Algorithms in Python Programming

Week 11: Alogirthms and Pseudocode

Learning objectives

- ❑ To know algorithms and Pseudocode
- ❑ To know recursion in Python programming
- ❑ To explore programming language processors:
- ❑ Compiler Vs Interpreter

At the conclusion of this lecture, students will be able to understand the role of alogrithms in programming and writing simple algorithms by using Python.

Algorithms

These are well defined instructions that mainly used to solve some intended problems or to perform a specific operations.

Example: searching, sorting, calculations, auotmatic decision making and more.

Like many other programming languages Python has inbuilt algorithms to perform afore noted operations.

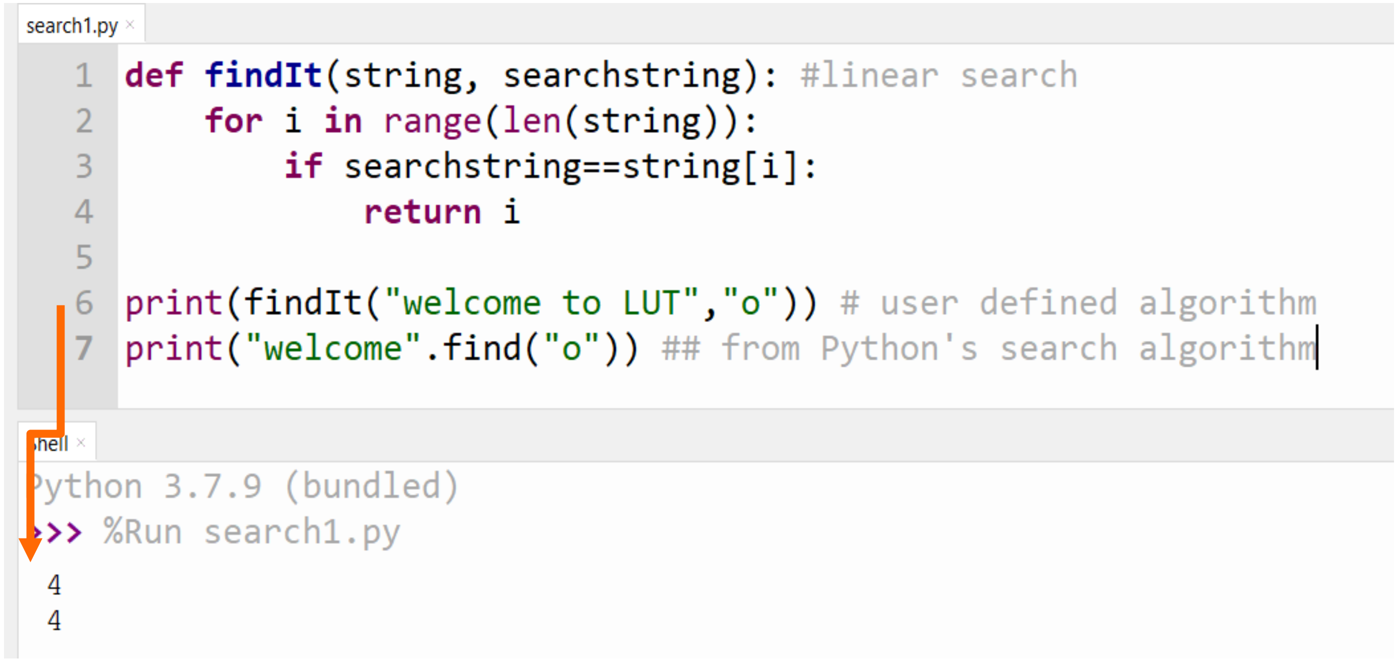
Example: **find ()** – to search for a string
sort () – to sort data in ascending or descending order
sqrt() - returns the square root value of a number
min()/max() – to find the smallest or biggest
plot() – to plot a graph

In this lecture, we discuss few familiar searching and sorting algorithms.

Searching algorithms in Python

It is a technique of selecting specific data from a collection of data based on some condition.

