## Into the Game

Taming the Python

- Jump into the cage
- Get hands dirty
- Start with examples
- Do some programming

# A Sample Program

Let's have a look at the simple "Find the largest number" in an array program in the next page.

```
def largest_number(numlist):
    if len(numlist) <= 1:
        print "Empty Array"
        return -999;

    max = numlist(0)

    for x in numlist(1:):
        if max < x:
            max = x

    return max


def read_numbers():
    cardi = int(raw_input("How many numbers: "))

    listnum = ()
```

```

  if cardi > 0:
    while cardi != 0:
      next_num = (int(raw_input("Next number: "))
      listnum.append(next_num)
      cardi -= 1

  return listnum


list_of_numbers = read_numbers()

if len(list_of_numbers) > 0:
  print "\nThe largest number is",
  print largest_number( list_of_numbers )
```

## Line by line Analysis

It is not complicated.
But it is detail oriented.
Syntax is not important to learn byheart
Vital thing: Have the concept in mind.
All the rest come automatically.

```
def largest_number(numlist):
  if len(numlist) <= 1:
    print "Empty Array"
    return -999;
```

- Function definition with `def` keyword
- Not necessary to have the type of the parameter
- There is a colon ':' at the end of function definition; Also at the end of the `if` statement.
- `len` is the keyword for getting the length of arrays
- Just notice the print statement.
- Semicolon ';' at the end of `return` statement.

```
max = numlist(0)

for x in numlist(1:):
    if max < x:
        max = x

return max
```

- Python arrays (lists) start with index '0'
- Notice the `numlist[1:0]` - this is called slicing. It gives a list with all the elements of the original list starting from index '1'
- `for` can take each item from the list. (We'll learn about iterators later)

```
def read_numbers():
    cardi = int(raw_input("How many numbers: "))

    listnum = []
```

- Function without parameters
- Reading input `raw_input` - reads the input as a string.
- Initialising a list with empty list

```
    if cardi > 0:
        while cardi != 0:
            next_num = (int(raw_input("Next number: "))
            listnum.append(next_num)
            cardi -= 1

    return listnum
```

- How a `while` loop works.
- One of the list operation - `append` - adds the item provided, to the end of the list.
- `cardi = cardi - 1`

## How do we call the functions?

```
list_of_numbers = read_numbers()

if len(list_of_numbers) > 0:
  print "\nThe largest number is",
  print largest_number( list_of_numbers )
```

- Just call them from the command line / from outside the function
- Look at the different print statements

## The Output

```
(sadanand@lxmayr10 ~ pffp)python largestnumber.py
How many numbers: 2
Next number: 34
Next number: 4566

The largest number is 4566
(sadanand@lxmayr10 ~ pffp)
```

Observation: The two print statements printed in a single line. The imporance of "Comma".

## Variables, Values and Types

- Variables are just the positions of what you store in them.
- In the main memory
- Usual naming conventions. ‘_’ or alphabets as beginning, then could be followed by any alphanumeric characters or ‘_’ – `this_is_a_variable`, `____this_too_`, `_t_h_e_3rd_one`

## Values and Datatypes

- Values can be different datatypes
- Numbers
  `int, long, float, complex`
- Characters
  A single character, string, unicode, ..
- Collections
  List, Dict, Set, ...
- Other objects we could make
  Tree, Graph, ...,

## A break from 'data flooding'

How to write and run a program?

1. Open an editor - your favorite one
   Some editors support syntax highlighting for python. (e.g.: Vim, Emacs, IDLE, etc.)
   Some don't: Notepad
2. Type in the program
3. Save it with extension `py` giving `program.py`

## Four ways to Run it

1. Run it with `$python program.py` at the prompt
2. Use IDLE to run it (for Windows)
3. Have `#!/usr/bin/python` as the first line of the file; make the file executable and run it from terminal. $./program.py
4. Configure your editor to have a shortcut key to run it straight from the editor.

