

PRO TIP: DICTIONARY COMPREHENSIONS

Function Components

Defining Functions

Variable Scope

Modules

Packages

Lambda Functions

Comprehensions

Comprehensions can also **create dictionaries** from other iterables

```
Syntax: new_dict = {key: value for key, value in other_iterable(if condition)}

These can be expressions (including function calls!)
```

EXAMPLE

Creating a dictionary of inventory costs per item (stock quantity * price)

```
items = ['skis', 'snowboard', 'goggles', 'boots']
status = [[5, 249.99], [0, 219.99], [0, 99.99], [12, 79.99]]
inventory_costs = {k: round(v[0] * v[1], 2) for k, v in zip(items, status)}
inventory_costs
{'skis': 1249.95, 'snowboard': 0.0, 'goggles': 0.0, 'boots': 959.88}
```

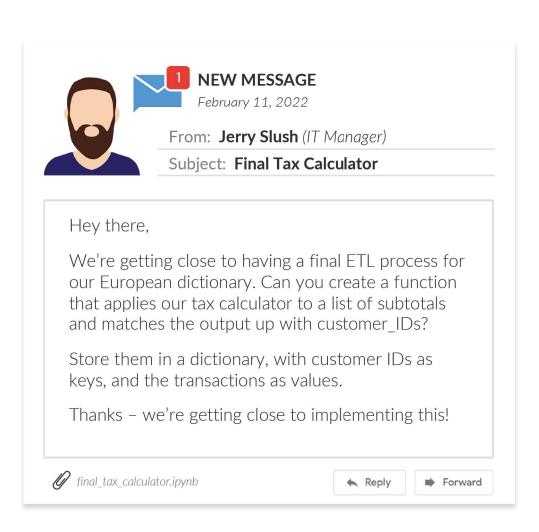
This is creating a dictionary by using *items* as keys, and the product of *status[0]* and *status[1]* as values

zip() is being used to stitch the two lists together into a single iterable

```
inventory_costs = {
    k: round(v[0] * v[1], 2) for k, v in zip(items, status) if v[0] > 0
}
inventory_costs
{'skis': 1249.95, 'boots': 959.88}
```

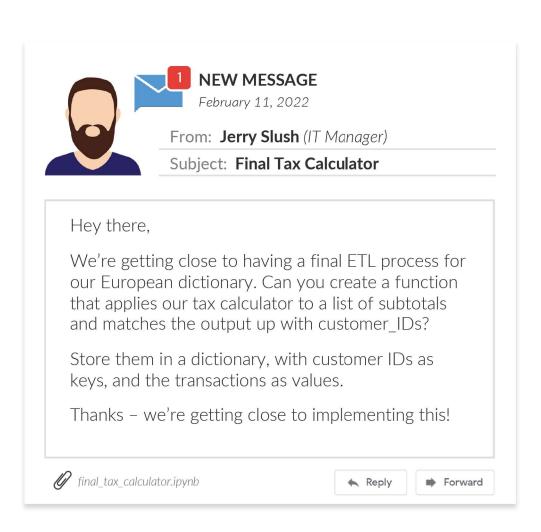
You can still use conditional logic!

ASSIGNMENT: DICTIONARY COMPREHENSIONS



Results Preview

ASSIGNMENT: DICTIONARY COMPREHENSIONS



Solution Code

```
from tax calculator import tax calculator
customer ids = ['C00004', 'C00007', 'C00015', 'C00016', 'C00020', 'C00010']
subtotals = [15.98, 899.97, 799.97, 117.96, 5.99, 599.99]
def transaction dict creator(customer ids, subtotals, tax rate):
    # docstring omitted for screenshot
    customer dict = {
        customer id: tax calculator(subtotal, tax rate)
        for customer id, subtotal in zip(customer ids, subtotals)
    return customer dict
transaction dict creator(customer ids, subtotals, .08)
{'C00004': [15.98, 1.28, 17.26],
 'C00007': [899.97, 72.0, 971.97],
 'C00015': [799.97, 64.0, 863.97],
 'C00016': [117.96, 9.44, 127.4],
 'C00020': [5.99, 0.48, 6.47],
 'C00010': [599.99, 48.0, 647.99]}
```