Beyond Basic Functions

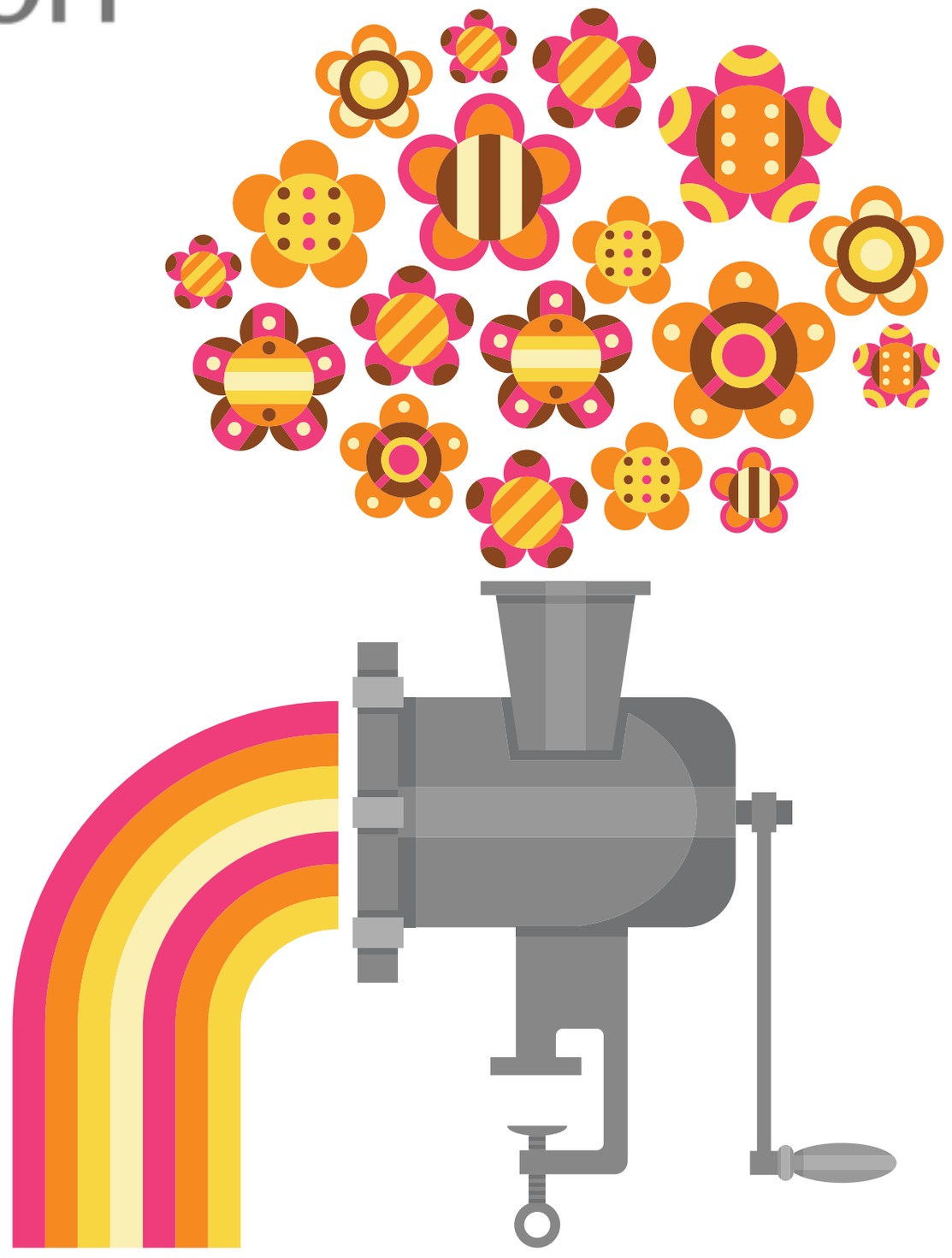
Robert Smallshire
@robsmallshire
rob@sixty-north.com

Presenter

Austin Bingham
@austin_bingham
austin@sixty-north.com

pluralsight
hardcore dev and IT training

python™



default argument value

```
def function_name(arg1, arg2, arg3=1.0):
    """Function docstring"""
    print("Function body")
    return (arg1 + arg2) / arg3
```

positional argument keyword argument

```
function_name(arg1, arg2=1.618)
```

Callable *instances*

```
import socket


class Resolver:

    def __init__(self):
        self._cache = {}

    def __call__(self, host):
        if host not in self._cache:
            self._cache[host] = socket.gethostbyname(host)
        return self._cache[host]

    def clear(self):
        self._cache.clear()

    def has_host(self, host):
        return host in self._cache
```

