DOCUMENTATION AND TESTING

ERIC MARTIN

Part 1. Standard documentation

1. List of built-in functions and methods

```
>>> dir('__buitins__')
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__'
...
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

>>> [name for name in dir('__buitins__') if not name.startswith('__')]
['capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs',
...
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

2. Help on the built-in module

```
>>> help(__builtins__)
Help on built-in module builtins:

NAME
    builtins - Built-in functions, exceptions, and other objects.
...
FILE
    (built-in)
```

3. Help on a built-in function

```
>>> help('__builtins__'.strip)
Help on built-in function strip:

strip(...) method of builtins.str instance
    S.strip([chars]) -> str

    Return a copy of the string S with leading and trailing
    whitespace removed.
    If chars is given and not None, remove characters in chars instead.
```

Date: Session 2, 2015.

4. List of functions and methods in a class in the built-in module

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__',
...
'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']

>>> [name for name in dir(list) if not name.startswith('__')]
['append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop',
                                                    'remove', 'reverse', 'sort']
```

5. Help on a class in the built-in module

```
>>> help(list)
Help on class list in module builtins:

class list(object)
 |  list() -> new empty list
 |  list(iterable) -> new list initialized from iterable's items
...
 |  sort(...)
 |      L.sort(key=None, reverse=False) -> None -- stable sort *IN PLACE*
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

6. Help on a method in a class in the built-in module

```
>>> help(list.pop)
Help on method_descriptor:

pop(...)
    L.pop([index]) -> item -- remove and return item at index (default last).
    Raises IndexError if list is empty or index is out of range.
```

7. List of names in a module

```
>>> import math
>>> dir(math)
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
...
'log2', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
>>> [name for name in dir(math) if not name.startswith('__')]
['acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos',
...
'log2', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

8. Help on a module

```
>>> import math
>>> help(math)
Help on module math:

NAME
    math
...
DATA
    e = 2.718281828459045
    pi = 3.141592653589793

FILE
    /Library/Frameworks/Python.framework/Versions/3.4/lib/python3.4/lib-dynload/math.so
```

9. Help on a name in a module

```
>>> import math
>>> help(math.log2)
Help on built-in function log2 in module math:

log2(...)
    log2(x)

    Return the base 2 logarithm of x.

>>> from math import trunc
>>> help(trunc)
Help on built-in function trunc in module math:

trunc(...)
    trunc(x:Real) -> Integral

    Truncates x to the nearest Integral toward 0. Uses the __trunc__ magic method.
```

Part 2. Documentation from modules we create

10. List of names in a module

```
>>> import bit_set
>>> dir(bit_set)
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
...
'cardinality', 'display_encoded_set', 'is_in_encoded_set', 'set_encoding']
>>> [name for name in dir(bit_set) if not name.startswith('__')]
['cardinality', 'display_encoded_set', 'is_in_encoded_set', 'set_encoding']
```

11. Help on a module thanks to doctrings

```
>>> import bit_set
>>> help(bit_set)
Help on module bit_set:

NAME
    bit_set

DESCRIPTION
    Performs operations on encodings of a set of (distinct)
    nonnegative integers {n_1, ..., n_k} as 2^{n_1} + ... + 2^{n_k}.

FUNCTIONS
    cardinality(encoded_set)
        Returns the number of elements in the set
        encoding as the argument.

        >>> cardinality(0)
        0
        >>> cardinality(1)
        1
        >>> cardinality(76)
        3

    display_encoded_set(encoded_set)
        Displays the members of the set encoded by the argument,
        from smallest to largest element, in increasing order.

        >>> display_encoded_set(0)
        {}
        >>> display_encoded_set(1)
        {0}
...
FILE
    /Users/emartin/Documents/COMP9021/Lectures/Lecture_2/bit_set.py
```

12. Help on a name in a module thanks to doctrings

```
>>> import bit_set
>>> help(bit_set.set_encoding)
Help on function set_encoding in module bit_set:

set_encoding(set_of_nonnegative_integers)
    Encodes a set and returns the encoding.
    Here an empty set of curly braces provided as argument
    denotes the empty set, not the empty dictionary.

    >>> set_encoding({})
    0
    >>> set_encoding({0})
    1
    >>> set_encoding({0, 1})
    3
    >>> set_encoding({2, 3, 6})
    76

>>> from bit_set import display_encoded_set
>>> help(display_encoded_set)
Help on function display_encoded_set in module bit_set:

display_encoded_set(encoded_set)
    Displays the members of the set encoded by the argument,
    from smallest to largest element, in increasing order.

    >>> display_encoded_set(0)
    {}
    >>> display_encoded_set(1)
    {0}
    >>> display_encoded_set(3)
    {0, 1}
    >>> display_encoded_set(76)
    {2, 3, 6}
```

Part 3. Unit testing of functions from modules we create

Made possible thanks to:

```
if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

13. Passing all tests

```
$ python3 bit_set.py
$ python3 bit_set.py -v
Trying:
    cardinality(0)
Expecting:
    0
ok
Trying:
    cardinality(1)
Expecting:
    1
ok
Trying:
    cardinality(76)
Expecting:
    3
ok
Trying:
    display_encoded_set(0)
Expecting:
    {}
ok
Trying:
    display_encoded_set(1)
Expecting:
    {0}
ok
...
1 items had no tests:
    __main__
4 items passed all tests:
   3 tests in __main__.cardinality
   4 tests in __main__.display_encoded_set
   4 tests in __main__.is_in_encoded_set
   4 tests in __main__.set_encoding
15 tests in 5 items.
15 passed and 0 failed.
Test passed.
```

14. Failing some tests

Changing

```
    >>> display_encoded_set(1)
    {0}
```

to

```
    >>> display_encoded_set(1)
    {1}
```

in file.

```
$ python3 bit_set.py
**********************************************************************
File "bit_set.py", line 17, in __main__.display_encoded_set
Failed example:
    display_encoded_set(1)
Expected:
    {1}
Got:
    {0}
**********************************************************************
1 items had failures:
   1 of   4 in __main__.display_encoded_set
***Test Failed*** 1 failures.
```