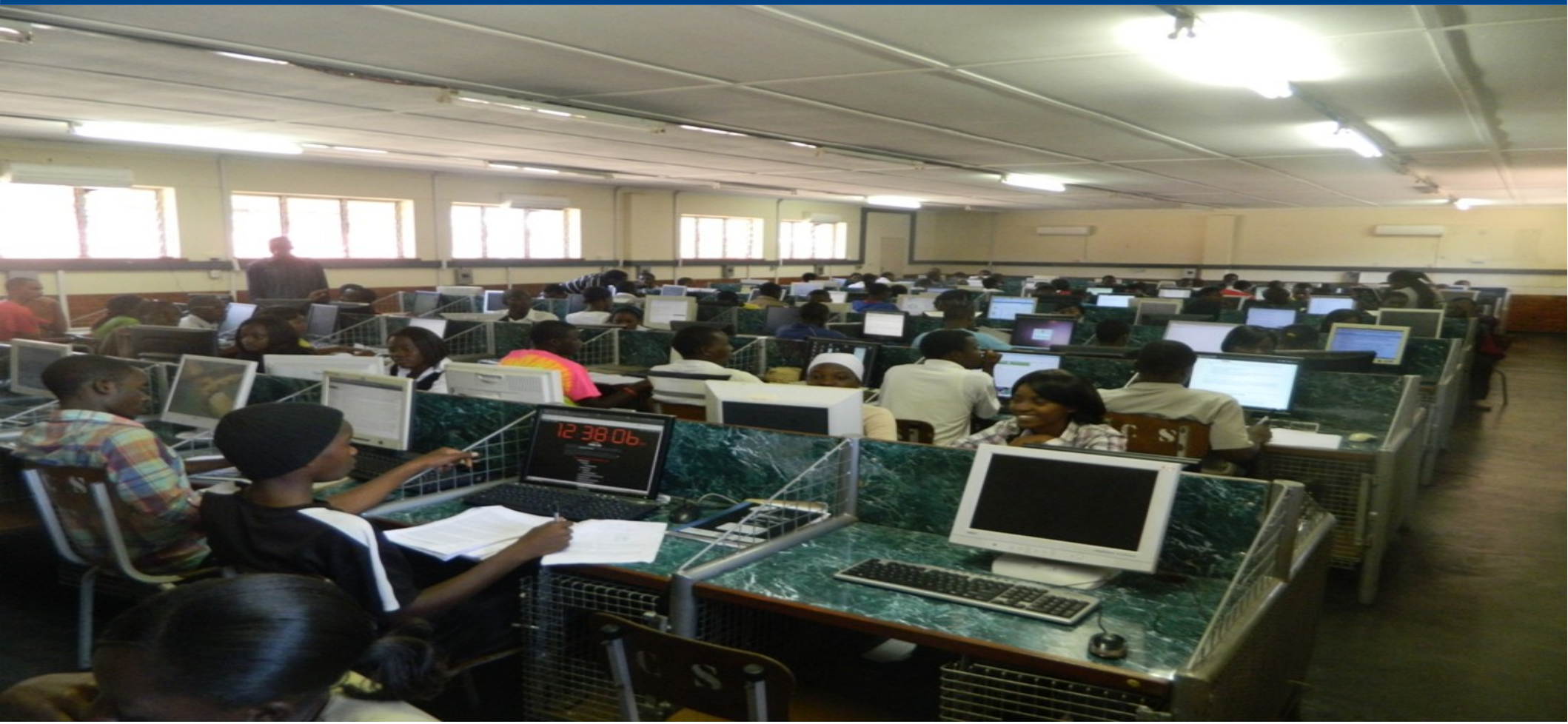
2nd Lecture



ICT & Electronics

Introduction to Data Structures and Algorithms in Python

T D KAVU

Contents

- Input and Output
- Control Structures
- Defining Functions
- Object-Oriented Programming in Python

Input and Output

- We often have a need to interact with users, either to get data or to provide some sort of result
- Python provides us with a function that allows us to ask a user to enter some data and returns a reference to the data in the form of a string. The function is called input.
- Python's input function takes a single parameter that is a string.
- This string is often called the prompt because it contains some helpful text prompting the user to enter something
- *aName = input('Please enter your name: ')*
- *aName = input("Please enter your name ")*
- *print("Your name in all capitals is",aName.upper(),*
- *"and has length", len(aName))*

Type Conversion

- If you want this string interpreted as another type, you must provide the type conversion explicitly
- sradius = input("Please enter the radius of the circle ")
- radius = float(sradius)
- diameter = 2 * radius

