Algorithm Implementation: Searching in Python

Lesson Objectives

At the end of this lesson, you should be able to:

- Use index method in Python
- Explain the importance of coding searching algorithms in Python
- Code linear search in Python
- Code binary search in Python
- Identify other search algorithms written in Python
- Apply your knowledge and understanding of searching algorithms to your problem solving in Python

Topic Outline

Index Method in Python

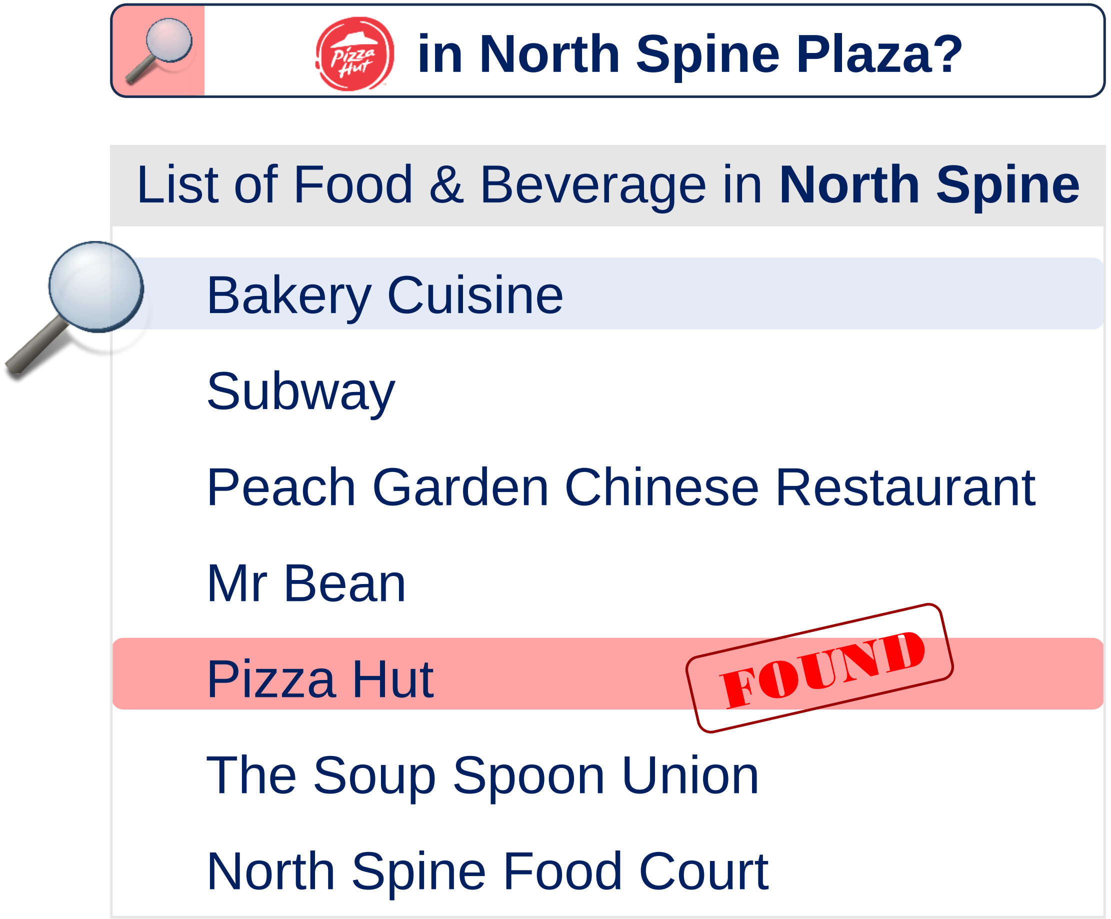
Linear Search

Binary Search

Comparison of Searching Algorithms

Linear Search: Recall

- Iterates over the sequence, one item at a time, until the specific item is found or all items have been examined
  - The element that needs to be found is called a search key
- Linear search/ sequential search
  - Intuitive approach
  - Starts at the first item
  - Is it the one I am looking for?
  - If not, goes to next item
  - Repeats until found or all the items are checked
- This approach is necessary if items are not sorted



in North Spine Plaza?

