# Warm-Up: Match the description to the code

```
w = {...}  # a dict with unique keys and values

m = {v: k for k, v in w.items()}
```

Which expression evaluates to?

1. The key that has the smallest value in w
2. The value that has the smallest key in w
3. The smallest absolute difference between a key and its value

```
min(w.keys(), key=lambda k: w[k])

min(w.keys(), key=lambda k: m[k])

min(w.values(), key=lambda v: w[v])

min(w.values(), key=lambda v: m[v])

min(w.keys(), key=lambda k: abs(k - w[k]))

min(w.keys(), key=lambda k: abs(k - m[k]))

min(map(lambda k: abs(k - w[k]), w.keys()))

min(map(lambda k: abs(k - m[k]), w.keys()))
```

# Generators

# Announcements

# Generators

# Generators and Generator Functions

```
>>> def plus_minus(x):
...     yield x
...     yield -x

>>> t = plus_minus(3)
>>> next(t)
3
>>> next(t)
-3
>>> t
<generator object plus_minus ...>
```

A *generator function* is a function that **yield**s values instead of **return**ing them

A normal function **return**s once; a *generator function* can **yield** multiple times

A *generator* is an iterator created automatically by calling a *generator function*

When a *generator function* is called, it returns a *generator* that iterates over its yields

(Demo)