

Optimizando Python usando Cython

jampp

Disclaimer



DISCLAIMER

No soy ningun experto en el tema de usar Cython

Vean la charla de Facundo Batista:
Python más rapido que C

Hay varias fomas de usar Cython. Aca voy a mostrar una.

github.com/tzulberti/charlas

Levenshtein Distance

Calcula la cantidad de operaciones (cambiar un char por otro, sacar, o agregar uno) para convertir de una palabra a otra

1. ola y Hola => necesito agregar la H asique la distancia es 1
2. chau y char => necesito cambiar la U por la R asique la distancia también es 1

Levenshtein Distance

```
function LevenshteinDistance(char s[1..m], char t[1..n]):  
    // for all i and j, d[i,j] will hold the Levenshtein distance between  
    // the first i characters of s and the first j characters of t  
    // note that d has (m+1)*(n+1) values  
    declare int d[0..m, 0..n]  
  
    set each element in d to zero  
  
    // source prefixes can be transformed into empty string by  
    // dropping all characters  
    for i from 1 to m:  
        d[i, 0] := i  
  
    // target prefixes can be reached from empty source prefix  
    // by inserting every character  
    for j from 1 to n:  
        d[0, j] := j  
  
    for j from 1 to n:  
        for i from 1 to m:  
            if s[i] = t[j]:  
                substitutionCost := 0  
            else:  
                substitutionCost := 1  
            d[i, j] := minimum(d[i-1, j] + 1,           // deletion  
                               d[i, j-1] + 1,           // insertion  
                               d[i-1, j-1] + substitutionCost) // substitution  
  
    return d[m, n]
```

Levenshtein Distance

```
def levenshtein(seq1, seq2):
    size_x = len(seq1) + 1
    size_y = len(seq2) + 1
    matrix = [[0] * size_y for _ in range(size_x)]

    for x in range(size_x):
        matrix[x][0] = x
    for y in range(size_y):
        matrix[0][y] = y

    for x in range(1, size_x):
        for y in range(1, size_y):
            if seq1[x-1] == seq2[y-1]:
                substitution_cost = 0
            else:
                substitution_cost = 1

            matrix[x][y] = min(
                matrix[x-1][y] + 1, # deletion
                matrix[x][y-1] + 1, # insertion
                matrix[x-1][y-1] + substitution_cost, #substitution
            )

    return matrix[size_x - 1][size_y - 1]
```

Codigo C

```
// taken from https://en.wikibooks.org/wiki/Algorithm\_Implementation/Strings/Levenshtein\_distance#C
int levenshtein(char *s1, char *s2) {
    unsigned int x, y, s1len, s2len;
    s1len = strlen(s1);
    s2len = strlen(s2);
    unsigned int matrix[s2len+1][s1len+1];
    matrix[0][0] = 0;
    for (x = 1; x <= s2len; x++)
        matrix[x][0] = matrix[x-1][0] + 1;
    for (y = 1; y <= s1len; y++)
        matrix[0][y] = matrix[0][y-1] + 1;
    for (x = 1; x <= s2len; x++)
        for (y = 1; y <= s1len; y++)
            matrix[x][y] = MIN3(matrix[x-1][y] + 1, matrix[x][y-1] + 1, matrix[x-1][y-1] + (s1[y-1] == s2[x-1] ? 0 : 1));

    return(matrix[s2len][s1len]);
}
```

Benchmark

- Use este diccionario de palabras en ingles:
<https://raw.githubusercontent.com/dwyl/english-words/master/words.txt>
- Convierto en un set cosa de que no necesariamente tenga el mismo orden
- Creo distintos archivos con diferentes cantidad de pares de palabras del set. Los distintos archivos además de tener diferente cantidad de palabras tienen diferentes pares de palabras

Comparacion de performance

Cantidad de comparaciones	C Puro	Python Puro	X veces mas lento
233271	0.061	15.174	248
116635	0.030	9.007	300
77757	0.019	5.605	295
58317	0.011	4.010	364
46654	0.009	3.796	421
38878	0.008	2.558	319
33324	0.009	2.066	229
29158	0.008	2.029	253

Usando extensiones en C

```
#include <Python.h>

static PyObject *
greet_name(PyObject *self, PyObject *args)
{
    const char *name;

    if (!PyArg_ParseTuple(args, "s", &name))
    {
        return NULL;
    }

    printf("Hello %s!\n", name);

    Py_RETURN_NONE;
}

static PyMethodDef GreetMethods[] = {
    {"greet", greet_name, METH_VARARGS, "Greet an entity."},
    {NULL, NULL, 0, NULL}
};

static struct PyModuleDef greet =
{
    PyModuleDef_HEAD_INIT,
    "greet",      /* name of module */
    "",          /* module documentation, may be NULL */
    -1,          /* size of per-interpreter state of the module, or -1 if the module keeps state in global variables. */
    GreetMethods
};

PyMODINIT_FUNC PyInit_greet(void)
{
    return PyModule_Create(&greet);
}
```

Buscando el cuello de botella

Python is primarily slow because of its dynamic nature and versatility. It can be used as a tool for all sorts of problems, where more optimised and faster alternatives are probably available.