Well, let's analyse how string ->find() function works in Python?



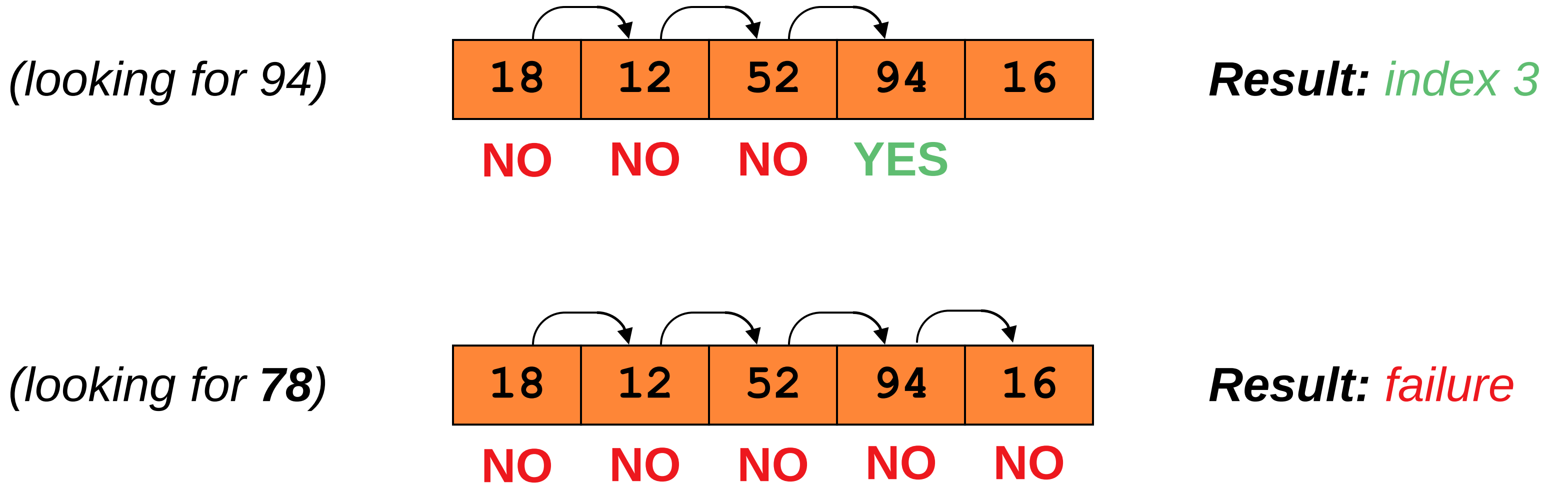
```python
def findIt(string, searchstring): #linear search
    for i in range(len(string)):
        if searchstring==string[i]:
            return i

print(findIt("welcome to LUT","o")) # user defined algorithm
print("welcome".find("o")) ## from Python's search algorithm
```

```
Python 3.7.9 (bundled)
>>> %Run search1.py

 4
 4
```

Here **findIt()** used **linear search** technique to find the occurance of the search element (”o” here for example) .

