Problem Wk.7.2.7: optOverLine

Given a function *f(x)*, how can we find a value *x** such that *f(x*) ≤ f(x)* for all *x*? If *f* is differentiable, then we can do this relatively easily by taking the derivative, setting it to 0 and solving for *x*. This gets tricky when the *f* is complicated and there may be multiple minima, and when we wish to extend to functions with multiple arguments. For functions that aren't differentiable (such as those involving *max* or *abs*), there is no straightforward mathematical approach at all. In one dimension, if we know a range of values of *x* that is likely to contain the minimum, we can plausibly sample different values of *x*, evaluate *f* at each of them, and return the sampled *x* for which *f(x)* is minimized.

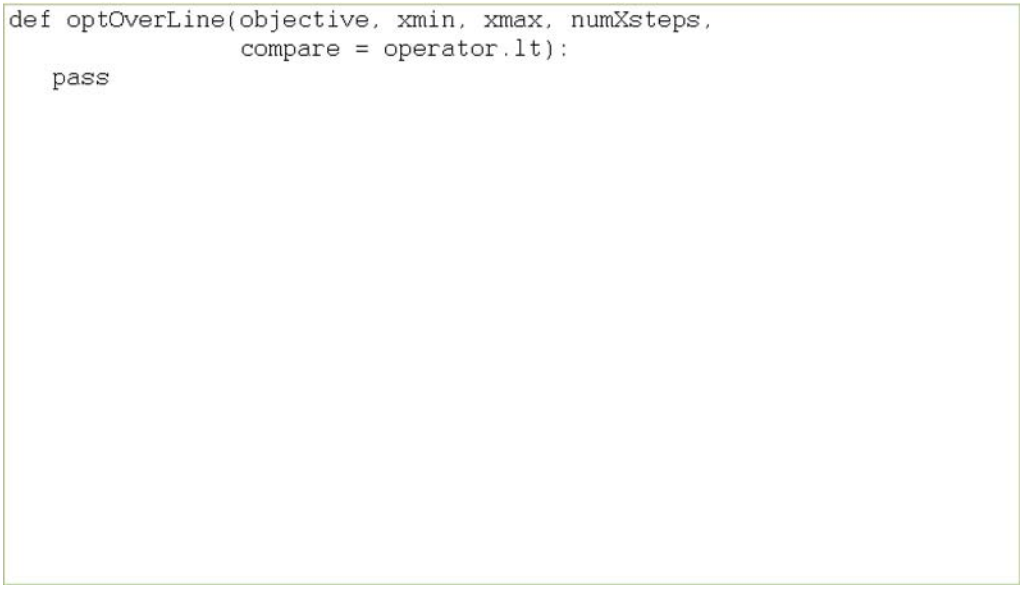
Implement the `optOverLine` function, which takes:

- `objective`: a function of one argument,
- `xmin, xmax`: a range of values for the argument,
- `numXsteps`: how many points to test within the range (starting with xmin to just below xmax),
- `compare`: an optional comparison function that defaults to be `operator.lt`, which is the less than, `<` , operator.

It should return a tuple `(bestObjValue, bestX)` with the best value of the objective and the x value that corresponds to it.

All of `xmin, xmax, numXSteps` could be integers.

The functions `floatRange` and `argopt` (from earlier problems) are already defined. You should use them in your solution.



```
def optOverLine(objective, xmin, xmax, numXsteps,
                compare = operator.lt):
   pass
```