Support for List Initializers

- Objective: allow users to provide a python tuple (or list?) to implicitly construct and initialize an object that is needed in a function or method argument.
- Instead of doing the following when calling a function/method that requires const A&

```
a = ROOT.A(arg1,arg2)
ROOT.function(a)
```

  the user can do

```
ROOT.function((arg1,arg2))
```

- Implemented by Pere in his PR

https://github.com/root-project/root/pull/993

Example

```
import ROOT

cppcode = """
struct A {
  A(int _i, double _f) : i(_i), f(_f) {}
  A(int _i) : i(_i), f(99.99) {}
  int i;
  double f;
};
int get_i(const A& a) { return a.i; }
double get_f(const A& a) { return a.f; }
"""

ROOT.gInterpreter.Declare(cppcode)

// Pass an object: this prints 1
print( ROOT.get_i(ROOT.A(1,2.)) )

// Pass a tuple: this prints 3
print( ROOT.get_i((3,4.)) )

// What about a list? This prints 6.0
print( ROOT.get_f([5,6.]) )
```

Tuple vs List

```
t = (1, 2.) // immutable
ROOT.get_i(t)

l = [3, 4.] // mutable
ROOT.get_f(l) // returns 4.0

// Modify and call again
l[0] = 5
l[1] = 6.
ROOT.get_f(l) // returns 6.0
```