menti.com code 1379 5127

Why Knowing Cython Helps in Understanding Python:

A Deep Dive into Cython

menti.com code 1379 5127

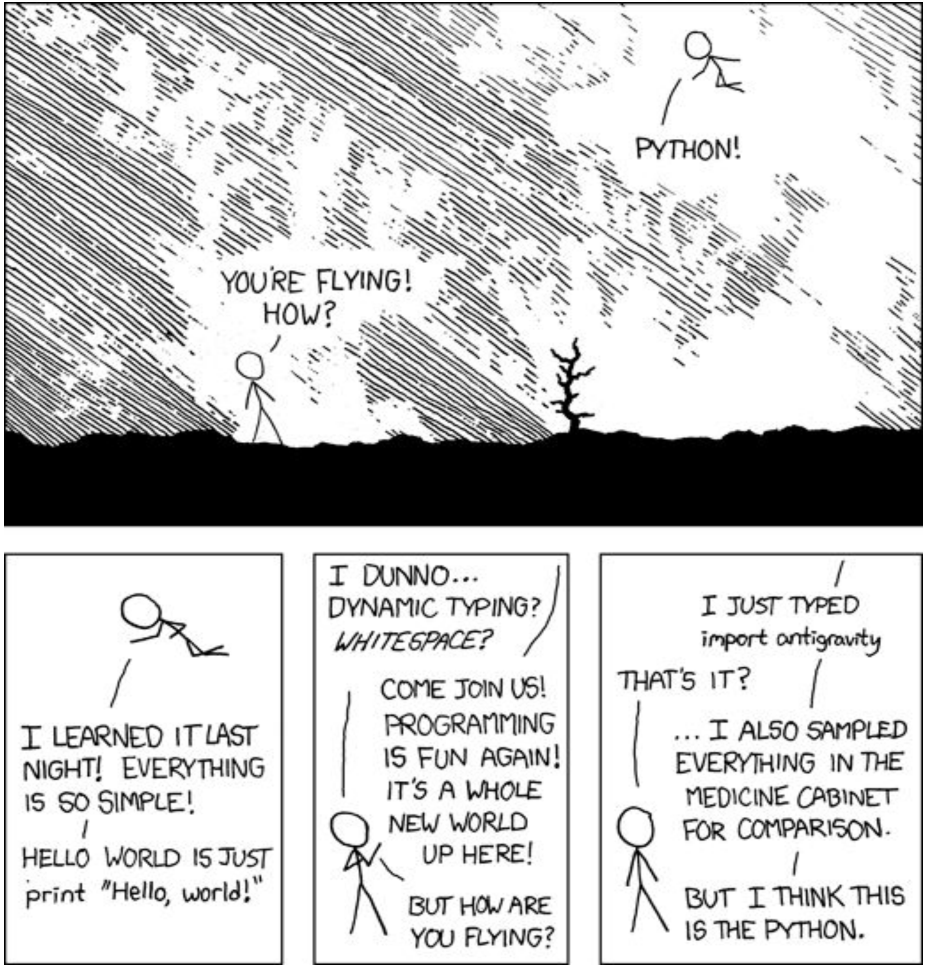
Summary

The more precise the mind, the more precise the decision.

- Human

The more precise the data type, faster the code.

