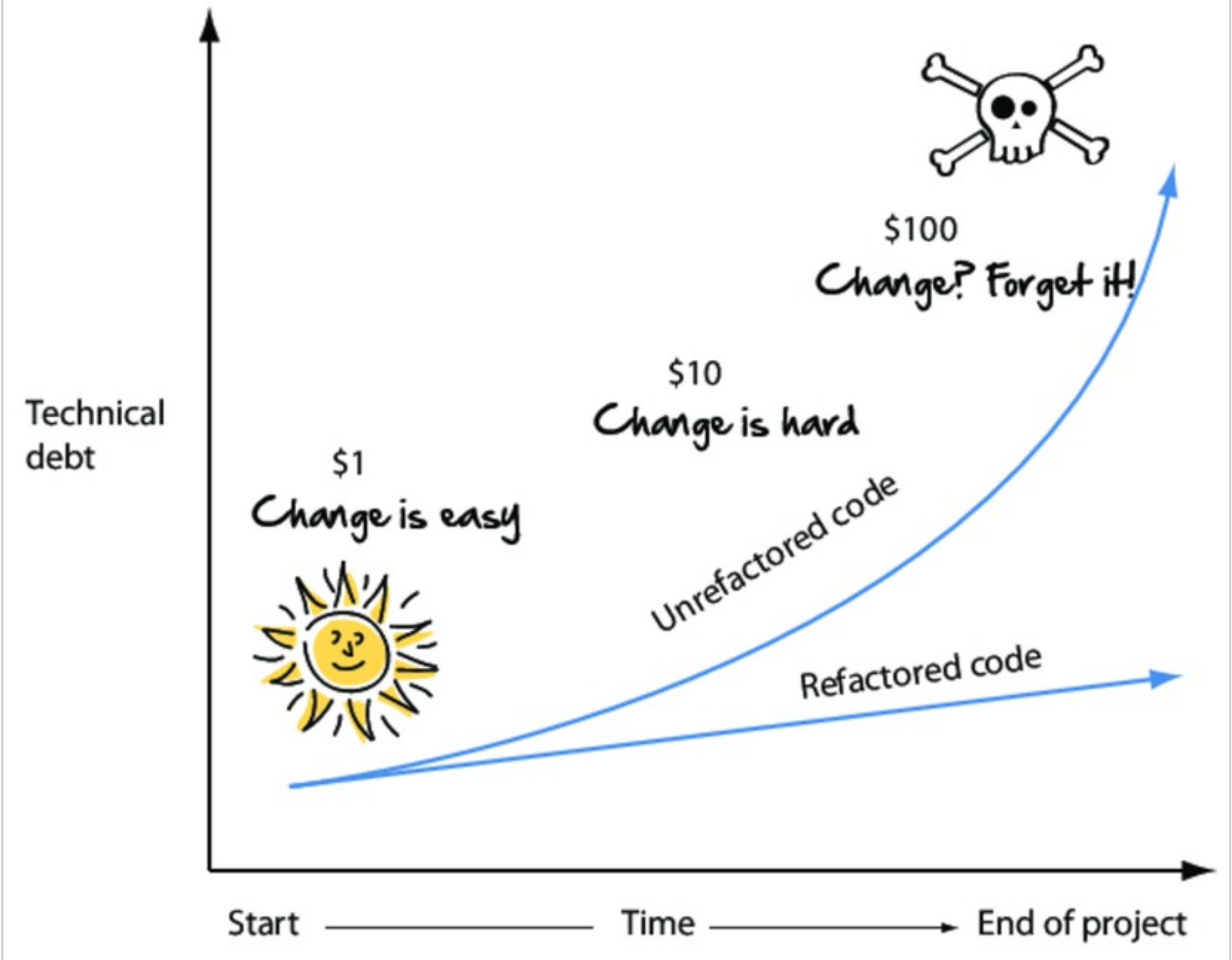
What does this function do?

```
def o(f):
    return f%2==0 if f > 0 else False
```

Documentation and technical debt

Code gets old. Even for the author. Yes, that's you.

- Problem:
  - You're a business that needs to develop software *fast*
  - You have a single Python `.py` file with 2000 lines of code that looks like the above



- Solution:
  - You either close the business, or you make it cheaper to develop new code

Solution part 1/3: Naming conventions

- Give things reasonable names
- Write clear code, even if it takes up more space

What can be improved?

Type this into Mu and press the `Check` button:

```
def o(f):
    return f%2==0 if f>0 else False
```

In [ ]:

```
def is_even(f):
    return f % 2 == 0 if f > 0 else False
```

In [1]:

```
def is_even(number):
    if number > 0:
        number_is_even = number % 2 == 0
        return number_is_even
    else:
        return False
```

Solution 2/3: Write documentation (meta-data)

- Documentation helps to understand what the code does *without reading the code*
- Similar topic: pseudo-code

Looking up documentation 1/2: Python documentation

Search for: Python documentation

Or go to https://docs.python.org/3/ (https://docs.python.org/3/)

Looking up documenation 2/2: `help`

```
print(help(str.find))
```

In [ ]:

```
print(help(is_even))
```

Writing documentation

- What is the purpose?
- What is the input?
- What is the output?

Documenting the purpose

In [2]:

```
def is_even(number):
    """Examines whether a positive number is even.
    Returns `False` if zero or negative
    """
    if number > 0:
        number_is_even = number % 2 == 0
        return number_is_even
    else:
        return False
```

Documenting input:

In [ ]:

```
def is_even(number):
    """Examines whether a positive number is even.
    Returns `False` if zero or negative

    Parameters
    ----------
    number : int
        The number to examine
    """
    if number > 0:
        number_is_even = number % 2 == 0
        return number_is_even
    else:
        return False
```

Documenting output:

In [ ]:

```
def is_even(number):
    """Examines whether a positive number is even.

    Parameters
    ----------
    number : int
        The number to examine

    Returns
    -------
    bool
        True if the number is even and above 0,
        False otherwise
    """
    if number > 0:
        number_is_even = number % 2 == 0
        return number_is_even
    else:
        return False
```

Documenting examples:

In [3]:

```

def is_even(number):
    """Examines whether a positive number is even. False if negative

    Parameters
    ----------
    number : int
        The number to examine

    Returns
    -------
    bool
        True if the number is even and above 0, False otherwise

    Examples
    --------
        Examples should be written in doctest format, and should illustrate h
        to use the function.

        >>> is_even(4)
        True

        >>> is_even(5)
        False
    """
    if number > 0:
        number_is_even = number % 2 == 0
        return number_is_even
    else:
        return False
```

In [ ]:

```
print(help(is_even))
```

In [ ]:

```
print(is_even.__doc__)
```

Another example:

```
def random_number_generator(arg1, arg2):
    """
    Summary line.

    Extended description of function.

    Parameters
    ----------
    arg1 : int
        Description of arg1
    arg2 : str
        Description of arg2

    Returns
    -------
    int
        Description of return value
    """
    return 42
```

Your turn:

Clean up and document this function:

```
def p(b, z):
    return b ** z
```

Summary

- Naming conventions
  - Call things what they are
- Documentation
  - Meta information on functions, modules, methods, and classes