Buscando el cuello de botella

```
(charlas) tzulberti@desktop /media/data/Proyectos/charlas/meetup-pyar-2018-cython/pure-python (master) $ python -m cProfile -o cprofile.output main.py ../dataset.15.txt
(charlas) tzulberti@desktop /media/data/Proyectos/charlas/meetup-pyar-2018-cython/pure-python (master) $ ipython
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.5.0 -- An enhanced Interactive Python. Type '?' for help.
```

```
In [1]: import pstats
```

```
In [2]: p = pstats.Stats('cprofile.output')
```

```
In [3]: p.sort_stats('cumulative').print_stats(10)
```

```
Wed Aug 15 20:52:26 2018      cprofile.output
```

2793606 function calls (2793605 primitive calls) in 8.262 seconds

Ordered by: cumulative time

List reduced from 99 to 10 due to restriction <10>

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
2/1	0.000	0.000	8.262	8.262	{built-in method builtins.exec}
1	0.000	0.000	8.262	8.262	main.py:3(<module>)
1	0.000	0.000	8.261	8.261	main.py:9(main)
1	0.130	0.130	8.257	8.257	main.py:17(do_logic)
29158	4.657	0.000	7.988	0.000	/media/data/Proyectos/charlas/meetup-pyar-2018-cython/pure-python/difference.py:5(levenshtein)
2589177	3.178	0.000	3.178	0.000	{built-in method builtins.min}
29158	0.066	0.000	0.100	0.000	main.py:13(<lambda>)
29158	0.093	0.000	0.093	0.000	/media/data/Proyectos/charlas/meetup-pyar-2018-cython/pure-python/difference.py:8(<listcomp>)
58320	0.060	0.000	0.060	0.000	{built-in method builtins.len}
29158	0.038	0.000	0.038	0.000	{method 'split' of 'str' objects}

Cython

- Permite escribir extensiones de C de Python en Python.
- Es código que corre en el Python runtime environment, pero en vez de compilar a bytecode interpretado de Python compila a código nativo
- Se instala como cualquier otro paquete de python

`pip install cython`

- Seguramente tengan que instalar cosas del sistema:

`sudo apt-get install python-dev build-essentials python3-dev`

Usando Cython

```
(charlas) tzulberti@laburo ~/workspace/charlas/meetup-pyar-2018-cython/cython-first-version (master) $ ls
difference.py  main.py
(charlas) tzulberti@laburo ~/workspace/charlas/meetup-pyar-2018-cython/cython-first-version (master) $ cythonize --inplace difference.py
Compiling /home/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first-version/difference.py because it changed.
[1/1] Cythonizing /home/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first-version/difference.py
running build_ext
building 'difference' extension
creating /home/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first-version/tmpxci3n4vd/home
creating /home/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first-version/tmpxci3n4vd/home/tzulberti
creating /home/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first-version/tmpxci3n4vd/home/tzulberti/workspace
creating /home/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first-version/tmpxci3n4vd/home/tzulberti/workspace/charlas
creating /home/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first-version/tmpxci3n4vd/home/tzulberti/workspace/charlas/meetup
r-2018-cython
creating /home/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first-version/tmpxci3n4vd/home/tzulberti/workspace/charlas/meetup
r-2018-cython/cython-first-version
x86_64-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-prototypes -g -fstack-protector-strong -Wformat -Werror=format-security
-Wdate-time -D_FORTIFY_SOURCE=2 -fPIC -I/usr/include/python3.5m -I/home/tzulberti/envs/charlas/include/python3.5m -c /home/tzulberti/worksp
charlas/meetup-pyar-2018-cython/cython-first-version/difference.c -o /home/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first
on/tmpxci3n4vd/home/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first-version/difference.o
x86_64-linux-gnu-gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,-Bsymbolic-functions -Wl,-z,relro -Wl,-Bsymbolic-functions -Wl
lro -g -fstack-protector-strong -Wformat -Werror=format-security -Wdate-time -D_FORTIFY_SOURCE=2 /home/tzulberti/workspace/charlas/meetup
2018-cython/cython-first-version/tmpxci3n4vd/home/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first-version/difference.o -l
e/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first-version/difference.cpython-35m-x86_64-linux-gnu.so
(charlas) tzulberti@laburo ~/workspace/charlas/meetup-pyar-2018-cython/cython-first-version (master) $ ls
difference.c  difference.cpython-35m-x86_64-linux-gnu.so  difference.py  main.py
(charlas) tzulberti@laburo ~/workspace/charlas/meetup-pyar-2018-cython/cython-first-version (master) $ █
```

