Notes.py

1 Sequential Search

```
def sequential_search_itr(xs, y):
    '''
    Check whether y is contained in the list xs

    >>> sequential_search_itr([1, 3, 5, 4, 2, 0], 2)
    True
    >>> sequential_search_itr([1, 3, 5, 4, 2, 0], 6)
    False
    '''
    for x in xs:
        if x == y:
            return True
    return False
```

```
def sequential_search_itr2(xs, y):
    '''
    Check whether y is contained in the list xs

    >>> sequential_search_itr2([1, 3, 5, 4, 2, 0], 2)
    True
    >>> sequential_search_itr2([1, 3, 5, 4, 2, 0], 6)
    False
    '''
    for i in range(len(xs)):
        if xs[i] == y:
            return True
    return False
```

```
def sequential_search_rec(xs, y):
    '''
    Check whether y is contained in the list xs

    >>> sequential_search_rec([1, 3, 5, 4, 2, 0], 2)
    True
    >>> sequential_search_rec([1, 3, 5, 4, 2, 0], 6)
    False
    '''
    if len(xs) == 0:
        return False
    if xs[0] == y:
        return True
    return sequential_search_rec(xs[1:], y)
```

```
def sequential_search_rec2(xs, y):
    '''
    Check whether y is contained in the list xs

    >>> sequential_search_rec2([1, 3, 5, 4, 2, 0], 2)
    True
    >>> sequential_search_rec2([1, 3, 5, 4, 2, 0], 6)
    False
    '''
    def go(xs):
        if len(xs) == 0:
            return False
        if xs[0] == y:
            return True
        return go(xs[1:])
    return go(xs)
```

```
def sequential_search_rec3(xs, y):
    '''
    Check whether y is contained in the list xs

    >>> sequential_search_rec3([1, 3, 5, 4, 2, 0], 2)
    True
    >>> sequential_search_rec3([1, 3, 5, 4, 2, 0], 6)
    False
    '''
    def go(i):
        if i == len(xs):
            return False
        if xs[i] == y:
            return True
        return go(i+1)
    return go(0)
```

2 Binary Search

```
def binary_search_itr(xs, y):
    '''
    Assume that xs is a list of numbers sorted from LOWEST to HIGHEST.
    Return True if y is in the list xs.

    >>> binary_search_itr([1, 3, 5, 7, 9, 11], 9)
    True
    >>> binary_search_itr([1, 3, 5, 7, 9, 11], 8)
    False
    >>> binary_search_itr(list(range(-1001, 1001, 2)), 9)
    True
    >>> binary_search_itr(list(range(-1000, 1000, 2)), 9)
    False
    '''
    if len(xs) == 0:
        return False
    left = 0
    right = len(xs) - 1

    while left != right:
        mid = (left + right) // 2
        if xs[mid] > y:
            right = mid
        if xs[mid] < y:
            left = mid + 1
        if xs[mid] == y:
            return True
            left = mid + 1

    if xs[left] == y:
        return True
    else:
        return False

    return go(0, len(xs) - 1)
```

```
def binary_search_rec(xs, y):
    '''
    Assume that xs is a list of numbers sorted from LOWEST to HIGHEST.
    Return True if y is in the list xs.

    >>> binary_search_rec([1, 3, 5, 7, 9, 11], 9)
    True
    >>> binary_search_rec([1, 3, 5, 7, 9, 11], 8)
    False
    >>> binary_search_rec(list(range(-1001, 1001, 2)), 9)
    True
    >>> binary_search_rec(list(range(-1000, 1000, 2)), 9)
    False
    '''
    if len(xs) == 0:
        return False

    def go(left, right):
        if left == right:
            if xs[left] == y:
                return True
            else:
                return False
        mid = (left + right) // 2
        if xs[mid] > y:
            right = mid
        if xs[mid] < y:
            left = mid + 1
        if xs[mid] == y:
            return True
            left = mid + 1
        return go(left, right)

    return go(0, len(xs) - 1)
```

```
def binary_search_rec2(xs, y):
    '''
    Assume that xs is a list of numbers sorted from LOWEST to HIGHEST.
    Return True if y is in the list xs.

    >>> binary_search_rec2([1, 3, 5, 7, 9, 11], 9)
    True
    >>> binary_search_rec2([1, 3, 5, 7, 9, 11], 8)
    False
    >>> binary_search_rec2(list(range(-1001, 1001, 2)), 9)
    True
    >>> binary_search_rec2(list(range(-1000, 1000, 2)), 9)
    False
    '''
    if len(xs) == 0:
        return False

    mid = len(xs) // 2
    if xs[mid] > y:
        return binary_search_rec2(xs[:mid], y)
    if xs[mid] < y:
        return binary_search_rec2(xs[mid+1:], y)
    if xs[mid] == y:
        return True
```