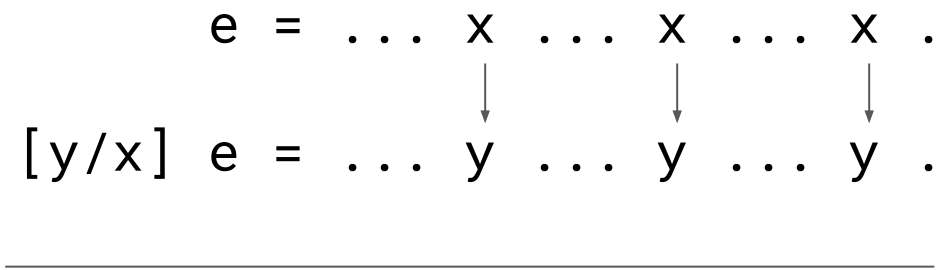
Rewriting Expressions As Computation

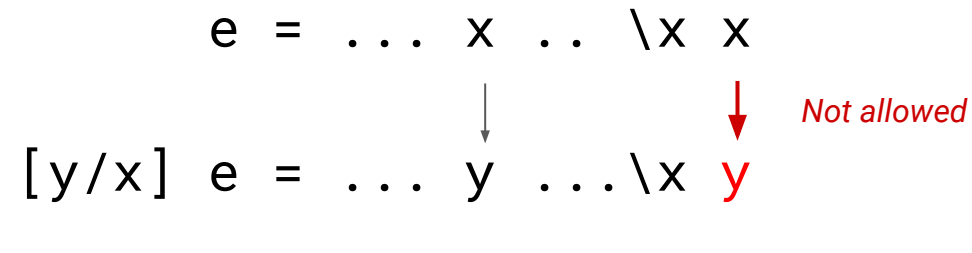
Ken Q Pu, Faculty of Science, Ontario Tech University

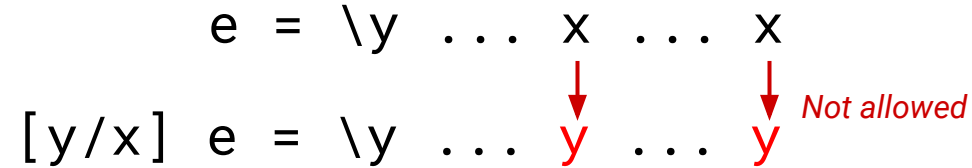
Substitution Revisited

[y/x] e

1. Substitute all *free* occurrences of **x** in **e** with **y**.
2. The substituted **y** remains free in **e**. Namely, it is not accidentally bound to an existing parameter in **e**.







Ken Q Pu, Faculty of Science, Ontario Tech University

Alpha α-conversion

The alpha conversion is all about renaming parameters *correctly*.

$$\backslash x.\ e \longrightarrow_{\alpha} \backslash y.\ [y/x]\ e$$

Provided that we observe the following:

1. We only replace free **x** in e with **y**
2. The name **y** must appear free in e

Thinking about alpha conversion in Python

We can *practice* the alpha conversion using a more familiar programming language that supports function abstraction.

```
y = 10
def add(x):
    return x + y
```

```
y = 10
def add(num):
    return num + y
```

Observe that **y** is free in the function expression, while **x** is bounded to the parameter.

We can replace **x** with **num** without violating the alpha conversion requirements.

Ken Q Pu, Faculty of Science, Ontario Tech University

Thinking about alpha conversion in Python

We can *practice* the alpha conversion using a more familiar programming language that supports function abstraction.

```
y = 10
def add(x):
    return x + y
```

```
y = 10
def add(y):
    return y + y
```

Observe that **y** is free in the function expression, while **x** is bounded to the parameter.

But we cannot replace **x** with **y** because it already appears in expression as a free variable.

Ken Q Pu, Faculty of Science, Ontario Tech University

Thinking about alpha conversion in Python

```
def quad(x):
    def double(x):
        return 2 * x
    return double(2 * x)
```

This **x** are bound in the body of **quad**.

This **x** is free in the body of **quad**.

```
def quad(num):
    def double(x):
        return 2 * x
    return double(2 * num)
```

We can replace all *free* occurrences of **x** in the body.

But we cannot replace the bound occurrences of **x**.

Try to evaluate quad(2).

Ken Q Pu, Faculty of Science, Ontario Tech University

Thinking about alpha conversion in Python

```
def quad(x):
    def double(x):
        return 2 * x
    return double(2 * x)
```

This **x** are bound in the body of **quad**.

This **x** is free in the body of **quad**.

```
def quad(num):
    def double(x):
        return 2 * num
    return double(2 * num)
```

If we naively substitutes all occurrences of **x** in the body of **quad**, we would get into trouble.

Try to evaluate quad(2).

Ken Q Pu, Faculty of Science, Ontario Tech University

Why alpha conversion is so important

Using alpha conversion, we can always rename the parameter of a function.

So, given two expressions, we can make sure that there are no name conflicts between the parameter names and free variable names.

$$\text{Free}(e_1) \cap \text{Bound}(e_2) = \varnothing$$

$$\text{Bound}(e_1) \cap \text{Bound}(e_2) = \varnothing$$

$$\text{Bound}(e_1) \cap \text{Free}(e_2) = \varnothing$$

Ken Q Pu, Faculty of Science, Ontario Tech University

Beta β-reduction

The beta reduction evaluates a function application expression.

It's called a reduction because it strictly reduces the length of the expression.

$$(\backslash x.\ e)\ e' \longrightarrow_{\beta} [e'/x]\ e$$

We must ensure that the substitution is sound.

e' does not create name shadowing.

We can ensure soundness by preprocessing e:

- Rename all parameters in e.
- Rename all the parameters in e'

so there is no overlap..

Eta η-Reduction

Eta reduction can be seen as an optimization rule: it reduces a trivial function definition to a simpler form.

$$\lambda x.\ (e\ x) \rightarrow_{\eta} e$$

```
def myFunc(x):
    return f(x)
```



```
myFunc == f
```

Ken Q Pu, Faculty of Science, Ontario Tech University