Usando el archivo cythonizado

```
(charlas) tzulberti@laburo ~/workspace/charlas/meetup-pyar-2018-cython/cython-first-version (master) $ python
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import difference
>>> dir(difference)
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'levenshtein']
>>> difference.levenshtein("a", "b")
1
>>> difference.__file__
'/home/tzulberti/workspace/charlas/meetup-pyar-2018-cython/cython-first-version/difference.py'
>>> █
```

Archivos Generados por Cython

Cuando se corre **cythonize** comando aparecen dos nuevos archivos

- difference.c es el archivo generado con código C generado a partir del código python generado por Cython
- difference.so o difference.cython.*.so: el sharedlibrary compilado a partir del difference.c

Codigo generado por Cython

```
*      size_y = len(seq2) + 1
*      matrix = [[0] * size_y for _ in range(size_x)]          # <<<<<<<<<<<<<<<<
*
*      for x in range(size_x):
*/
__pyx_t_2 = PyList_New(0); if (unlikely(!__pyx_t_2)) __PYX_ERR(0, 8, __pyx_L1_error)
__Pyx_GOTREF(__pyx_t_2);
__pyx_t_3 = __Pyx_PyObject_CallOneArg(__pyx_builtin_range, __pyx_v_size_x); if (unlikely(!__pyx_t_3)) __PYX_ERR(0, 8, __pyx_L1_error)
__Pyx_GOTREF(__pyx_t_3);
if (likely(PyList_CheckExact(__pyx_t_3)) || PyTuple_CheckExact(__pyx_t_3)) {
    __pyx_t_4 = __pyx_t_3; __Pyx_INCREF(__pyx_t_4); __pyx_t_1 = 0;
    __pyx_t_5 = NULL;
} else {
    __pyx_t_1 = -1; __pyx_t_4 = PyObject_GetIter(__pyx_t_3); if (unlikely(!__pyx_t_4)) __PYX_ERR(0, 8, __pyx_L1_error)
    __Pyx_GOTREF(__pyx_t_4);
    __pyx_t_5 = Py_TYPE(__pyx_t_4)->tp_iternext; if (unlikely(!__pyx_t_5)) __PYX_ERR(0, 8, __pyx_L1_error)
}
__Pyx_DECREF(__pyx_t_3); __pyx_t_3 = 0;
for (;;) {
    if (likely(!__pyx_t_5)) {
        if (likely(PyList_CheckExact(__pyx_t_4))) {
            if (__pyx_t_1 >= PyList_GET_SIZE(__pyx_t_4)) break;
            #if CYTHON_ASSUME_SAFE_MACROS && !CYTHON_AVOID_BORROWED_REFS
            __pyx_t_3 = PyList_GET_ITEM(__pyx_t_4, __pyx_t_1); __Pyx_INCREF(__pyx_t_3); __pyx_t_1++; if (unlikely(0 < 0)) __PYX_ERR(0, 8, __pyx_L1_error)
            #else
            __pyx_t_3 = PySequence_ITEM(__pyx_t_4, __pyx_t_1); __pyx_t_1++; if (unlikely(!__pyx_t_3)) __PYX_ERR(0, 8, __pyx_L1_error)
            __Pyx_GOTREF(__pyx_t_3);
            #endif
        }
```


Comparacion de Performance

Cantidad de comparaciones	C Puro	Python Puro	Cython compilando codigo	X mas lento que C	X mas rapido que python puro
233271	0.061	15.174	8.978	147	1.69
116635	0.030	9.007	5.720	190	1.57
77757	0.019	5.605	3.199	168	1.75
58317	0.011	4.010	2.798	254	1.43
46654	0.009	3.796	1.476	164	2.57
38878	0.008	2.558	1.703	212	1.5
33324	0.009	2.066	1.328	147	1.55
29158	0.008	2.029	1.052	131	1.92

Ayudando a Cython

Le podemos indicar el tipo de las variables a Cython para que pueda generar código más óptimo. Hay dos formas:

- PXD
- Decoradores

```
import cython

@cython.locals(
    seq1=str,
    seq2=str,
    matrix=list,
    size_x=cython.int,
    size_y=cython.int,
    x=cython.int,
    y=cython.int,
)
def levenshtein(seq1, seq2):
    size_x = len(seq1) + 1
    size_y = len(seq2) + 1
    matrix = [[0] * size_y for _ in range(size_x)]
```

