Finding an item in a sequence

```python
def isIn(seq, t):
    # (sequence, item) -> boolean
    Search for item n in a sequence, seq.
    Return True if n is a member, else False.
    >>> isIn('hello', 'Y')
    False
    >>> isIn([10, 20, 30, 40, 50, 60, 70, 80, 90], 80)
    True
    ...  # developing code
    return #Boolean value
```

Finding an item in a sequence

```python
def isIn(seq, t):
    # (sequence, item) -> boolean
    Search for item n in a sequence, seq.
    Return True if n is a member, else False.
    >>> isIn('hello', 'Y')
    False
    >>> isIn([10, 20, 30, 40, 50, 60, 70, 80, 90], 80)
    True
    ...  # developing code
    return #Boolean value
```

Finding an item in a sequence

```python
def isIn(seq, t):
    # (sequence, item) -> boolean
    Search for item n in a sequence, seq.
    Return True if n is a member, else False.
    >>> isIn('hello', 'Y')
    False
    >>> isIn([10, 20, 30, 40, 50, 60, 70, 80, 90], 80)
    True
    ...  # developing code
    return #Boolean value
```

Finding an item in a sequence

```python
def isIn(seq, t):
    # (sequence, item) -> boolean
    Search for item n in a sequence, seq.
    Return True if n is a member, else False.
    >>> isIn('hello', 'Y')
    False
    >>> isIn([10, 20, 30, 40, 50, 60, 70, 80, 90], 80)
    True
    ...  # developing code
    return #Boolean value
```

Finding an item in a sequence

```python
def isIn(seq, t):
    # (sequence, item) -> boolean
    Search for item n in a sequence, seq.
    Return True if n is a member, else False.
    >>> isIn('hello', 'Y')
    False
    >>> isIn([10, 20, 30, 40, 50, 60, 70, 80, 90], 80)
    True
    ...  # developing code
    return #Boolean value
```

Finding an item in a sequence

```python
def isIn(seq, t):
    # (sequence, item) -> boolean
    Search for item n in a sequence, seq.
    Return True if n is a member, else False.
    >>> isIn('hello', 'Y')
    False
    >>> isIn([10, 20, 30, 40, 50, 60, 70, 80, 90], 80)
    True
    ...  # developing code
    return #Boolean value
```

Finding an item in a sequence

```python
def isIn(seq, t):
    # (sequence, item) -> boolean
    Search for item n in a sequence, seq.
    Return True if n is a member, else False.
    >>> isIn('hello', 'Y')
    False
    >>> isIn([10, 20, 30, 40, 50, 60, 70, 80, 90], 80)
    True
    ...  # developing code
    return #Boolean value
```

Finding an item in a sequence

```python
def isIn(seq, t):
    # (sequence, item) -> boolean
    Search for item n in a sequence, seq.
    Return True if n is a member, else False.
    >>> isIn('hello', 'Y')
    False
    >>> isIn([10, 20, 30, 40, 50, 60, 70, 80, 90], 80)
    True
    ...  # developing code
    return #Boolean value
```
```python
CIS 210
Finding an item in a sequence

def isIn(seq, t):
    # (sequence, item) -> boolean
    Search for item n in a sequence, seq.
    Return True if n is a member, else False.
    >>> isIn('hello', 'y')
    False
    >>> isIn((10, 20, 30, 40, 50, 60, 70, 80, 90), 80)
    True
    
    for item in seq:
        if item == t:
            found = True
        else:
            found = False
    # developing code
    return found

return False
```

```python
CIS 210
Finding an item in a sequence

def isIn(seq, t):
    # (sequence, item) -> boolean
    Search for item n in a sequence, seq.
    Return True if n is a member, else False.
    >>> isIn('hello', 'y')
    False
    >>> isIn((10, 20, 30, 40, 50, 60, 70, 80, 90), 80)
    True
    
    for item in seq:
        if item == t:
            found = True
        else:
            found = False
    # developing code
    return found

return False
```

```python
CIS 210
Finding an item in a sequence

def isIn(seq, t):
    # (sequence, item) -> boolean
    Search for item n in a sequence, seq.
    Return True if n is a member, else False.
    >>> isIn('hello', 'y')
    False
    >>> isIn((10, 20, 30, 40, 50, 60, 70, 80, 90), 80)
    True
    
    for item in seq:
        if item == t:
            return True
    if item > t:
        return False
    return False
```

```python
CIS 210
Finding an item in a sequence

def isIn(seq, t):
    # (sequence, item) -> boolean
    Search for item n in a sorted sequence, seq.
    Return True if n is a member, else False.
    >>> isIn('hello', 'y')
    False
    >>> isIn((10, 20, 30, 40, 50, 60, 70, 80, 90), 80)
    True
    
    for item in seq:
        if item == t:
            return True
        if item > t:
            return False
    return False
```

```python
CIS 210

Binary Search: efficient search technique, as long as the list is already sorted.

Each step divides the remaining data into equal parts and discards one part:

If remaining part is empty, then done (not found).

Go to mid-point of remaining part and compare to target.

If mid-point is the target, then done (found).

Otherwise, keep the part of the list where n could be, and search that. (Discard the rest.)
```
Each step divides the remaining data into equal parts and discards one part:
If remaining part is empty, then done (not found).
Go to mid-point of remaining part and compare to target.
If mid-point is the target, then done (found).
Otherwise, keep the part of the list where n could be, and search that. (Discard the rest.)

For example,

nums = (1, 3, 4, 6, 8, 9, 11)
target value: 4

For example,

nums = (1, 3, 4, 6, 8, 9, 11, 13, 15, 25, 99, 100, 102)
target value: 42

Sequential search \( O(n) \)
Binary search \( O(\log n) \)

When \( n = 150 \) ...

When \( n = 1,000 \) ...

When \( n \) is around \( 1,000,000 \) ...

“Big O” notation for general, worst-case scenario
Check even or odd \( O(1) \)
Binary search \( O(\log n) \)
Sequential search \( O(n) \)
“Long” multiplication \( O(n^2) \)
Match passwords \( O(2^n) \)
CIS 210
List Comprehensions (declarative style programming)

Given:

$S = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$

Generate:

$T$ is a list of $x$ such that $x$ is a member of $S$ and $x$ is even

$T = [x \text{ for } x \text{ in } S \text{ if } \text{even}(x)]$

#declarative (filter pattern)

CIS 210
Python/programming toolkit so far


CIS 210
Programming/Software Engineering/Computer Science Concepts

Executing and revising functions – hand-tracing program execution; hand-tracing function execution; program function diagrams; refactoring functions

Algorithms – iterative and recursive algorithms; accumulator pattern; Monte Carlo algorithms; binary representation and conversion; encryption and decryption; data analysis and data mining (k-means cluster analysis); binary search (and big-O notation)

CIS 210
CIS 210 Learning Outcomes

• understand, develop, implement algorithms for computational problem solving;
• use structured design and testing methods to develop and implement programs;
• read, write, revise, document, test, and debug code;
• demonstrate robust mental models of data representation and code execution;
• demonstrate good understanding of a high level programming language;
• introduce and/or implement a sampling of classic computer science problem domains and algorithms.