List of Food & Beverage in **North Spine**

Bakery Cuisine

Subway

Peach Garden Chinese Restaurant

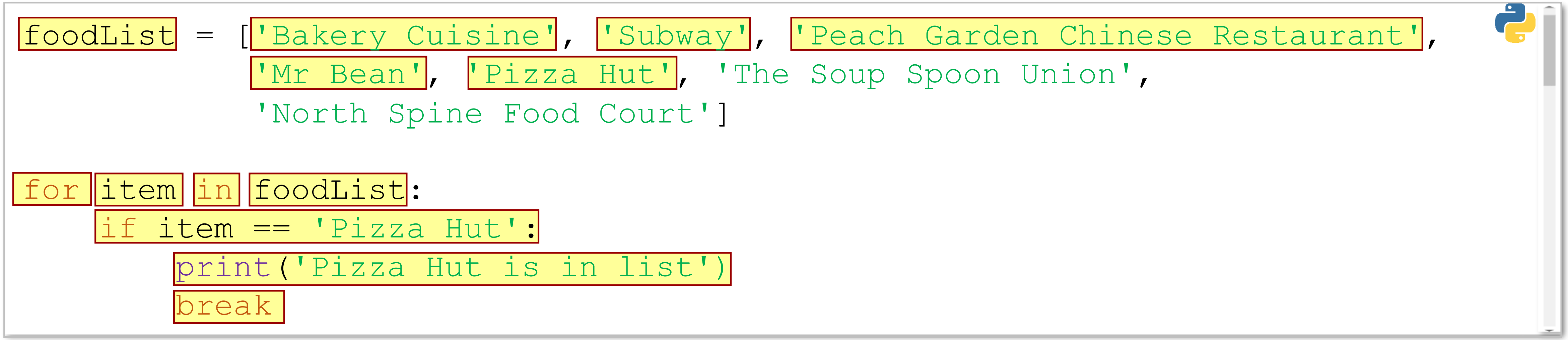
Mr Bean

Pizza Hut

The Soup Spoon Union

North Spine Food Court

Linear Search in Python



```python
foodList = ['Bakery Cuisine', 'Subway', 'Peach Garden Chinese Restaurant',
            'Mr Bean', 'Pizza Hut', 'The Soup Spoon Union',
            'North Spine Food Court']

for item in foodList:
    if item == 'Pizza Hut':
        print('Pizza Hut is in list')
        break
```

Python List `index()` Method

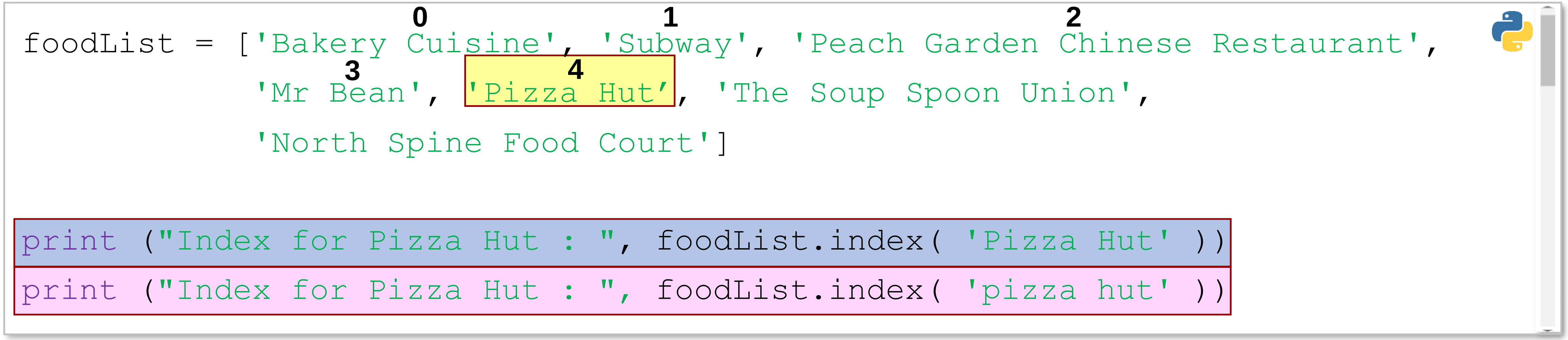
Description

The method **`index()`** returns the lowest index in list that ***obj*** appears.