- Compiler


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

source:xkcd

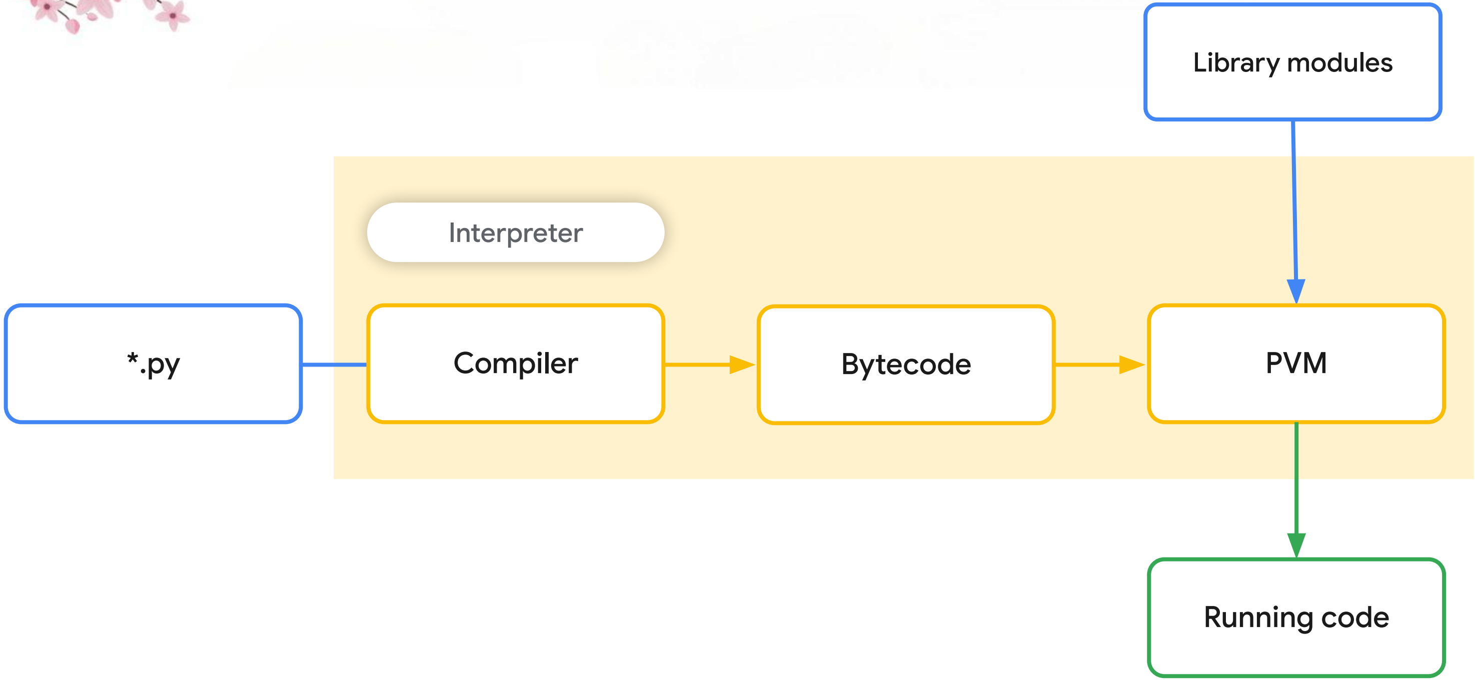
menti.com code 1379 5127

Cython CPython Python

Python	CPython	Cython
1.High-level	1.Reference implementation of Python	1. Super set of python,compile to C code.
2.Interpreted	2.C programming used	2 .Used to write high performance python extension modules and speeds up Python code
3.Used for web development, scientific computing, data analysis, etc.	3.Used in implementation of python	3.Allows python-like code to be optimized as C
4.Simple,readable, and flexible	4.Free,open source software project.	

