in a function's docstring.

Function names typically convey their effect (**print**), their behavior (**triple**), or the value returned (**abs**).
Choosing Names

Names typically don’t matter for correctness

*but*

they matter a lot for composition

<table>
<thead>
<tr>
<th>From:</th>
<th>To:</th>
</tr>
</thead>
<tbody>
<tr>
<td>true_false</td>
<td>rolled_a_one</td>
</tr>
<tr>
<td>d</td>
<td>dice</td>
</tr>
<tr>
<td>helper</td>
<td>take_turn</td>
</tr>
</tbody>
</table>

Names should convey the meaning or purpose of the values to which they are bound.

The type of value bound to the name is best documented in a function's docstring.

Function names typically convey their effect (*print*), their behavior (*triple*), or the value returned (*abs*).
Choosing Names

Names typically don’t matter for correctness

but

they matter a lot for composition

<table>
<thead>
<tr>
<th>From:</th>
<th>To:</th>
</tr>
</thead>
<tbody>
<tr>
<td>true_false</td>
<td>rolled_a_one</td>
</tr>
<tr>
<td>d</td>
<td>dice</td>
</tr>
<tr>
<td>helper</td>
<td>take_turn</td>
</tr>
<tr>
<td>my_int</td>
<td>num_rolls</td>
</tr>
</tbody>
</table>

Names should convey the meaning or purpose of the values to which they are bound.

The type of value bound to the name is best documented in a function's docstring.

Function names typically convey their effect (**print**), their behavior (**triple**), or the value returned (**abs**).
Choosing Names

Names typically don’t matter for correctness

*but*

they matter a lot for composition

<table>
<thead>
<tr>
<th>From:</th>
<th>To:</th>
</tr>
</thead>
<tbody>
<tr>
<td>true_false</td>
<td>rolled_a_one</td>
</tr>
<tr>
<td>d</td>
<td>dice</td>
</tr>
<tr>
<td>helper</td>
<td>take_turn</td>
</tr>
<tr>
<td>my_int</td>
<td>num_rolls</td>
</tr>
<tr>
<td>l, I, 0</td>
<td>k, i, m</td>
</tr>
</tbody>
</table>

Names should convey the meaning or purpose of the values to which they are bound.

The type of value bound to the name is best documented in a function's docstring.

Function names typically convey their effect (*print*), their behavior (*triple*), or the value returned (*abs*).
Which Values Deserve a Name

Reasons to add a new name
Which Values Deserve a Name

Reasons to add a new name

Repeated compound expressions:
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))
```
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))

hypotenuse = sqrt(square(a) + square(b))
if hypotenuse > 1:
    x = x + hypotenuse
```
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))
```

```python
hypotenuse = sqrt(square(a) + square(b))
if hypotenuse > 1:
    x = x + hypotenuse
```

Meaningful parts of complex expressions:
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))
```

```python
hypotenuse = sqrt(square(a) + square(b))
if hypotenuse > 1:
    x = x + hypotenuse
```

*Meaningful parts of complex expressions:*

```python
x1 = (-b + sqrt(square(b) - 4 * a * c)) / (2 * a)
```
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))
```

```python
hypotenuse = sqrt(square(a) + square(b))
if hypotenuse > 1:
    x = x + hypotenuse
```

Meaningful parts of complex expressions:

```python
x1 = (-b + sqrt(square(b) - 4 * a * c)) / (2 * a)
```

```python
discriminant = square(b) - 4 * a * c
x1 = (-b + sqrt(discriminant)) / (2 * a)
```
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))
```

```python
hypotenuse = sqrt(square(a) + square(b))
if hypotenuse > 1:
    x = x + hypotenuse
```

Meaningful parts of complex expressions:

```python
x1 = (-b + sqrt(square(b) - 4 * a * c)) / (2 * a)
```

```python
discriminant = square(b) - 4 * a * c
x1 = (-b + sqrt(discriminant)) / (2 * a)
```
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))
```

```python
hypotenuse = sqrt(square(a) + square(b))
if hypotenuse > 1:
    x = x + hypotenuse
```

Meaningful parts of complex expressions:

```python
x1 = (-b + sqrt(square(b) - 4 * a * c)) / (2 * a)
```

```python
discriminant = square(b) - 4 * a * c
x1 = (-b + sqrt(discriminant)) / (2 * a)
```

More Naming Tips

• Names can be long if they help document your code:

```python
average_age = average(age, students)
```

is preferable to

```python
# Compute average age of students
aa = avg(a, st)
```
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))
```

```python
hypotenuse = sqrt(square(a) + square(b))
if hypotenuse > 1:
    x = x + hypotenuse
```

Meaningful parts of complex expressions:

```python
x1 = (-b + sqrt(square(b) - 4 * a * c)) / (2 * a)
```

```python
discriminant = square(b) - 4 * a * c
x1 = (-b + sqrt(discriminant)) / (2 * a)
```

More Naming Tips

• Names can be long if they help document your code:

```python
average_age = average(age, students)
```

is preferable to

```python
# Compute average age of students
aa = avg(a, st)
```

• Names can be short if they represent generic quantities: counts, arbitrary functions, arguments to mathematical operations, etc.

n, k, i – Usually integers
x, y, z – Usually real numbers
f, g, h – Usually functions
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))
```

```python
hypotenuse = sqrt(square(a) + square(b))
if hypotenuse > 1:
x = x + hypotenuse
```

Meaningful parts of complex expressions:

```python
x1 = (-b + sqrt(square(b) - 4 * a * c)) / (2 * a)
```

```python
discriminant = square(b) - 4 * a * c
x1 = (-b + sqrt(discriminant)) / (2 * a)
```

More Naming Tips

• Names can be long if they help document your code:

  ```python
  average_age = average(age, students)
  ```

  is preferable to

  ```python
  # Compute average age of students
  aa = avg(a, st)
  ```

• 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
  ```
Function Example: Sounds
WAV Files

https://en.wikipedia.org/wiki/Triangle_wave
https://en.wikipedia.org/wiki/Sampling_(signal_processing)
WAV Files

The Waveform Audio File Format encodes a sampled sound wave.

https://en.wikipedia.org/wiki/Triangle_wave
https://en.wikipedia.org/wiki/Sampling_(signal_processing)
The Waveform Audio File Format encodes a sampled sound wave

https://en.wikipedia.org/wiki/Triangle_wave
https://en.wikipedia.org/wiki/Sampling_(signal_processing)
WAV Files

The Waveform Audio File Format encodes a sampled sound wave.

A triangle wave is the simple waveform with the most pleasing sound.

https://en.wikipedia.org/wiki/Triangle_wave
https://en.wikipedia.org/wiki/Sampling_(signal_processing)
WAV Files

The Waveform Audio File Format encodes a sampled sound wave

A triangle wave is the simple waveform with the most pleasing sound

https://en.wikipedia.org/wiki/Triangle_wave
https://en.wikipedia.org/wiki/Sampling_(signal_processing)
WAV Files

The Waveform Audio File Format encodes a sampled sound wave.

A triangle wave is the simple wave form with the most pleasing sound.

- Sine
- Square
- Triangle
- Sawtooth

(Demo)