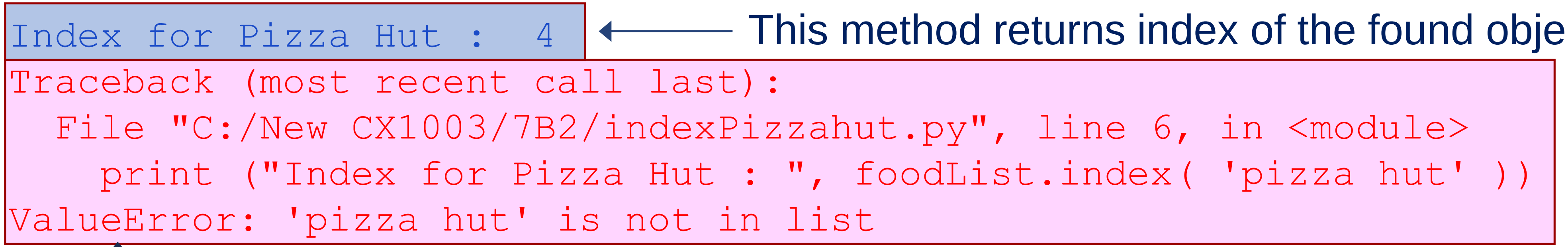
The **index** method does a linear search and stops at the first matching item.

If no matching item is found, it raises a **ValueError** exception.

Python List `index()` Method: Example



```python
foodList = ['Bakery Cuisine', 'Subway', 'Peach Garden Chinese Restaurant',
            'Mr Bean', 'Pizza Hut', 'The Soup Spoon Union',
            'North Spine Food Court']


print ("Index for Pizza Hut : ", foodList.index( 'Pizza Hut' ))
print ("Index for Pizza Hut : ", foodList.index( 'pizza hut' ))
```

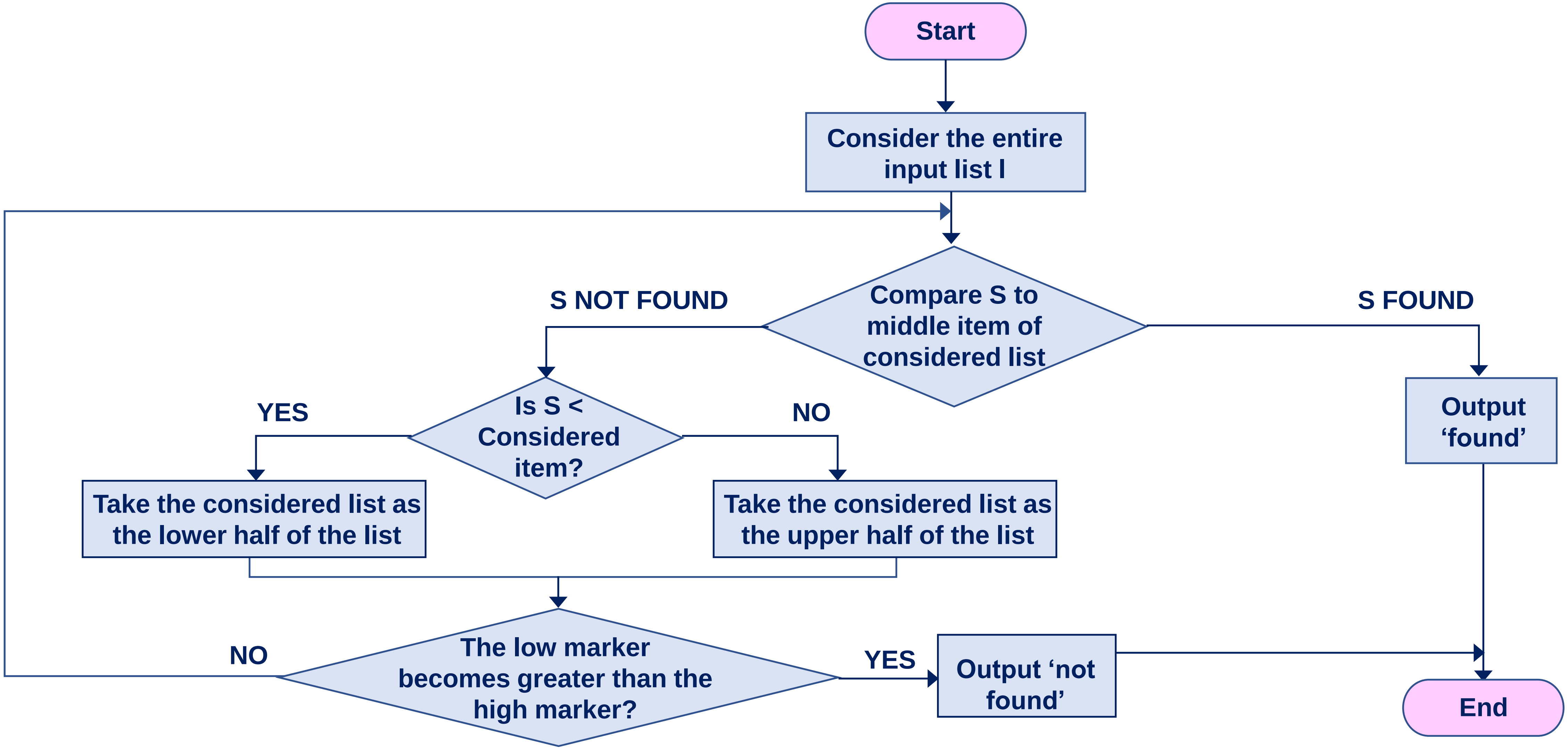


```
Index for Pizza Hut :  4
Traceback (most recent call last):
  File "C:/New CX1003/7B2/indexPizzahut.py", line 6, in <module>
    print ("Index for Pizza Hut : ", foodList.index( 'pizza hut' ))
ValueError: 'pizza hut' is not in list
```