Linear/sequential search

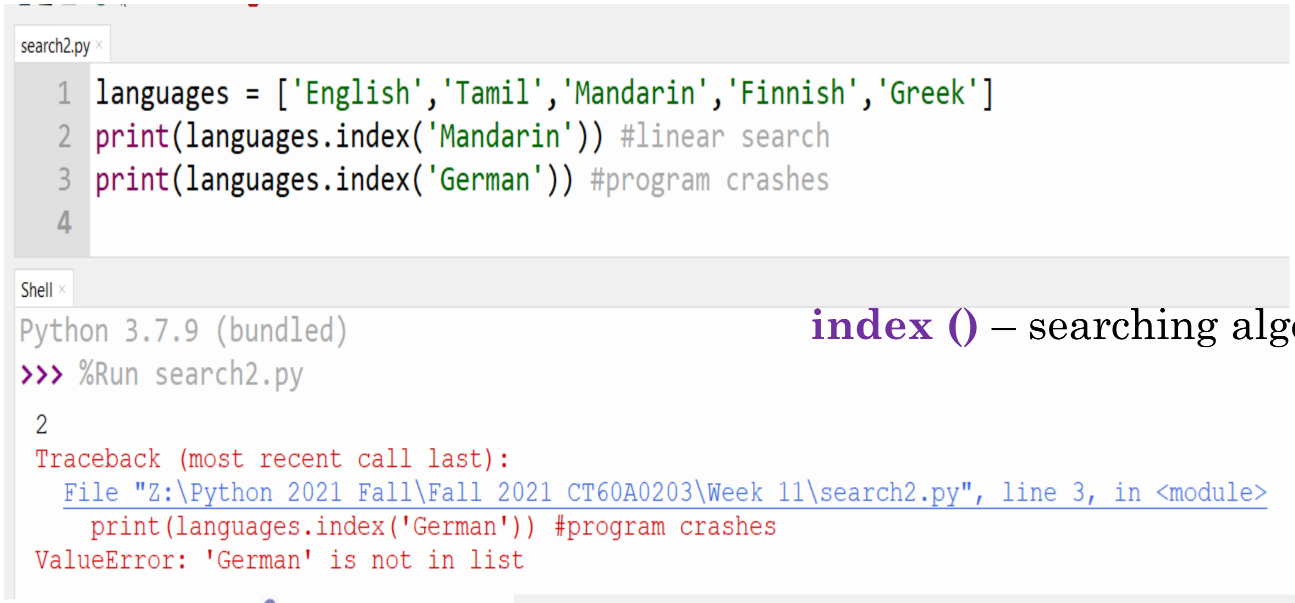
It is one of the simplest searching algorithms.
It merely iterates through the array/series/characters, checking for a match between the data at that position and that index.



Write a function **extractString()** that gets the string from the given index position till the end. Should not use Python's inbuilt functions

Example: print(extractString("Welcome to LUT", 11) -→ LUT

Linear/sequential search : Python



search2.py

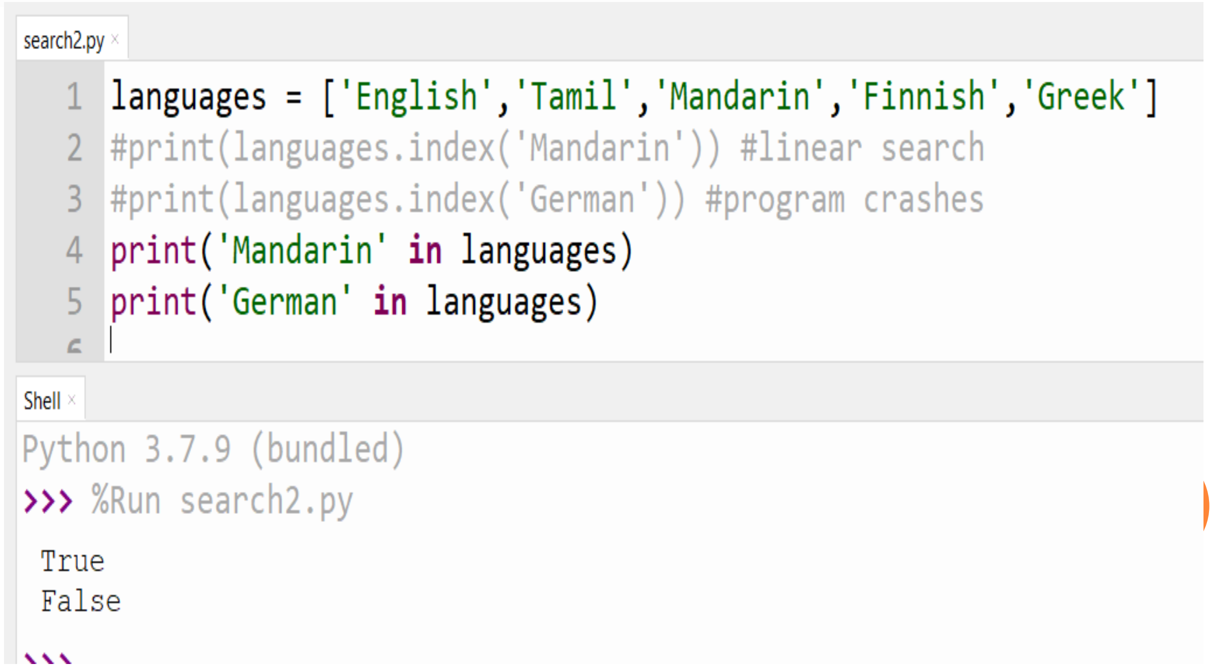
```python
languages = ['English','Tamil','Mandarin','Finnish','Greek']
print(languages.index('Mandarin')) #linear search
print(languages.index('German')) #program crashes

