Anonymous Functions: lambda

- Python also provides an expression form that generates function objects, it's called lambda
- Lambda returns the function instead of assigning it to a name, This is why lambdas are sometimes known as *anonymous* (i.e., unnamed) functions
- The lambda's general form:

```
lambda argument1, argument2,... argumentN : expression using arguments
```

- lambda is an expression, not a statement.
- lambda's body is a single expression, not a block of statements.

1

Anonymous Functions: lambda

```
>>> def func(x, y, z): return x + y + z

>>> func(2, 3, 4)
9

>>> f = lambda x, y, z: x + y + z
>>> f(2, 3, 4)
9
```

```
>>> def knights():
        title = 'Sir'
        action = (lambda x: title + ' ' + x)      # Title in enclosing def scope
        return action                             # Return a function object

>>> act = knights()
>>> msg = act('robin')                            # 'robin' passed to x
>>> msg
'Sir robin'

>>> act                                           # act: a function, not its result
<function knights.<locals>.<lambda> at 0x00000000029CA488>
```

2

Multiway branch switches: The finale

```
>>> key = 'got'
>>> {'already': (lambda: 2 + 2),
     'got':     (lambda: 2 * 4),
     'one':     (lambda: 2 ** 6)}[key]()
8

>>> def f1(): return 2 + 2

>>> def f2(): return 2 * 4

>>> def f3(): return 2 ** 6

>>> key = 'one'
>>> {'already': f1, 'got': f2, 'one': f3}[key]()
64
```

3