# CSEN 102– Introduction to Computer Science

## Lecture 5: Algorithmic Problem Solving Iterative Operations Over Lists

Prof. Dr. Slim Abdennadher
Dr. Mohammed Salem
slim.abdennadher@guc.edu.eg,
mohammed.salem@guc.edu.eg

German University Cairo, Department of Media Engineering and Technology

27.10.2018 - 01.11 2018

## What you should have learned so far. . .

Algorithms can be constructed by the following operations:

- Sequential Operation
- Conditional Operation
- Iterative Operation

## Syntax

### Conditional control flow: general format

```
if condition:
        # <operations for the then-part>
else
        # <operations for the else-part>
```

### Iterative control flow: general format

```
while <condition>:
    step 1: <operation>
    ...
    step i: <operation>
```

## Syntax

By the way…

what is the control-flow syntax for sequential operations?

## Sequence, conditional, and iteration in one algorithm

- Remember the Euclidean Algorithm from lecture 1, slide 24 to determine the greatest common divisor (GCD) of two integers.
- Method: To find the GCD of two numbers, repeatedly replace the larger by subtracting the smaller from it until the two numbers are equal.

Consider this little warm-up...

```
A, B = eval(input()), eval(input())
while not A == B:
    if A > B:
        A = A - B
    else:
        B = B - A
print("The GCD is ")
print(A)
```

## Lists

- A list is a collection of data.
- In Python, we denote a list with `[]`
- `A[i]` corresponds to the value of the item in position `i`
- To get a list of `n` elements:

  ```
  A = eval(input())
  ```
- To get its length `n`:

  ```
  n = len(A)
  ```

## Example I

Example

Given a list of *n* numbers, where *n* is odd, find the middle number in the list.

```python
list_A = eval(input())
n = len(list_A)
i = int(n/2)
mid = list_A[i]
print(mid)
```

## Example II

### Example

Given a list of numbers, find the sum of the numbers in the list.

```
list_A = eval(input())
n = len(list_A)
i = 0
result = 0
while i < n:
  result = result + list_A[i]
  i = i + 1

print(result)
```

## Example III

### Example

Given a list of numbers, find the number of times a given number occurs in the list.

```python
number = eval(input())
list_A = eval(input())
n = len(list_A)
count = 0
i = 0
while (i < n):
  if (list_A[i] == number):
    count += 1
  i +=1
print(count)
```

## Example IV

### Sequential search

Problem: Find the phone number of a given Name in an (unsorted) list of names and their phone numbers

| Names | Phone numbers |
|---|---|
| N0 | T0 |
| N1 | T1 |
| ... | ... |
| N999 | T999 |

## Example IV

### Sequential search

Problem: Find the phone number of a given Name in an (unsorted) list of names and their phone numbers

### Sequential search, 1st attempt

```python
Name = input()
list_N = eval(input())
list_T = eval(input())
if Name == list_N[0]: print(list_T[0])
if Name == list_N[1]: print(list_T[1])
# ...
if Name == list_N[999]: print(list_T[999])
```

## Example IV

### Sequential search, using a loop (2nd attempt)

```
Name = input()
list_N = eval(input())
list_T = eval(input())
i = 0
Found = False

while Found == False and i < 1000:
    if Name == list_N[i]:
        print(list_T[i])
        Found = True
    else:
        i = i + 1
if Found == False:
    print("Sorry, name is not in directory")
```

## Example V

### Lists: Find The Largest Number

Problem: Given a list of values $A_0, \ldots, A_{(n-1)}$, find the largest value and its (first) location

- Example:

|  |  |  | $\downarrow$ |  |  |  |  |
|---|---|---|---|---|---|---|---|
|  | A0 | A1 | A2 | A3 | A4 | A5 | A6 |
| Value | 5 | 2 | 8 | 4 | 8 | 6 | 4 |

  The largest number is 8 at location 2.