```

Shell

```
Python 3.7.9 (bundled)
>>> %Run search2.py

 2
 Traceback (most recent call last):
   File "Z:\Python 2021 Fall\Fall 2021 CT60A0203\Week 11\search2.py", line 3, in <module>
     print(languages.index('German')) #program crashes
 ValueError: 'German' is not in list
```

index () – searching algorithm in Python

in operator – searching algorithm (linear search) in Python- works well on iterable type data structure such as list, tuple, str..



search2.py

```python
languages = ['English','Tamil','Mandarin','Finnish','Greek']
#print(languages.index('Mandarin')) #linear search
#print(languages.index('German')) #program crashes
print('Mandarin' in languages)
print('German' in languages)

```

Shell

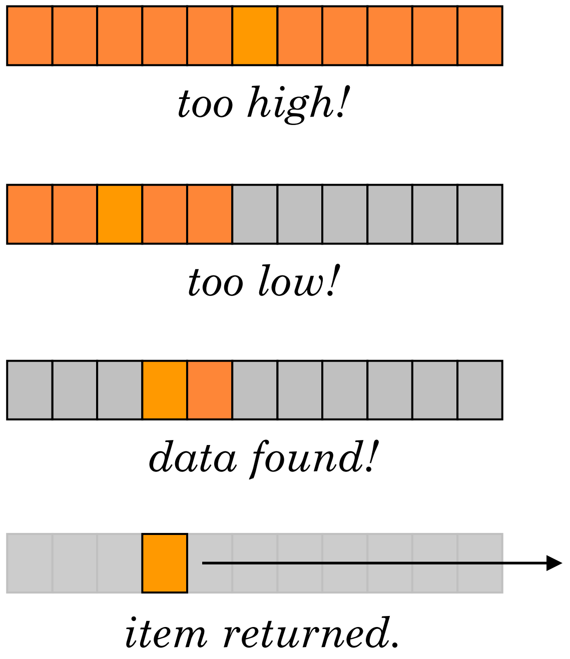
```
Python 3.7.9 (bundled)
>>> %Run search2.py

 True
 False

>>>
```

Binary (bisection) search

It is also used to find the element from the list of elements.
It is far more efficient than the linear search. However, it relies on one condition: the data that is being searched, has first been sorted.

It works by looking at the **middle element of the array**; depending on whether that item is **larger** or **smaller**, it **disregards the other half** of the array.

This is **repeated** until the target data has been **tracked down**, or the **boundaries are invalid** (the data does **not exist**.)



There are some more searching algorithms namely, hash based search, sublist search….

