University College Cork
Computer Science
Department

Programming Training

Main Points:

- More Fundamental algorithms on Arrays.
- Reading / Writing from files
- Problem Solving

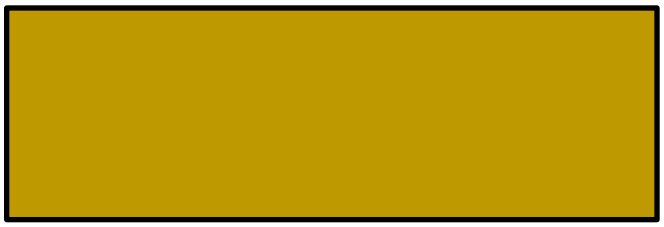
Functions in Python

A Python program is a sequence of a functions which can be executed.

Function = Routine = Procedure = Method = Solution for a sub-problem.

A Python function is written once and used/called as many times as needed.

```
def function_name(arg1, arg2,…):
```



```
# end function
```

1. function_name is the name of the function
2. arg1, arg2,…are the function arguments (usually the inputs)
3. the function body must find the output and return it.

Steps to work with functions

1. Define these functions you want to work with

- Start from the arguments as inputs.
- Give the sequence of statements to construct the output res.
- Return the output res.

2. Call a function at any time you need.

- The call is function_name(a1,a2,…) and returns a value that can be used.
- a1, a2,… are the actual/current arguments

3. Actual arguments are NOT change after the call even if they change in the function

Arrays / Lists

Python lists represent sequence of [**similar]** elements

- between [ ]
- separated by ,
- which are i**ndexed**.

[ ] ➔ represents lists / arrays.

Examples:

x = [1, 2, 3, 1, 0] ➔ list of 5 int numbers

name = [‘s’, ’a’, ’b’, ’i’, ’n’] ➔ list of 5 string elements

list = [1, 2, ‘sabin’, [0, 1, 3]] ➔ list with elements of different nature

Arrays / Lists Indexing

Python lists elements can be indexed using indices 0, 1, 2, ….

A list l with n elements has the elements **l[0], l[1], l[2],…, l[n-1].**

l represents the list while l[i] is an element of the list.

the first element of the list is indexed with 0.

it is possible to have negative indexing.

For a list l, the number of elements is given by the function len

len(l) ➔ number of elements in l.

ANY COMPUTATION WITH ARRAYS MUST USE THE LOOP FOR.

Arrays / Lists Slicing

Python lists can be sliced indices using a range introduced by ':'.

A list l with n elements has the elements **l[0], l[1], l[2],…, l[n-1].**

Slicing Types:

l[i1 : i2] ➔ the list with all the elements between i1 and i2-1

l[i1:] ➔ the list with all the elements from index i1 to the end.

l[:i2] ➔ the list with all elements from 0 to i2-1.

l[:] ➔ the list with all elements

l[::-1] ➔ the list in reverse

Arrays / Lists Operations

List operations:

list1 + list2 ➔ concatenates the lists. list1 * nr ➔ list1 clones nr times.

elem in list ➔ tests if elem is in list.

List functions

len(l) ➔ length of l

max(l) ➔ maximum element of a list min(l) ➔ minimum element

list(sequence) ➔ converts the sequence to a list.

l.append(elem) ➔ appends elem to the list

l.insert(index, elem) ➔ inserts elem to the list at the specified index

l.count(elem) ➔ counts how many times elem is in the list

l.index(elem) ➔ returns the first index of elem in the list

Arrays / Lists Traversal

ANY COMPUTATION WITH LIST MUST USE THE LOOP FOR.

Ways to traverse the list

1) Traverse the list as iterator

```
for elem in l :
        # process elem
        …
# endfor
```

2) Traverse the list elements

```
n = len(l)
for i in range(n) :
        # process the element l[i]
        …
# endfor
```

Input a List

Suppose that we need a list with n int elements.

- read the number of elements in the list

- read n elements using a for loop

```
n = int(input(‘’n=))
l = []
for i in range(n) :
        elem = int(input())
        l.append(elem)
# endfor
```

This scheme will work for any programming language.

Input a List

Suppose that we need a list with int elements.

```
l = input('type elements: ').split()
l = list(map(int, l))
```

A Python approach:

- read some elements and make an array of strings from them
- convert them to a sequence of int elements
- make a list from this sequence.

Print a List

Suppose that we have a list with n int elements.

- use a for loop to print each element

```
n = len(l)
for i in range(n) :
        print(l[i], end = ‘ ‘)
# endfor
```

This scheme will work for any programming language.

print(l[i], end = ‘ ‘) printing on the same line.

print(l) ➔ the whole list is printed between []

The summation problem

The summation problem: If a=[a[0], a[1], …, a[n-1]] is a list with n elements find sum = a[0]+a[1]+…+a[n-1].

Inputs: a=[a[0], a[1], …, a[n-1]].

Output: sum.

How to do it:

Step 1. (initialization) sum = 0

Step 2. (repetition) repeat for i=0,1,2,…,n-1

increase s with a[i];

For the multiplication problem we have the steps:

Step 1. (initialization) p=1

Step 2. (repetition) repeat for i=0,1,2,…,n-1

increase p by a[i];

```
def listSum(a):

    n = len(a)
    # initialise sum
    sum = 0
    # traverse the list
    for i in range(n) :
        sum = sum + a[i]
    # endfor

    return sum
# end listSum
```

```
def listProduct(a):

    n = len(a)
    # initialise prod
    prod = 1
    # traverse the list
    for i in range(n) :
        prod = prod * a[i]
    # endfor

    return prod

# end listProduct
```

The maximum problem

The maximum problem:

If a = [a[0], a[1], …, a[n-1]] is a list with n elements find the maximum element max and the position which holds it.

Inputs: a=[a[i], i=0,1,…,n-1].

Output: max and pos.

How to do it:

Step 1. (initialization) max=-infinity; pos=-1;

Step 2. (repetition) repeat for i=0,1,2,…,n-1

test max against a[i]; if defeated change max and pos

```
def listMax(a):

    n = len(a)

    # initialise max, pos
    max = -10000000
    pos = -1
    # traverse the list
    for i in range(n) :
        if max < a[i] :
            max = a[i]
            pos = i
    # endfor

    return max, pos

# end listProduct
```

```
def listMin(a):

    n = len(a)

    # initialise max, pos
    min = 10000000
    pos = -1
    # traverse the list
    for i in range(n) :
        if min > a[i] :
            min = a[i]
            pos = i
    # endfor

    return min, pos

# end listProduct
```

The counting problem

The counting problem:

If a [a[0], a[1], …, a[n-1]] is a list with n elements find the number of elements that satisfy a condition C.

Inputs: a= [a[0], a[1], …, a[n-1]].

Output: nr.

How to do it:

Step 1. (initialization) nr=0;

Step 2. (repetition) repeat for i=0,1,2,…,n-1

test if a[i] satisfies C and then increase nr;

COUNT HOW NAMY NUMBERS ARE POSITIVE

```
def listPositiveCount(a):

    n = len(a)

    # initialise count
    count = 0

    # traverse the list
    for i in range(n) :
        if a[i] > 0:
            count = count +1
        # endif
    # endfor

    return count

# end listProduct
```

Python Approach

If a [a[0], a[1], …, a[n-1]] is a list then

max(a) ➔ maximum element

min(a) ➔ minimum element

sum(a) ➔ the sum of the elements

To do List

1. Solve the HW problems.
2. Read about lists