This method returns index of the found object.

Otherwise, it raises an exception indicating that the value is not found.

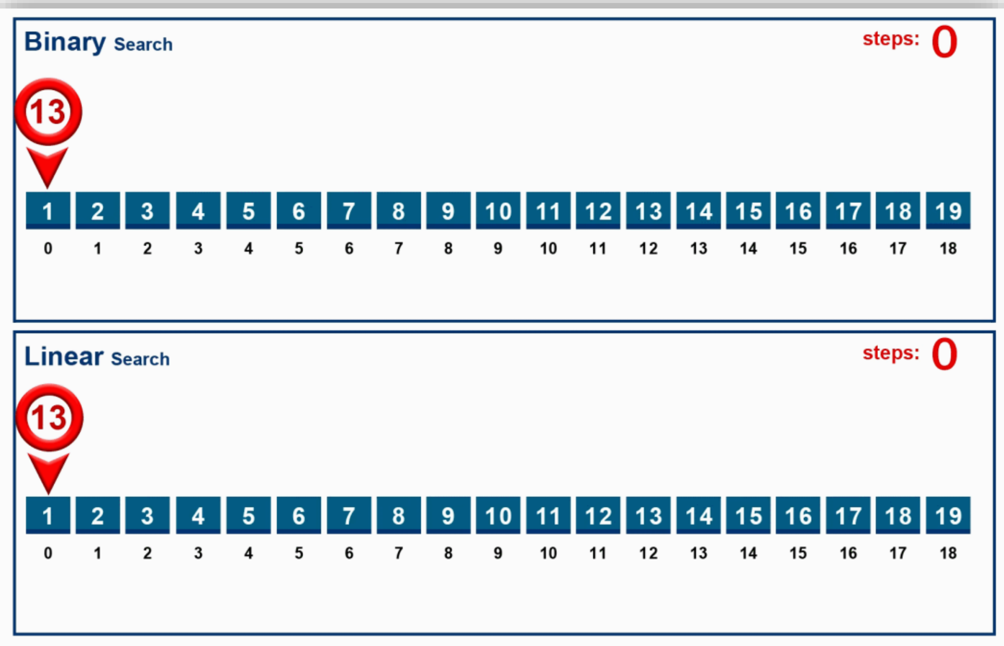
Binary Search Flowchart



Start
Consider the entire input list l
Compare S to middle item of considered list
S NOT FOUND
S FOUND
Output 'found'
Is S < Considered item?
YES
NO
Take the considered list as the lower half of the list
Take the considered list as the upper half of the list
The low marker becomes greater than the high marker?
NO
YES
Output 'not found'
End