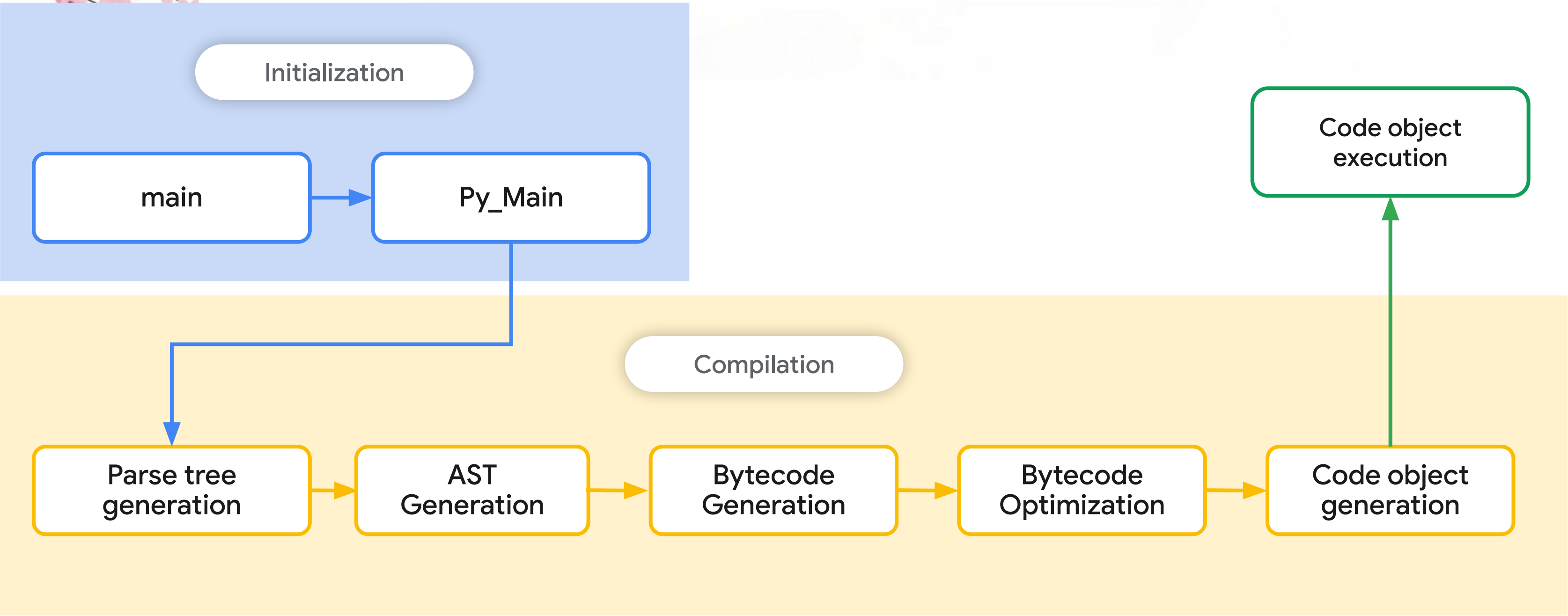
menti.com code 1379 5127

Python Process Chart



menti.com code 1379 5127

Cython Process Chart



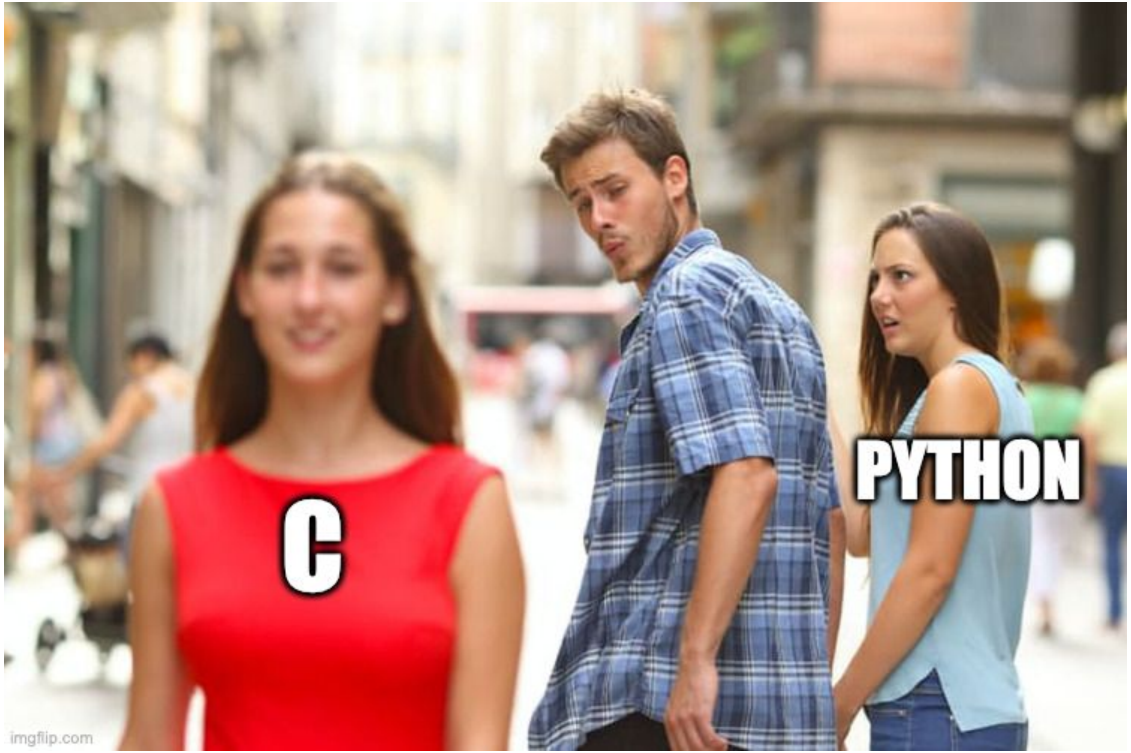
menti.com code 1379 5127

Why Combine Them?

Difference between them is complementary.

C	Python
1. **Static** types	1. **Dynamic** types
2. **Compiled**	**2. Interpreted**
3. **Hard** to read	3. **Easy** to read
4. Very fast and **low level**	4. Very **high level**

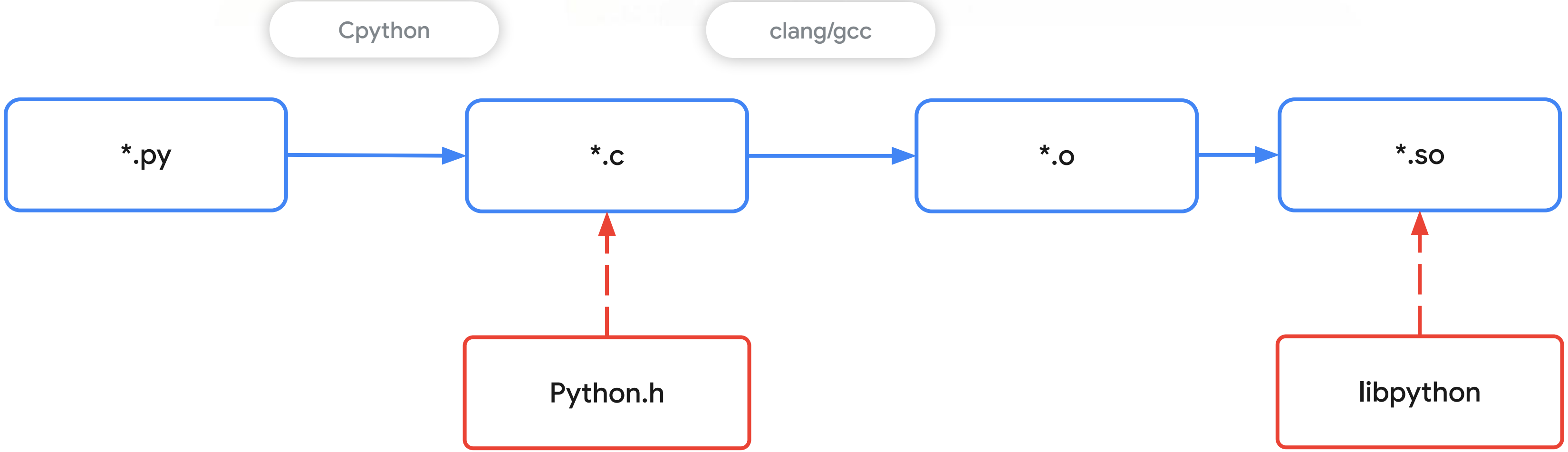
menti.com code 1379 5127



Deep Dive Into Cython Code

menti.com code 1379 5127

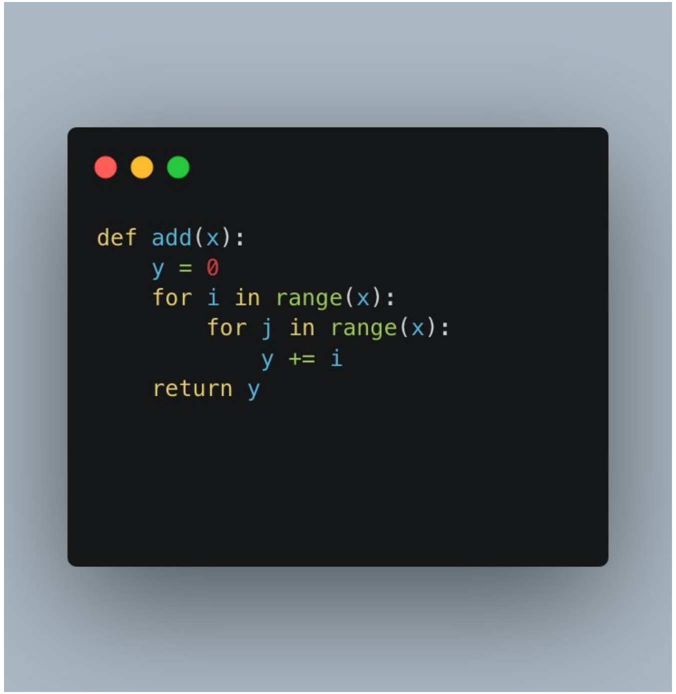
How Cython works?



