In Python, **`lambda`** is used to create a small, *anonymous function* (a function without a name) in a single line.

General form:

```
lambda arguments: expression
```

- `arguments` → the input parameters
- `expression` → what gets returned

It's like writing a mini `def` function, but shorter.

Example 1: Simple lambda

```
f = lambda x: x * 2
print(f(5))  # 10
```

This is the same as:

```
def f(x):
    return x * 2
```

Example 2: With two arguments

```
f = lambda a, b: a + b
print(f(3, 4))  # 7
```

Same as:

```
def f(a, b):
    return a + b
```

In your code:

```
lambda a, b: "Invalid"
```

This is a function that:

- Takes **two arguments** (`a` and `b`),
- But **ignores them**,
- And always returns `"Invalid"`.

So if no matching operation is found, the program won't crash (with a `KeyError`), it will just call this fallback lambda and get `"Invalid"`.

🔑 Why use `lambda` here?
Because we need to return *something callable* (a function) from `.get()`.
That way, we can still do `(...)(5, 3)` without worrying if the key existed.