String Formatting

- It is possible to change the separator character by setting the sep argument. In addition, each print ends with a newline character by default. This behavior can be changed by setting the end argument
- *>>> print("Hello")*
- *Hello*
- *>>> print("Hello","World")*
- *Hello World*
- *>>> print("Hello","World", sep="***")*
- *Hello***World*
- *>>> print("Hello","World", end="***")*
- *Hello World***>>>*
- *print("%s is %d years old." % (aName, age))*

Continue

```
>>> price = 24
>>> item = "banana"
>>> print("The %s costs %d cents"%(item,price))
The banana costs 24 cents
>>> print("The %+10s costs %5.2f cents"%(item,price))
The     banana costs 24.00 cents
>>> print("The %+10s costs %10.2f cents"%(item,price))
The     banana costs      24.00 cents
>>> itemdict = {"item":"banana","cost":24}
>>> print("The %(item)s costs %(cost)7.1f cents"%itemdict)
The banana costs    24.0 cents
>>>
```

Control Structures

- As we noted earlier, algorithms require two important control structures: iteration and selection
- Both of these are supported by Python in various forms
- *>>> counter = 1*
- *>>> while counter <= 5:*
- *... print("Hello, world")*
- *... counter = counter + 1*
- *A fragment such as*
- *while counter <= 10 and not done:*
- *would cause the body of the statement to be executed only in the case where both parts of the condition are satisfied. The value of the variable counter would need to be less than or equal to 10 and the value of the variable done would need to be False (not False is True) so that True and True results in True.*

For Statement

- The for statement can be used to iterate over the members of a collection, so long as the collection is a sequence.
- *>>> for item in [1,3,6,2,5]:*
- *... print(item)*
- *assigns the variable item to be each successive value in the list [1,3,6,2,5]. The body of the iteration is then executed. This works for any collection that is a sequence (lists, tuples, and strings).*

Continue

- A common use of the for statement is to implement definite iteration over a range of values.
- >>> for item in range(5):
- ... print(item**2)
- ...
- will perform the print function five times. The range function will return a range object representing the sequence 0,1,2,3,4 and each value will be assigned to the variable item. This value is then squared and printed.

For in Strings

```
wordlist = ['cat','dog','rabbit']
letterlist = [ ]
for aword in wordlist:
    for aletter in aword:
        letterlist.append(aletter)
print(letterlist)
```

Numpy Arrays

- The NumPy module provides array sequences that can store numbers or characters in a space-efficient way
- Arrays in NumPy can be generated from a list or a tuple with the array-method, which transforms sequences of sequences into two dimensional arrays:
- x = np.array(((11,12,13), (21,22,23), (31,32,33)))
- print x
- [[11 12 13]
- [21 22 23]
- [31 32 33]]
- The attribute ndim tells us the number of dimensions of an array:
- x.ndim
- ax = np.array([1,2,3])
- ay = np.array([3,4,5])
- print(ax)
- print(ax*2)
- print(ax+10)
- print(np.sqrt(ax))

A bit of Lists

- There is an alternative method for creating a list that uses iteration and selection constructs known as a list comprehension
- A list comprehension allows you to easily create a list based on some processing or selection criteria.
- For example, if we would like to create a list of the first 10 perfect squares, we could use a for statement:

```
>>> sqlist=[]
>>> for x in range(1,11):
        sqlist.append(x*x)
>>> sqlist
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
## Using a list comprehension, we can do this in one step as
>>> sqlist=[x*x for x in range(1,11) if x%2 != 0]
>>> sqlist
[1, 9, 25, 49, 81]
```

Exercises

- 1.*Given a list b=[9.5,9.25,9.75,9.50]*
- *Add 9.00 to the end of the list*
- *Print the index of the greatest item in the list*
- 2. *Create a list using list comprehension of perfect squares of even numbers between 10 and 20*
- *3. S = [x**2 for x in range(10)]*
- *V = [2**i for i in range(13)]*
- *Creat a list of all odd numbers in list S divided by even numbers in list V*

Excercise

Given the list:

values = [-4, 4, -1, -2, 10, 3]

Write a **while** loop that creates two lists from this one: a list of positive values and a list of negative values.

Hint: you will need to start with two empty lists

positives = [] and **negatives = []**

Exercise

Create a function called ***count_letter*** that takes as input a string ***txt*** and a character ***char***, and returns the number of times char appears in ***txt***, ignoring case.

Foe example:

>>> count_letter("Php, Perl, or Python?", "p")