menti.com code 1379 5127

PYX Version 1

Changing py to pyx



```
def add(x):
    y = 0
    for i in range(x):
        for j in range(x):
            y += i
    return y
```

menti.com code 1379 5127

PYX Version 2

cdef int i

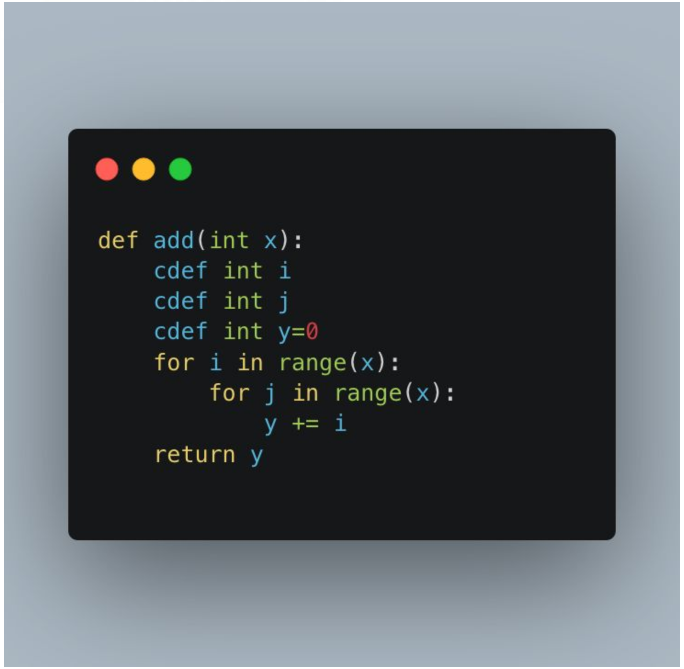
cdef double amount

cdef list modified_particles

cdef dict names_from_particles

cdef str pname

cdef set unique



```
def add(int x):
    cdef int i
    cdef int j
    cdef int y=0
    for i in range(x):
        for j in range(x):
            y += i
    return y
```

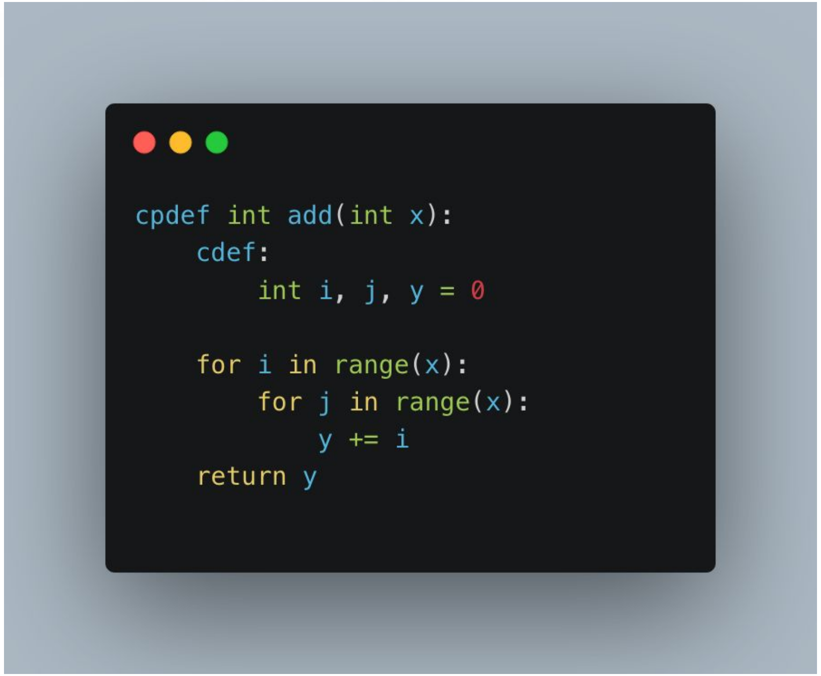
menti.com code 1379 5127

PYX Version 3

cpdef

Python and Cython code can both call a function defined with cpdef.

It allows you to write performance-optimized code in Cython, while still being able to call it from Python.



```
cpdef int add(int x):
    cdef:
        int i, j, y = 0

    for i in range(x):
        for j in range(x):
            y += i
    return y
```