Comparacion Performance

Cantidad de comparaciones	C Puro	Python Puro	Cython	Cython con tipos	X veces más lento que C	X veces más rápido que python
233271	0.061	15.174	8.978	3.967	65	3.82
116635	0.030	9.007	5.720	2.444	81	3.68
77757	0.019	5.605	3.199	1.555	81	3.64
58317	0.011	4.010	2.798	1.353	123	2.96
46654	0.009	3.796	1.476	1.086	120	3.49
38878	0.008	2.558	1.703	0.713	89	3.58
33324	0.009	2.003	1.328	0.696	77	2.87
29158	0.008	2.536	1.052	0.599	70	4.53

Buscando el cuello de botella

```
In [7]: p.sort_stats('tottime').print_stats(10)
```

```
Wed Aug 15 22:54:08 2018      cprofile.output
```

```
87733 function calls in 0.601 seconds
```

```
Ordered by: internal time
```

```
List reduced from 81 to 10 due to restriction <10>
```

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.462	0.462	0.596	0.596	main.py:17(do_logic)
29158	0.064	0.000	0.098	0.000	main.py:13(<lambda>)
29158	0.037	0.000	0.037	0.000	{method 'split' of 'str' objects}
29158	0.034	0.000	0.034	0.000	{method 'strip' of 'str' objects}
1	0.004	0.004	0.004	0.004	{method 'readlines' of '_io._IOBase' objects}
1	0.000	0.000	0.000	0.000	{built-in method _imp.create_dynamic}
76	0.000	0.000	0.001	0.000	/home/tzulberti/envs/charlas/lib/python3.5/codecs.py:318(decode)
76	0.000	0.000	0.000	0.000	{built-in method _codecs.utf_8_decode}
1	0.000	0.000	0.601	0.601	main.py:9(main)
1	0.000	0.000	0.000	0.000	{built-in method io.open}

```
- - -
```

```
- - -
```

CYTHON ALL THE THINGS



imgflip.com

Comparacion Performance

Cantidad de comparaciones	C Puro	Python Puro	Cython	Cython con tipos	Cython todo el código	X veces más lento que C	X veces más rápido que python
233271	0.061	15.174	8.978	3.967	4.304	3.52	70.5
116635	0.030	9.007	5.720	2.444	2.517	3.57	83.9
77757	0.019	5.605	3.199	1.555	1.395	4.01	73.4
58317	0.011	4.010	2.798	1.353	1.050	3.81	95.4
46654	0.009	3.796	1.476	1.086	0.690	5.5	76.6
38878	0.008	2.558	1.703	0.713	0.635	4.02	73.3
33324	0.009	2.003	1.328	0.696	0.574	3.48	63.7
29158	0.008	2.536	1.052	0.599	0.423	5.99	52.8



Haciendo cosas locas

```
@cython.locals(
    cseq1 = 'const char *', cseq2 = 'const char *', pmatrix = 'int[:,:]')
def levenshtein(seq1, seq2):
    size_x = len(seq1) + 1
    size_y = len(seq2) + 1
    pmatrix = matrix = numpy.zeros((size_x, size_y), numpy.int32)

    for x in range(size_x):
        pmatrix[x][0] = x
    for y in range(size_y):
        pmatrix[0][y] = y

    cseq1 = seq1
    cseq2 = seq2

    for x in range(1, size_x):
        for y in range(1, size_y):
            if cseq1[x-1] == cseq2[y-1]:
                substitution_cost = 0
            else:
                substitution_cost = 1

            pmatrix[x][y] = min(
                pmatrix[x-1][y] + 1, # deletion
                pmatrix[x][y-1] + 1, # insertion
                pmatrix[x-1][y-1] + substitution_cost, #substitution
            )

    return pmatrix[size_x - 1][size_y - 1]
```


Comparacion Performance

# de comparaciones	C	Python Puro	Cython	Cython con tipos	Cython todo el codigo	Cosas locas	X veces más lento que C	X veces más rápido que python
233271	0.061	15.174	8.978	3.967	4.304	0.832	18.23	13.63
116635	0.030	9.007	5.720	2.444	2.517	0.502	17.94	16.73
77757	0.019	5.605	3.199	1.555	1.395	0.391	14.33	20.57
58317	0.011	4.010	2.798	1.353	1.050	0.291	13.78	26.54
46654	0.009	3.796	1.476	1.086	0.690	0.230	16.5	25.55
38878	0.008	2.558	1.703	0.713	0.635	0.227	11.26	28.3
33324	0.009	2.003	1.328	0.696	0.574	0.221	9.06	24.55
29158	0.008	2.536	1.052	0.599	0.423	0.203	12.49	25.375

Cosas que me gustan de Cython

En caso de error, tira excepciones a la gran python, en vez de tirar Segmentation Fault (una gran mejora).

Se puede borrar el *.so y seguimos corriendo python en modo puro lo que nos permite debuggearlo con herramientas de Python

jampp