## Some basic Datatypes

- Numbers
- String
- Lists

1. Numbers

```
>>> 2+2
4
>>> (50-5*6)/4
5
>>> 7/3
2
>>> 7/-3
-3 (floor)

>>> width = 20
>>> height = 5*9
>>> width * height
900

>>> x = y = z = 0
>>> x
0
```

```
>>> y
0
>>> z
0

>>> 3 * 3.75 / 1.5
7.5
>>> 7.0 / 2
3.5

>>> 1j * 1J
(-1+0j)
>>> 1j * complex(0,1)
(-1+0j)
>>> 3+1j*3
(3+3j)
>>> (3+1j)*3
(9+3j)
```

```
>>> (1+2j)/(1+1j)
(1.5+0.5j)

>>> a=1.5+0.5j
>>> a.real
1.5
>>> a.imag
0.5
```

2. Strings

```
>>> 'spam eggs'
'spam eggs'
>>> 'doesn\'t'
"doesn't"
>>> "doesn't"
"doesn't"
>>> '"Yes," he said.'
'"Yes," he said.'
```

```
>>> "\"Yes,\" he said."
'"Yes," he said.'
>>> '"Isn\'t," she said.'
'"Isn\'t," she said.'

>>> word = 'Help' + 'A'
>>> word
'HelpA'
>>> '<' + word*5 + '>'
'<HelpAHelpAHelpAHelpAHelpA>'

>>> word(4)
'A'
>>> word(0:2)
'He'
>>> word(2:4)
'lp'

```

```
>>> word[:2]
'He'
>>> word[2:]
'lpA'

>>> word[0] = 'x'
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assign
```

3. Lists

```
>>> a = ['spam', 'eggs', 100, 1234]
>>> a
['spam', 'eggs', 100, 1234]


>>> a[0]
'spam'
```

```
>>> a(3)
1234
>>> a(-2)
100

>>> a(1:-1)
('eggs', 100)

>>> a(:2) + ('bacon', 2*2)
('spam', 'eggs', 'bacon', 4)

>>> 2*a(:3) + ('Boo!')
('spam', 'eggs', 100, 'spam', 'eggs', 100, 'B

>>> a
('spam', 'eggs', 100, 1234)
>>> a(2) = a(2) + 23
>>> a
```

```
('spam', 'eggs', 123, 1234)

Replace some items:
>>> a[0:2] = [1, 12]
>>> a
[1, 12, 123, 1234]

Remove some:
>>> a[0:2] = []
>>> a
[123, 1234]

Insert some:
>>> a[1:1] = ['bletch', 'xyzzy']
>>> a
[123, 'bletch', 'xyzzy', 1234]

Clear the list: replace all items with an em
```

```
>>> a[:] = []
>>> a
[]

>>> q = [2, 3]
>>> p = [1, q, 4]
>>> len(p)
3
>>> p[1]
[2, 3]
>>> p[1][0]
2
>>> p[1].append('xtra')
>>> p
[1, [2, 3, 'xtra'], 4]
>>> q
[2, 3, 'xtra']
```

## Basic Operators

- Numerical Operators
  `+, -, *, /, %`
- Logical Operators
  `True, False, and, not, or`
- Bitwise Operators
  `&, |, ^`
- Shift Operators
  `<<, >>`

## Little Points

- `chr(i)`
  Return a string of one character whose ASCII code is the integer *i*. For example, chr(97) returns the string `a'
- `ord(c)`
  Given a string of length one, return an integer representing character. For example, ord('a') returns the integer 97 [1]

[1] Unicode

## Three Different Fibonaccis

Write three different python functions, each of which gives the fibonacci number corresponding to the input number.
Bonus: Write a $4^{th}$ and better function.

## Find out the square root

Write a program to find out the square root of a given number. (Without the help of python math library)
Bonus: Extend this to $n^{th}$ root.

## atoi and itoa

Write a program, without using the `int` functionality of python, to convert a string (representing an integer) to the integer.
Also, do the reverse: Integer to String
Bonus: Extend this to floating points

## Combinations of Characters

Write a program to generate all the combinations of all the characters in a given string, or a list of characters.
Bonus: Beauty of the program.