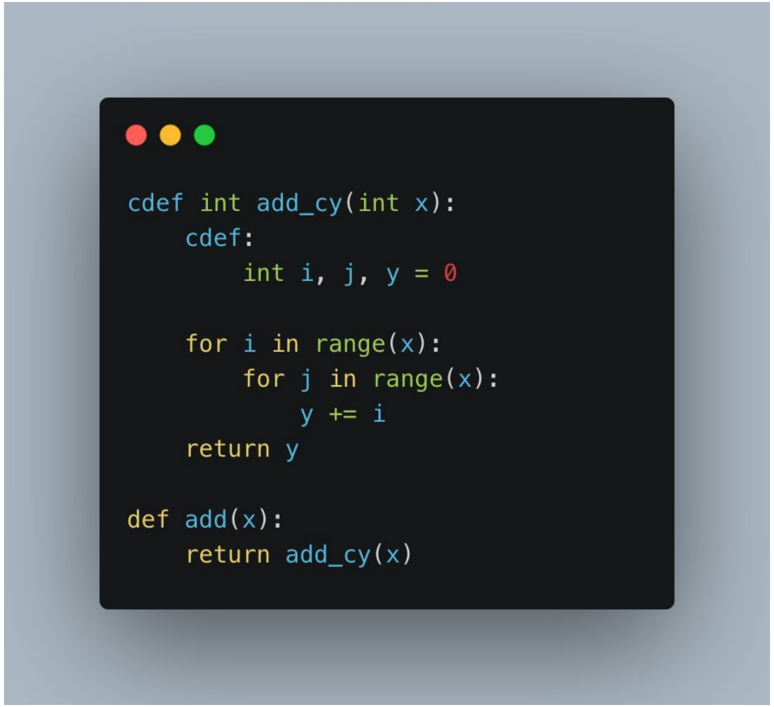
menti.com code 1379 5127

PYX Version 4

cdef

Only Cython code can call a function defined with cdef.

It is in pure C and having no Python Objects



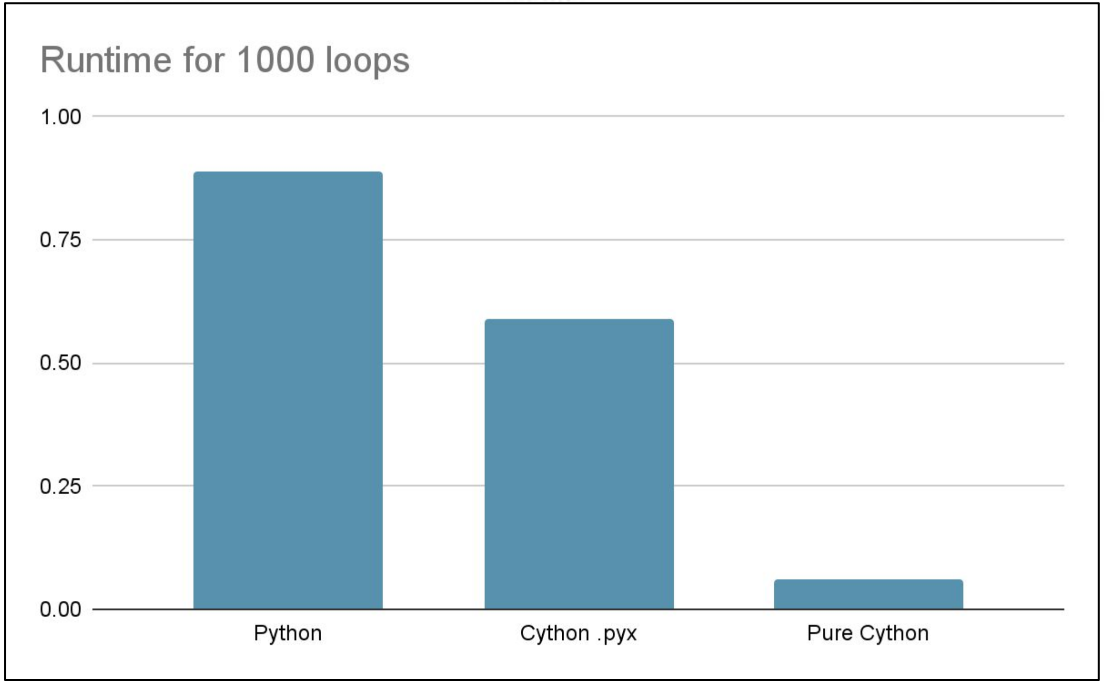
```
cdef int add_cy(int x):
    cdef:
        int i, j, y = 0

    for i in range(x):
        for j in range(x):
            y += i
    return y

def add(x):
    return add_cy(x)
```