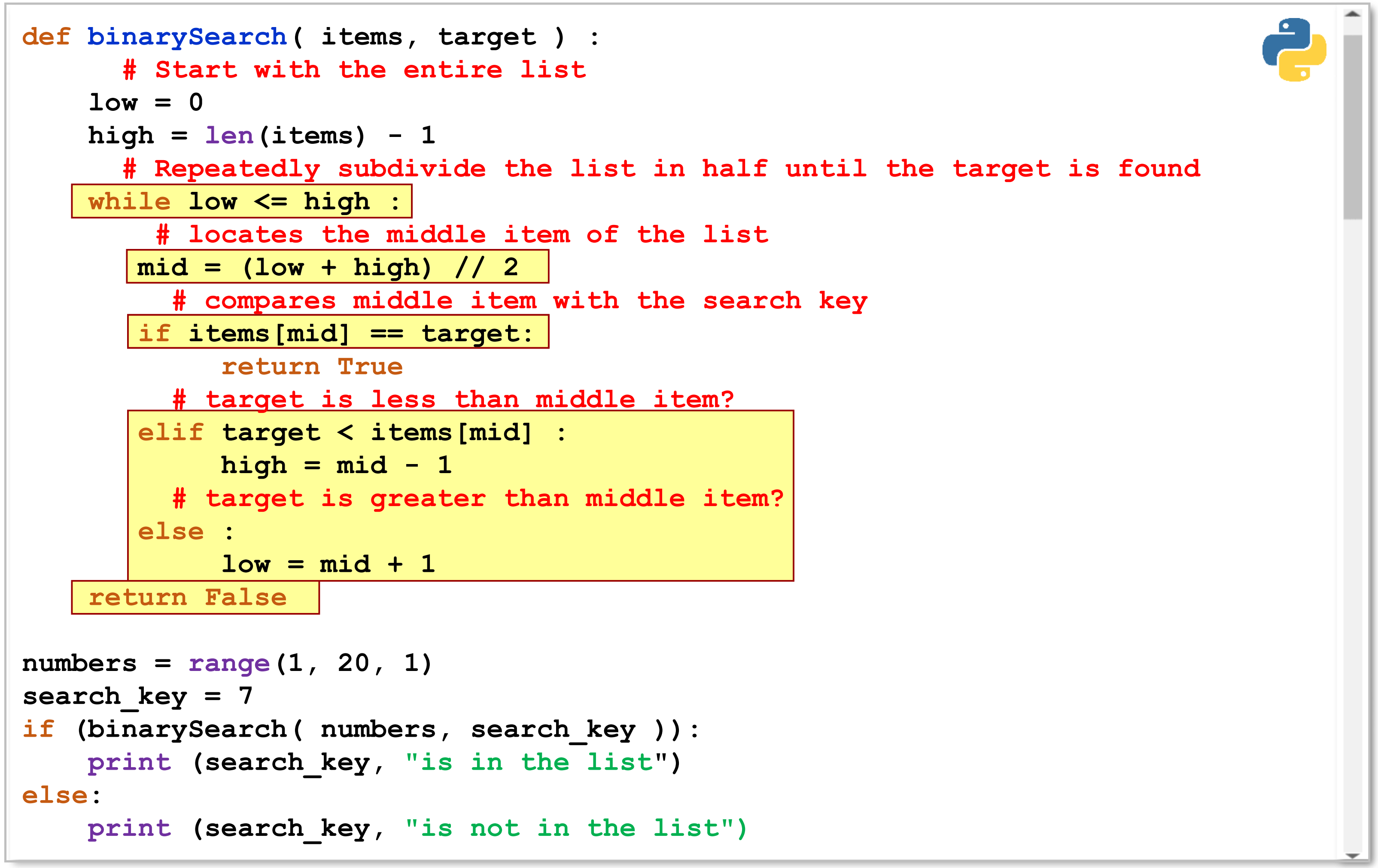
Linear Search vs. Binary Search

Variance of Hi-Low Number Guessing Game



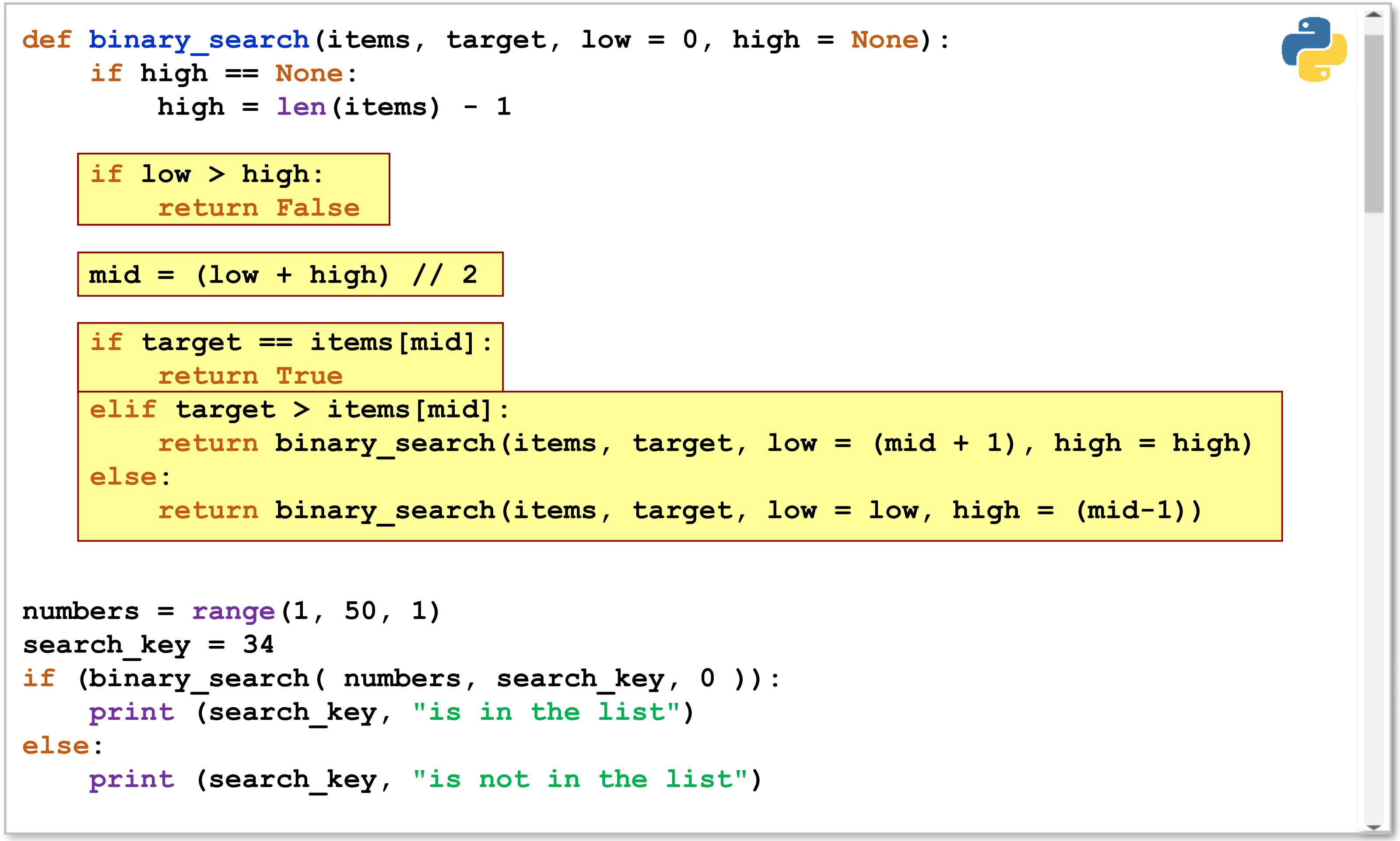
Algorithm	Best Time Complexity	Average Time Complexity	Worst Time Complexity	Worst Space Complexity
Linear Search	O(1)	O(n)	O(n)	O(1)
Binary Search	O(1)	O(log n)	O(log n)	O(1)

