No examples required on term test #1

Short Python function/method descriptions:

```python
__builtins__:
    len(x) -> integer
        Return the length of the list, tuple, dict, or string x.
    max(L) -> value
        Return the largest value in L.
    min(L) -> value
        Return the smallest value in L.
    range([start], stop, [step]) -> list of integers
        Return a list containing the integers starting with start and
        ending with stop - 1 with step specifying the amount to increment
        (or decrement). If start is not specified, the list starts at 0.
        If step is not specified, the values are incremented by 1.
    sum(L) -> number
        Returns the sum of the numbers in L.

dict:
    D[k] -> value
        Return the value associated with the key k in D.
    k in d -> boolean
        Return True if k is a key in D and False otherwise.
    D.get(k) -> value
        Return D[k] if k in D, otherwise return None.
    D.keys() -> list of keys
        Return the keys of D.
    D.values() -> list of values
        Return the values associated with the keys of D.
    D.items() -> list of (key, value) pairs
        Return the (key, value) pairs of D, as 2-tuples.

float:
    float(x) -> floating point number
        Convert a string or number to a floating point number, if
        possible.

int:
    int(x) -> integer
        Convert a string or number to an integer, if possible. A floating
        point argument will be truncated towards zero.

list:
    x in L -> boolean
        Return True if x is in L and False otherwise.
    L.append(x)
        Append x to the end of list L.
    L1.extend(L2)
        Append the items in list L2 to the end of list L1.
    L.index(value) -> integer
        Return the lowest index of value in L.
```
L.insert(index, x)
    Insert x at position index.
L.pop()
    Remove and return the last item from L.
L.remove(value)
    Remove the first occurrence of value from L.
L.sort()
    Sort the list in ascending order.

Module random:
    randint(a, b)
        Return random integer in range [a, b], including both end points.

str:
    x in s -> boolean
        Return True if x is in s and False otherwise.
str(x) -> string
    Convert an object into its string representation, if possible.
S.count(sub[, start[, end]]) -> int
    Return the number of non-overlapping occurrences of substring sub
    in string S[start:end]. Optional arguments start and end are
    interpreted as in slice notation.
S.find(sub[,i]) -> integer
    Return the lowest index in S (starting at S[i], if i is given)
    where the string sub is found or -1 if sub does not occur in S.
S.split([sep]) -> list of strings
    Return a list of the words in S, using string sep as the separator
    and any whitespace string if sep is not specified.

set:
    {1, 2, 3, 1, 3} -> {1, 2, 3}
s.add(...)
    Add an element to a set
set()
    Create a new empty set object
x in s
    True iff x is an element of s

list comprehension:
    [expression with x for x in <list or other iterable>]

functional if:
    <expression 1> if <boolean condition> else <expression 2>
    -> <expression 1> if the boolean condition is True,
    otherwise <expression 2>