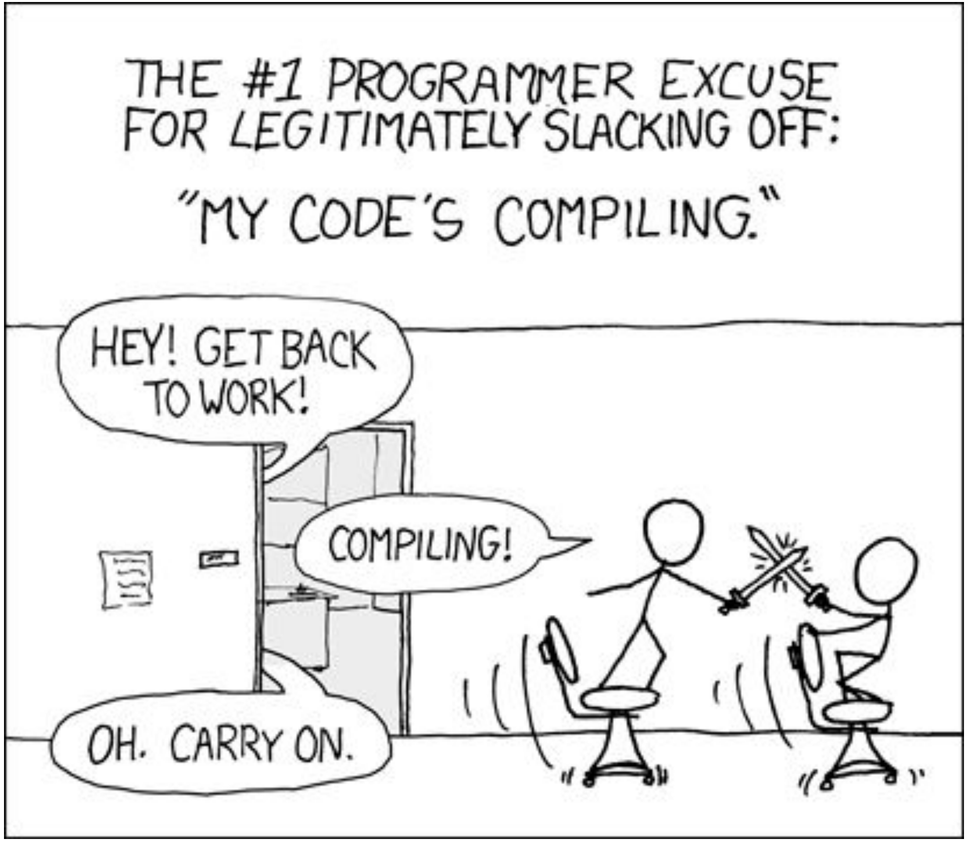
menti.com code 1379 5127

CODE DEMO

menti.com code 1379 5127

Performance Stats





THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.

source:xkcd

menti.com code 1379 5127

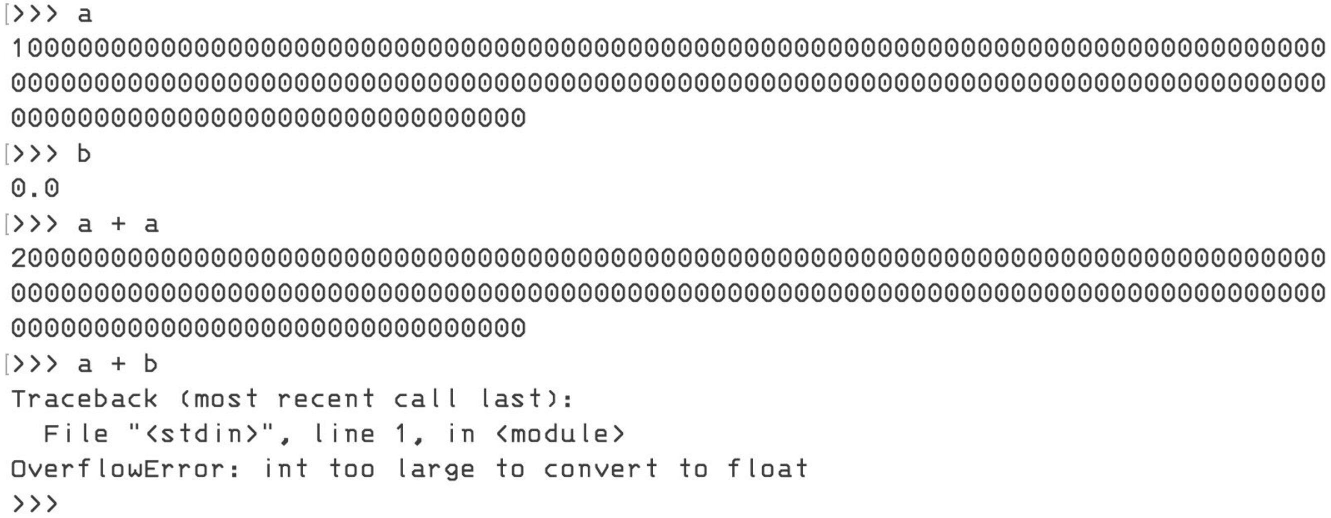
Compilation of Cython

- Manual
- iPython
- Distutils
- Setuptools
- pyximport

But Why Cython is Faster?

menti.com code 1379 5127

Python go home you are drunk !?



```
[>>> a
1000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000
[>>> b
0.0
[>>> a + a
2000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000
[>>> a + b
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
OverflowError: int too large to convert to float
>>>
```

menti.com code 1379 5127

Data Structures in Python

- In Python Everything is an Object, this approach has both pros and cons.

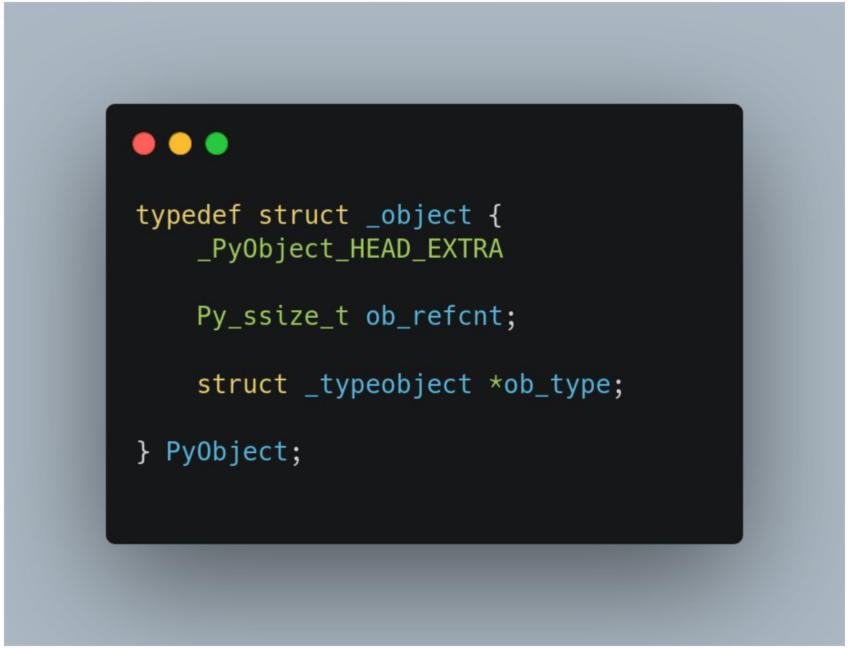
- We will focus on Cons here
  - They are dynamically typed and have a lot of overhead due to Python's dynamic type system.
  - Operations on Python numbers are slower because they involve looking up the type information and executing more bytecode