Binary search: Example

Consider the list of elements below, searching for **18**:

4	9	11	16	18	28	47	53	59	68	91

28 *is too* ***high*** *– the* ***right hand*** *half is discarded.*

4	9	11	16	18	28	47	53	59	68	91

11 *is too* ***low*** *– the* ***left hand*** *half is discarded.*

4	9	11	16	18	28	47	53	59	68	91

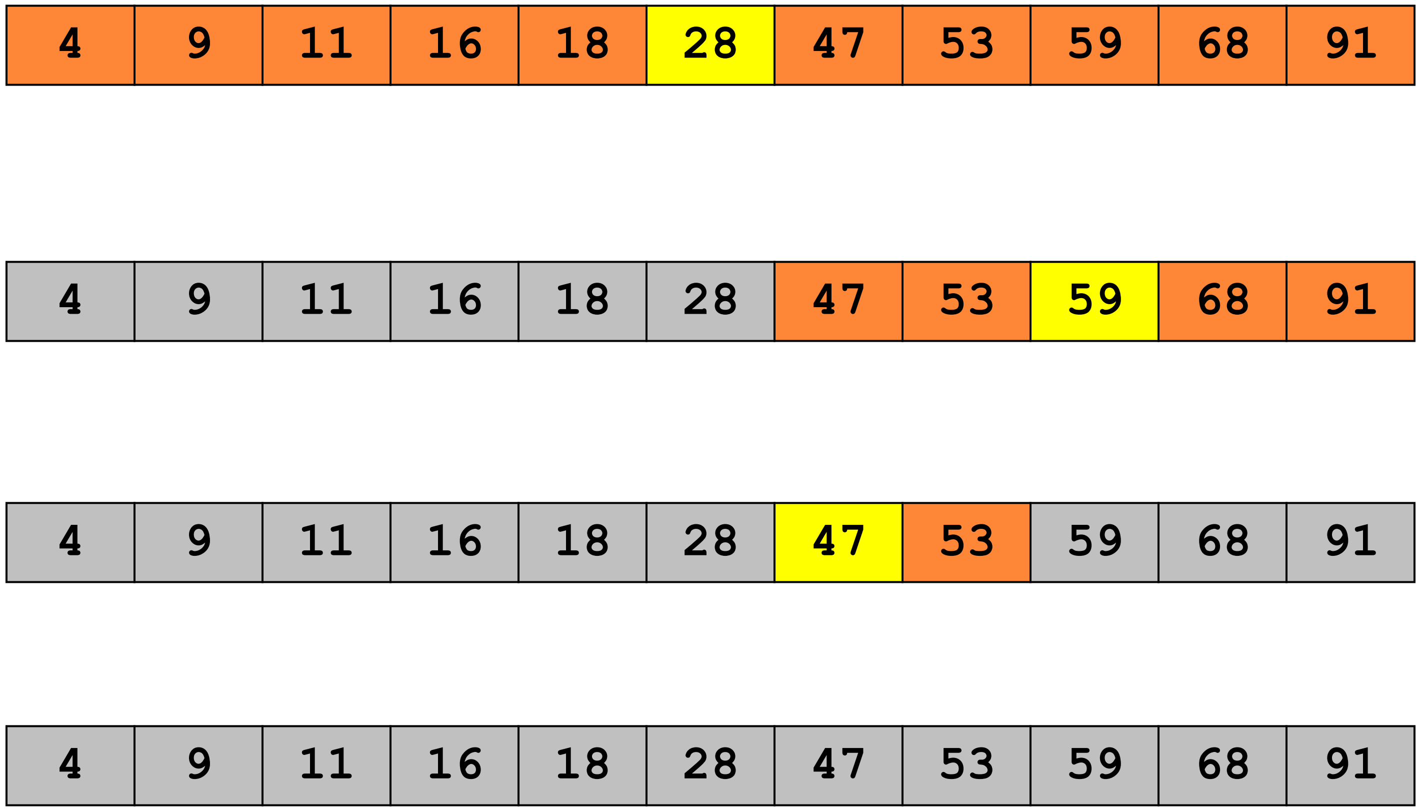
16 *is too* ***low*** *– the* ***left hand*** *half is discarded.*

4	9	11	16	18	28	47	53	59	68	91

*The target (**18**) has been **found**.*

Binary search: Example

searching for **37**:



Write a function **binsearchList(l1,value)** that accepts list and search value as arguments and return the index of search value if exist else “**not exists**” string to the called program.