Writing a '__main__' program

When a module is imported, the code in it is executed. For instance, consider the module `palindrome_v1`, which contains two function definitions, followed be 5 lines of code that prompts the user for input, calls `is_palindrome_v1`, and prints the result:

```
def is_palindrome_v1(s):
    """ (str) -> bool

    Return True if and only if s is a palindrome.

    >>> is_palindrome_v1('noon')
    True
    >>> is_palindrome_v1('racecar')
    True
    >>> is_palindrome_v1('dented')
    False
    """

    return reverse(s) == s


def reverse(s):
    """ (str) -> str

    Return a reversed version of s.

    >>> reverse('hello')
    'olleh'
    >>> reverse('a')
    'a'
    """

    rev = ''

    # For each character in s, add that char to the beginning of rev.
    for ch in s:
        rev = ch + rev

    return rev

word = input('Enter a word: ')
if is_palindrome_v1(word):
    print(word, 'is a palindrome.')
else:
    print(word, 'is not a palindrome.')
```

Module `palindrome_v2` imports `palindrome_v1`. When `palindrome_v1` is imported by another module, not only are the two function definitions executed, the last 5 lines of `palindrome_v1` are also executed.

Python's __name__ Variable

Every module has a variable named `__main__`. Since this variable is special (it is built in to Python), its name starts and ends with two underscores. If the function call `print(I am ', __name__)` is included in `palindrome_v1`, then when the module is executed, `I am __main__` will be printed. However, if `palindrome_v1` is imported into `palindrome_v2`, then executing `palindrome_v2`, will print `I am palindrome_v1`.

In other words, `__name__` will refer to `"__main__"` only if it is referenced inside the module being run. In all other cases, `__name__` will refer a string containing the module name.

Using `if __name__ == '__main__':`

We can use an `if` statement to check whether a module is the main one being executed (as opposed to being imported by another module), and only if it is, run certain lines of code.

```
if __name__ == __main__:
    print('This line is being executed because this is the main module being executed.')
```

For the code above, if variable `__name__` does not refer to `"__main__"`, then the code in the body of the `if` statement will not be executed.

Jennifer Campbell • Paul Gries
University of Toronto