PyObjects

_PyObject_HEAD_EXTRA : Additional information, used internally by CPython during debugging.

ob_refcnt : Reference count of the object

***ob_type** : Pointer to the object's type information



```c
typedef struct _object {
    _PyObject_HEAD_EXTRA

    Py_ssize_t ob_refcnt;

    struct _typeobject *ob_type;

} PyObject;
```

menti.com code 1379 5127

Python Addition

HTML

menti.com code 1379 5127

Cython Datastuctures

- Cython numbers, such as cdef int, cdef double, are statically typed and are directly mapped to C data types (like int, double). This removes the overhead associated with Python objects.
- Operations on Cython numbers are significantly faster because they translate directly to C-level operations, which are much closer to the hardware and require fewer instructions.
- Cython numbers, being C types, do not have the overhead of Python's memory management. They are allocated and deallocated directly on the stack (or heap, in the case of arrays) like in C.
- This makes memory usage more predictable and reduces the overall memory footprint, especially in numerical computations.

menti.com code 1379 5127

Cython GIL

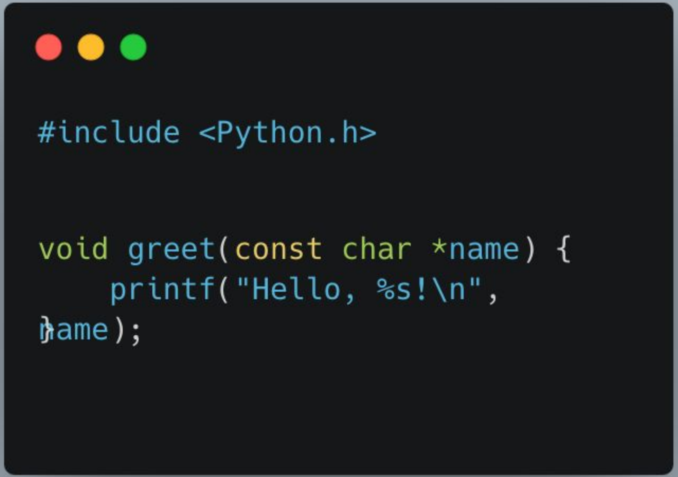
[HTML](#) [HTML](#)

menti.com code 1379 5127

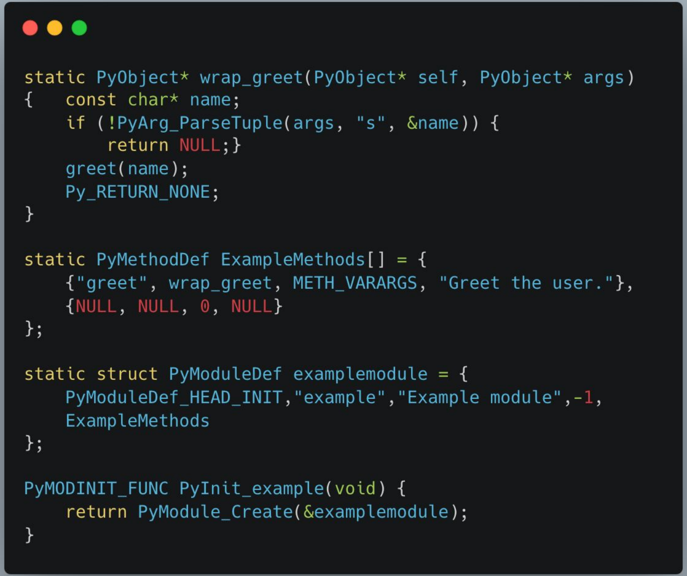
CODE DEMO

menti.com code 1379 5127

Importing C inside Python



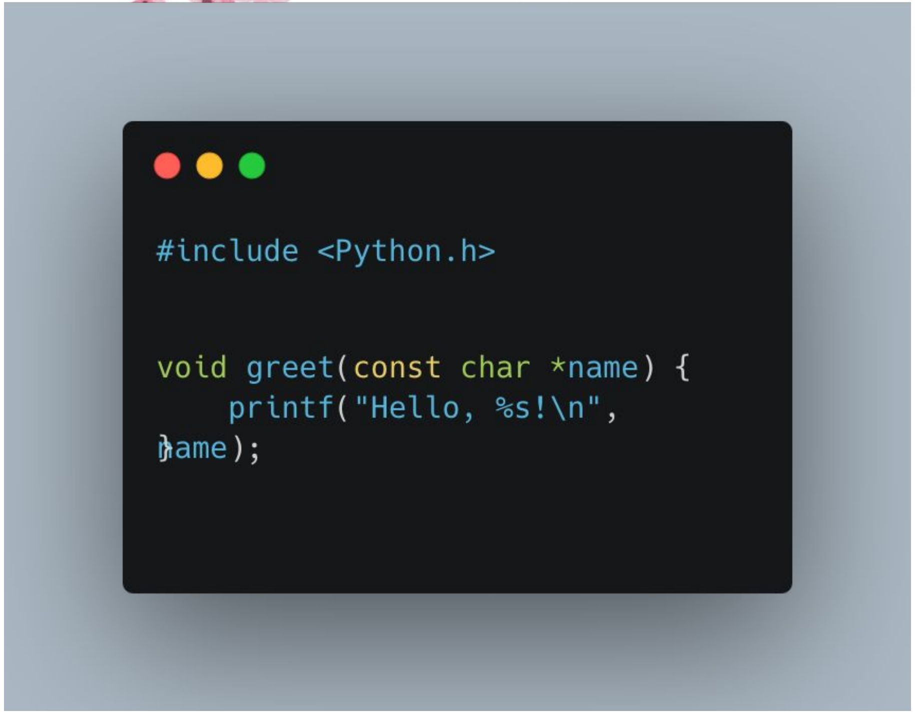
```c
#include <Python.h>


void greet(const char *name) {
    printf("Hello, %s!\n",
name);
}
```