Callable *classes*

```
>>> from resolver import Resolver
>>> resolve = Resolver()
```

calling

a class

Calling a class invokes the constructor

Conditionals

Conditional statement

```
if condition:
    result = true_value
else:
    result = false_value
```

Conditional expression

```
result = true_value if condition else false_value
```

z

s

sh

K κ kappa k

Λ λ lambda l

M μ mu m

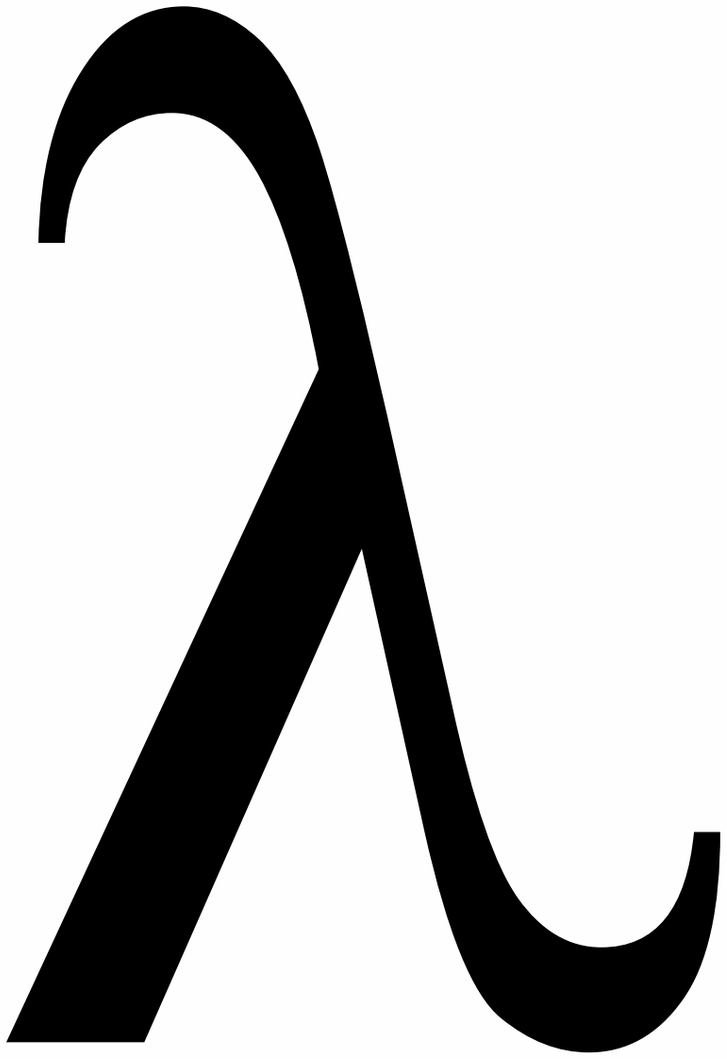
H

O

Π

P

lambda



Alonzo Church

```
def first_name(name):
    """Get first name"""
    return name.split()[0]
```

- *statement* which defines a function and binds it to a name
- Must have a name
- Arguments delimited by parentheses, separated by commas
- Zero or more arguments supported - zero arguments ⇒ empty parentheses
- Body is an indented block of statements
- A `return` statement is required to return anything other than `None`
- Regular functions can have docstrings
- Easy to access for testing

```
lambda name: name.split()[-1]
```

- *expression* which evaluates to a function
- Anonymous
- Argument list terminated by colon, separated by commas
- Zero or more arguments supported - zero arguments ⇒ `lambda:`
- Body is a single *expression*
- The return value is given by the body expression. No `return` statement is permitted.
- Lambdas cannot have docstrings
- Awkward or impossible to test