Iterative Binary Search



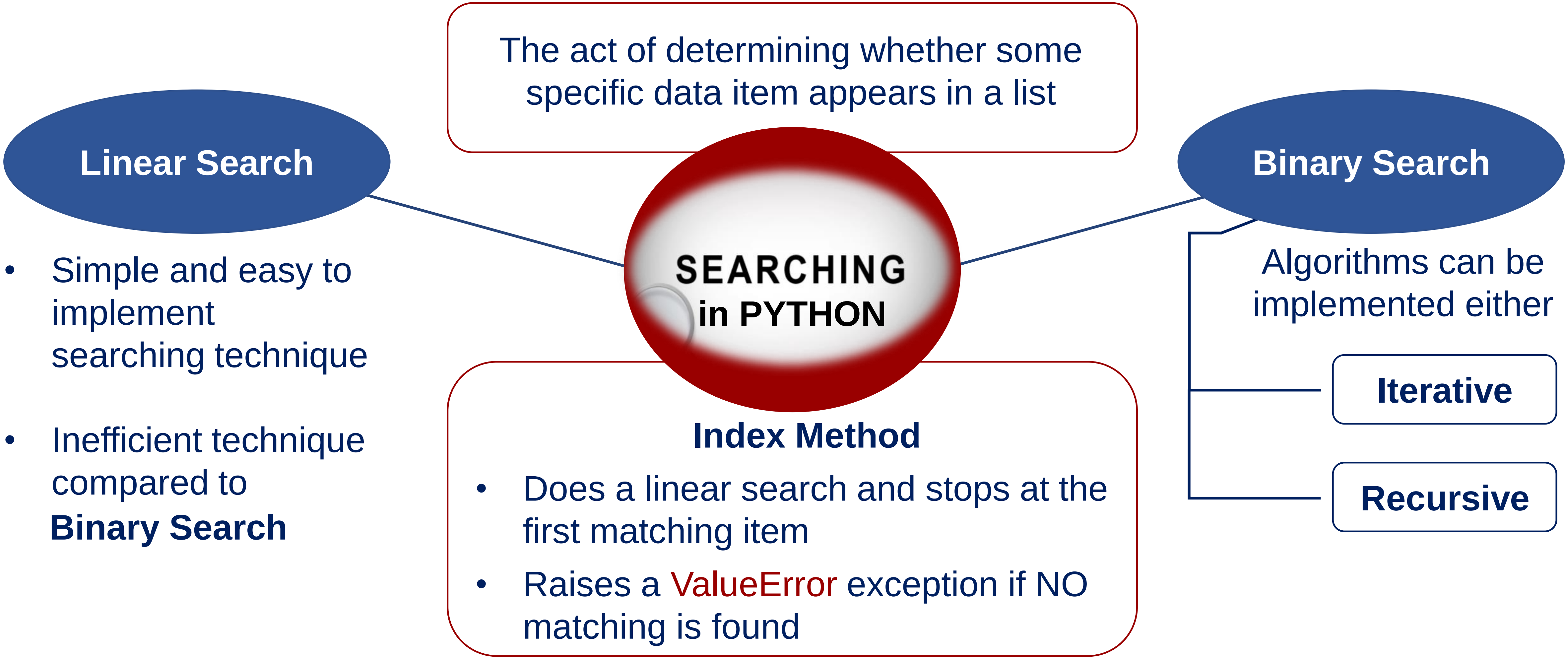
```
def binarySearch( items, target ) :
      # Start with the entire list
    low = 0
    high = len(items) - 1
      # Repeatedly subdivide the list in half until the target is found
    while low <= high :
        # locates the middle item of the list
       mid = (low + high) // 2
         # compares middle item with the search key
       if items[mid] == target:
            return True
         # target is less than middle item?
       elif target < items[mid] :
            high = mid - 1
         # target is greater than middle item?
       else :
            low = mid + 1
    return False

numbers = range(1, 20, 1)
search_key = 7
if (binarySearch( numbers, search_key )):
    print (search_key, "is in the list")
else:
    print (search_key, "is not in the list")
```

Recursive Binary Search



```python
def binary_search(items, target, low = 0, high = None):
    if high == None:
        high = len(items) - 1

    if low > high:
        return False

    mid = (low + high) // 2

    if target == items[mid]:
        return True
    elif target > items[mid]:
        return binary_search(items, target, low = (mid + 1), high = high)
    else:
        return binary_search(items, target, low = low, high = (mid-1))


numbers = range(1, 50, 1)
search_key = 34
if (binary_search( numbers, search_key, 0 )):
    print (search_key, "is in the list")
else:
    print (search_key, "is not in the list")
```

Summary



The act of determining whether some specific data item appears in a list

SEARCHING
in PYTHON

Linear Search

- Simple and easy to implement searching technique
- Inefficient technique compared to **Binary Search**

Index Method

- Does a linear search and stops at the first matching item
- Raises a ValueError exception if NO matching is found

Binary Search

Algorithms can be implemented either

Iterative

Recursive

References for Images

No.	Slide No.	Image	Reference
1	5		By Source, Fair use, retrieved July 16, 2018 from https://en.wikipedia.org/w/index.php?curid=22312809.
2	5		Magnifying Glass [Online Image]. Retrieved July 16, 2018 from http://www.publicdomainfiles.com/show_file.php?id=13534684215801.
3	All pages with Python codes		Python Logo [Online Image]. Retrieved April 18, 2018 from https://pixabay.com/en/language-logo-python-2024210/.