- Idea: Go through the entire list, at each iteration find the largest-so-far and record its location

## Example V

### Lists: Find The Largest Number

Problem: Given a list of values $A_0, \ldots, A_{(n-1)}$, find the largest value and its (first) location

- Example:

$$\begin{array}{cccccccc} & & i & & & & & \\ & & \downarrow & & & & & \\ & A0 & A1 & A2 & A3 & A4 & A5 & A6 \\ \text{Value} & 5 & 2 & 8 & 4 & 8 & 6 & 4 \end{array}$$

Largest value: $A_0$, 5
Position: 0

1. Set the largest-so-far to the value of $A_0$
2. Set location to 0
3. Set $i$ to 1

## Example V

### Lists: Find The Largest Number

Problem: Given a list of values $A_0, \ldots, A_{(n-1)}$, find the largest value and its (first) location

- Example:

|  |  | $\overset{i}{\downarrow}$ | $\overset{i}{\downarrow}$ |  |  |  |  |
|---|---|---|---|---|---|---|---|
|  | A0 | A1 | A2 | A3 | A4 | A5 | A6 |
| Value | 5 | 2 | 8 | 4 | 8 | 6 | 4 |

Largest value: $A_0$, 5
Position: 0

1. Compare the entry at position $i$ ($A_1$) with the current maximum
2. Since $A_0$ is bigger, do not update the current maximum
3. Set $i$ to $i + 1$ (now 2)

## Example V

### Lists: Find The Largest Number

Problem: Given a list of values $A_0, \ldots, A_{(n-1)}$, find the largest value and its (first) location

- Example:

$$\begin{array}{lccccccc} & & & i & i & & & \\ & & & \downarrow & \downarrow & & & \\ & A0 & A1 & A2 & A3 & A4 & A5 & A6 \\ \text{Value} & 5 & 2 & 8 & 4 & 8 & 6 & 4 \end{array}$$

Largest value: $A_0$, 5, $A_2$, 8
Position: 0,2

1. Compare the entry at position $i$ ($A_2$) with the current maximum
2. Since $A_2$ is bigger, update the current maximum
3. Set $i$ to $i + 1$ (now 3)

## Example V

### Lists: Find The Largest Number

Problem: Given a list of values $A_0, \ldots, A_{(n-1)}$, find the largest value and its (first) location

- Example:

$$\begin{array}{cccccccc} & & & & i & i & i & i \\ & & & & \downarrow & \downarrow & \downarrow & \downarrow \\ & \text{A0} & \text{A1} & \text{A2} & \text{A3} & \text{A4} & \text{A5} & \text{A6} \\ \text{Value} & 5 & 2 & 8 & 4 & 8 & 6 & 4 \end{array}$$

Largest value: $A_2$, 8
Position: 2

1. Compare the entry at position $i$ ($A_3$) with the current maximum
2. Since $A_2$ is bigger, do not update the current maximum
3. Set $i$ to $i + 1$ (now 4)... and so on (now 5)... and so on (now 6)...

## Example V

### Lists: Find The Largest Number

Problem: Given a list of values $A_0, \ldots, A_{(n-1)}$, find the largest value and its (first) location

- Example:

$$\begin{array}{lcccccccc} & & & & & & & i & i \\ & & & & & & & \downarrow & \downarrow \\ & A0 & A1 & A2 & A3 & A4 & A5 & A6 & \\ \text{Value} & 5 & 2 & 8 & 4 & 8 & 6 & 4 & \end{array}$$

Largest value: $A_2$, 8
Position: 2

1. As soon as $i$ is larger than the number of elements in the list
2. Stop the algorithm
3. Output the current maximum and position

## Example V

Largest number, python

```
list_A = eval(input())
n = len(list_A)
largest_so_far = list_A[0]
location = 0
i = 1
while i < n:
    if list_A[i] > largest_so_far:
        largest_so_far = list_A[i]
        location = i
    i = i + 1
print(largest_so_far)
print(location)
```

## Example VI

### List reversal I

Reverse a list in another list

```
list_A = eval(input())
n = len(list_A)
list_B = []
i = 0
while i < n:
  list_B = list_B + list_A[n-(i+1)]
  i = i + 1

print(list_B)
```

## Example VI

### List reversal II

Reverse a list in place

- Idea: Swap the first element with the last element and the second element with the last but one element and so on

```
list_A = eval(input())
n = len(list_A)
i = 0
while i < n//2:
  tmp = list_A[i]
  list_A[i] = list_A[n-(i+1)]
  list_A[n-(i+1)] = tmp
  i += 1

print(list_A)
```

## Iteration over Strings: Example VII

Characters in Strings:

Write an algorithm to print the characters in a String one by one

```
word = input()
n = len(word)    # len() gets the length of the String
i = 0            # the first char is at position 0
while(i<n):
  print(word[i])
  i +=1
```

## Iteration over Strings: Example VIII

Reverse a String:

Write an algorithm to reverse a given String

```
word = input()
n = len(word)    # len() gets the length of the String
i = n - 1        # the first char is at position 0
result = ""
while(i >= 0):
  result  += word[i]
  i -=1
print(result)
```