Extended formal argument syntax

```
def extended(*args, **kwargs):
```

Extended formal argument syntax

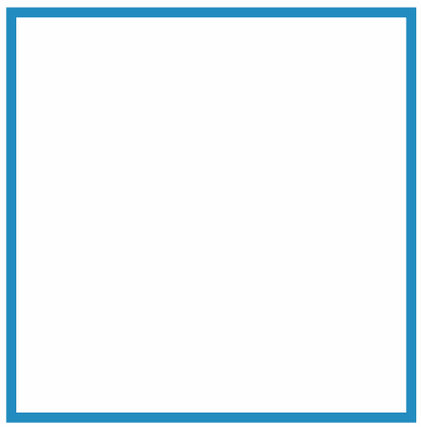
```
def extended(*args, **kwargs):
```

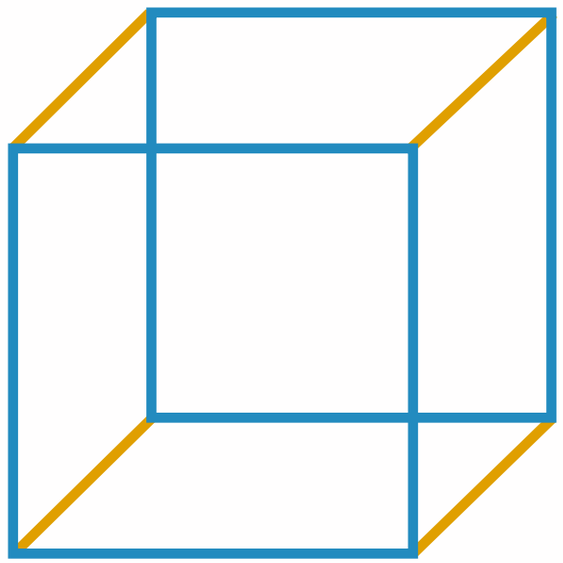
Extended formal argument syntax

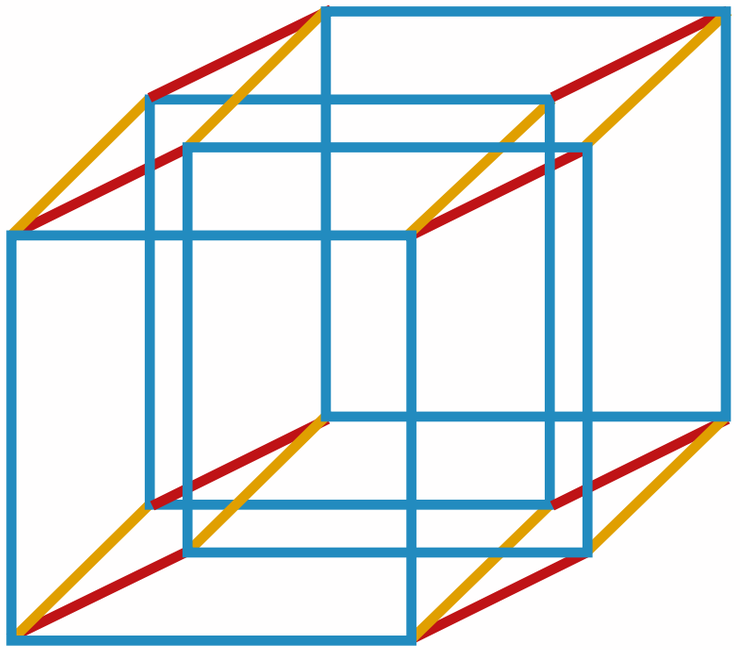
```
def extended(*args, **kwargs):
```

Formal Arguments

arguments at the function **definition** site

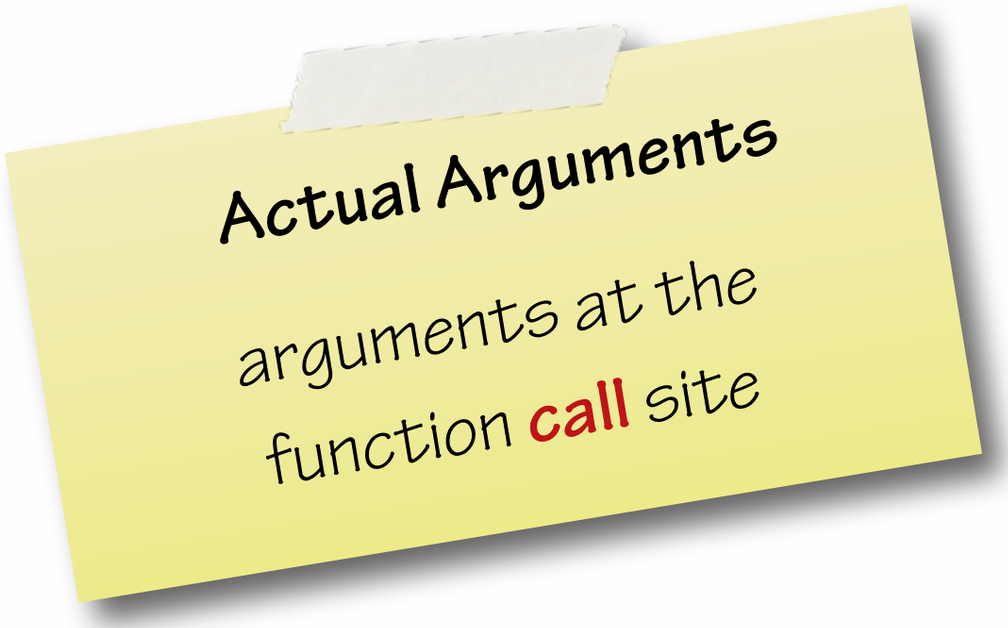






Extended actual argument syntax