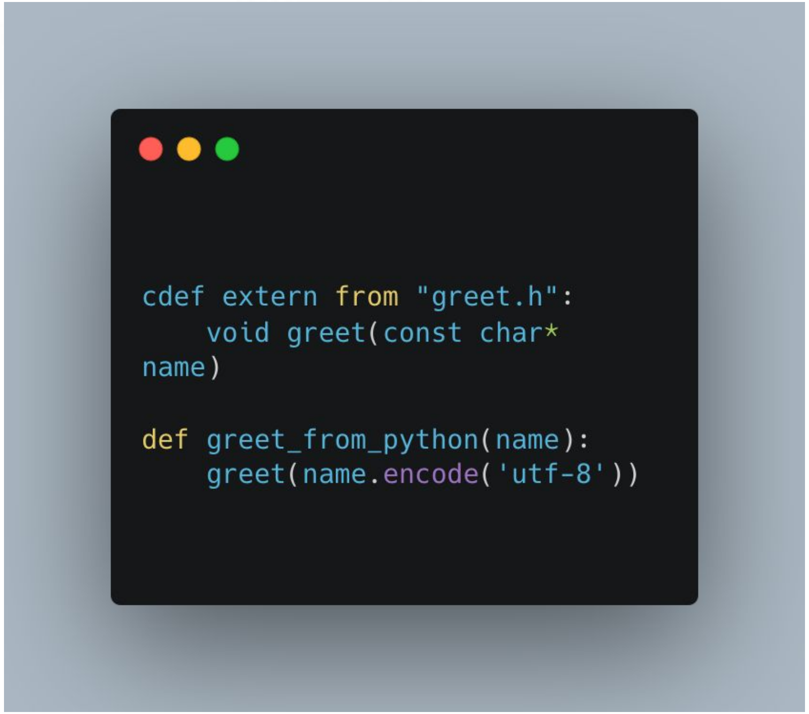
```c
static PyObject* wrap_greet(PyObject* self, PyObject* args)
{   const char* name;
    if (!PyArg_ParseTuple(args, "s", &name)) {
        return NULL;}
    greet(name);
    Py_RETURN_NONE;
}

static PyMethodDef ExampleMethods[] = {
    {"greet", wrap_greet, METH_VARARGS, "Greet the user."},
    {NULL, NULL, 0, NULL}
};

static struct PyModuleDef examplemodule = {
    PyModuleDef_HEAD_INIT,"example","Example module",-1,
    ExampleMethods
};

PyMODINIT_FUNC PyInit_example(void) {
    return PyModule_Create(&examplemodule);
}
```

menti.com code 1379 5127

Importing C inside Python: Cython Edition



```
#include <Python.h>


void greet(const char *name) {
    printf("Hello, %s!\n",
name);
}
```



```
cdef extern from "greet.h":
    void greet(const char*
name)

def greet_from_python(name):
    greet(name.encode('utf-8'))
```

menti.com code 1379 5127

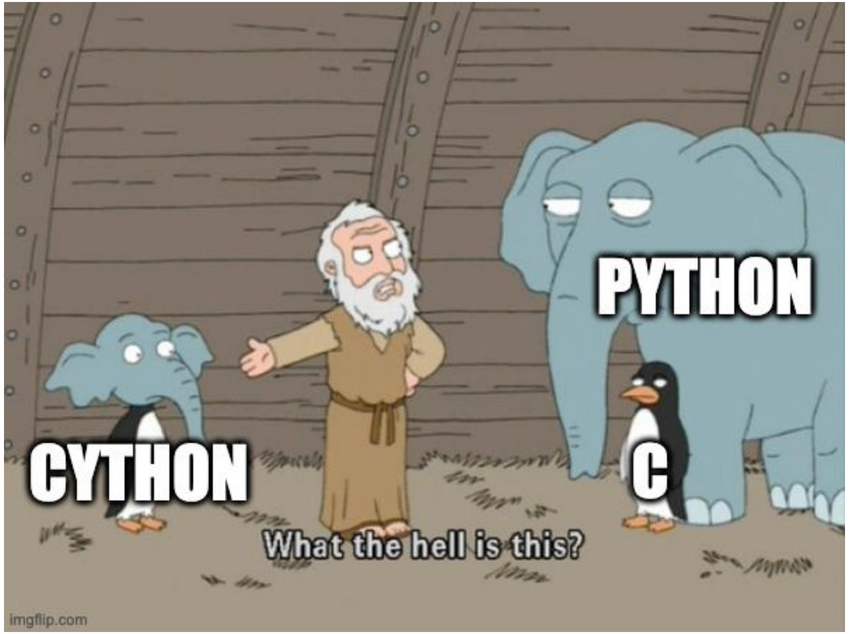
Who Uses This?

scikit
learn
pandas
SciPy

menti.com code 1379 5127

Conclusion

- Cython bridges the gap between Python's ease and C's speed.
- It's a powerful tool for optimizing performance-critical Python code.
- Cython's ability to interface with C libraries extends Python's capabilities.



menti.com code 1379 5127

website: abhik.xyz

@abhiksark in X

/abhiksark in github

/in/abhiksark in linkedin