```
extended(*args, **kwargs)
```



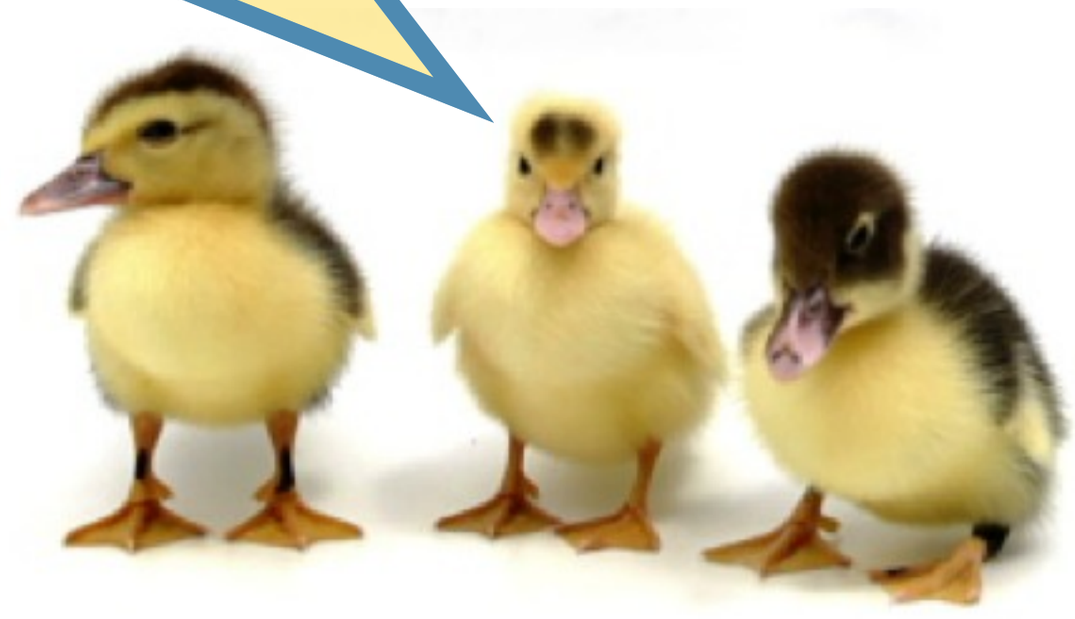
Duck Tails

Transposing Tables

Duck Tails

A

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$



Beyond Basic Functions

`callable(obj)`

`lambda`

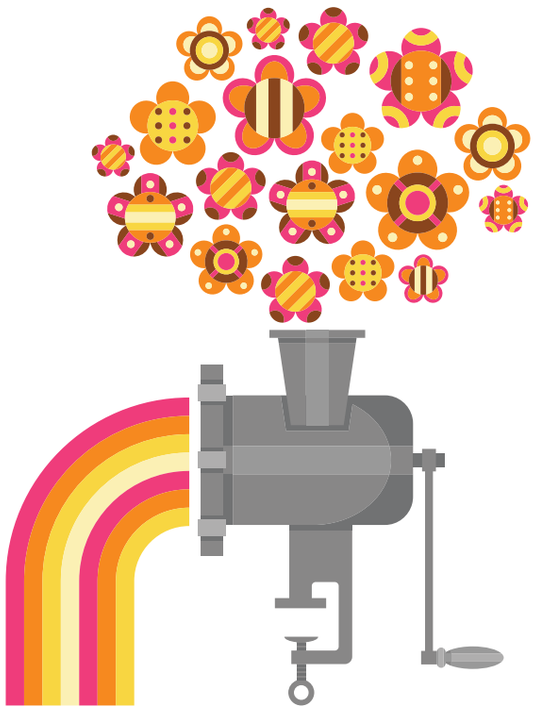
constructor

callable classes

attributes

functions with state

methods



`*args , **kwargs`

formal arguments

actual arguments

`timeit`

conditional expressions

`__call__()`

callables

`list(zip(*table))`