СОЕДИНЯЕМ C++ И PYTHON

ИГОРЬ САДЧЕНКО

IGOR.SADCHENKO@GMAIL.COM

WARGAMING.NET

МОТИВАЦИЯ

ЗАЧЕМ

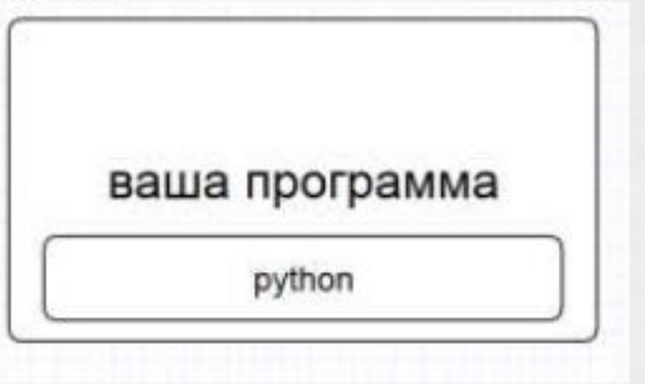
WARGAMING.NET
LET'S BATTLE

- Использование как предметно-ориентированный язык для конкретной области применения(DSL)
- Высокоуровневая разработка(VHLL)
- Быстрое разработка и прототипирование приложений(RAD)
- Встраивание интерактивной среды (REPL)
- Разработка графического интерфейса пользователя (GUI)
- Тестирование

СПОСОБЫ

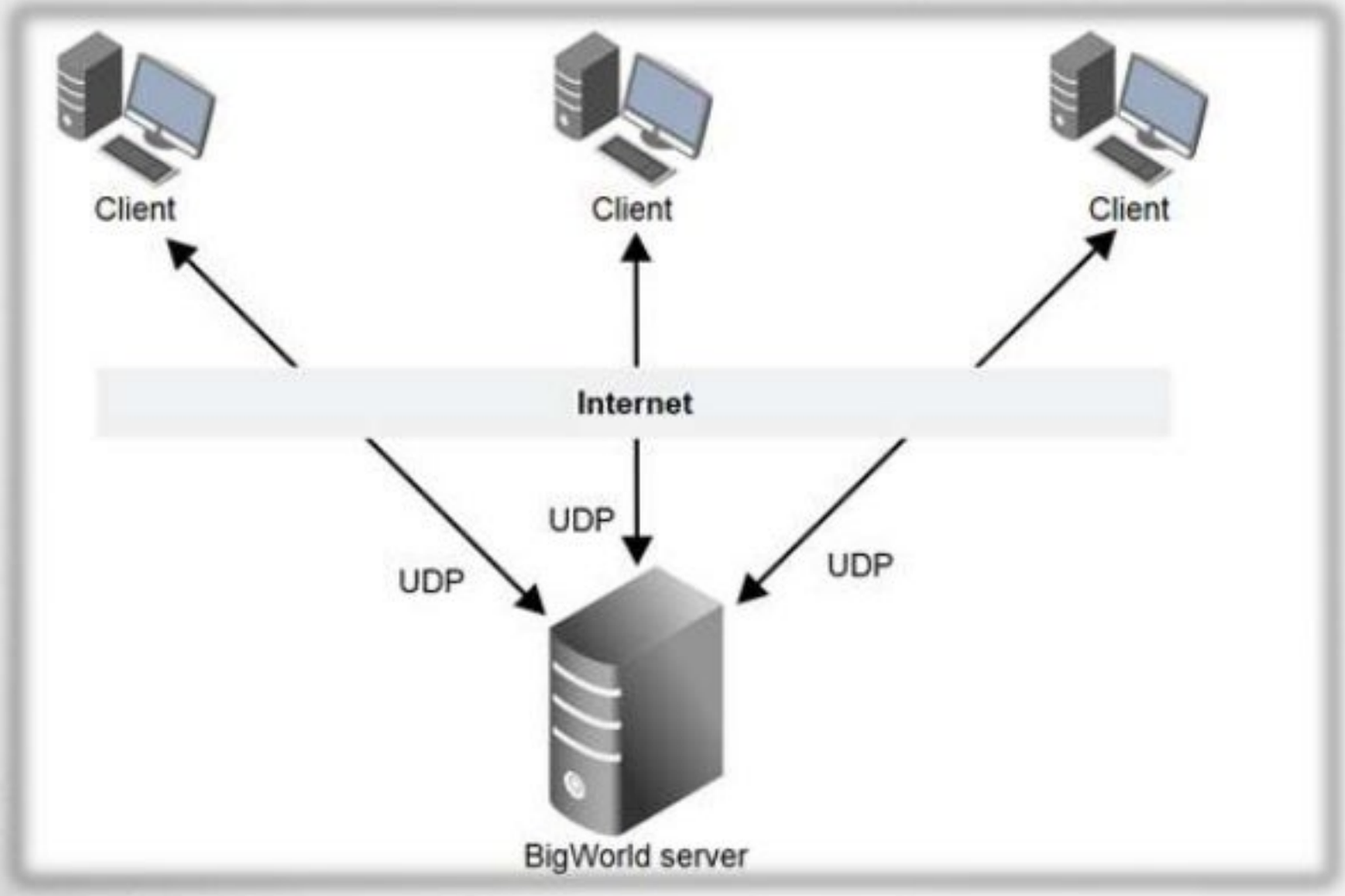
WARGAMING.NET
LET'S BATTLE

- Extensions. Написание расширений для Python на C++
  - улучшение производительности
  - портирование существующих библиотек
  - интеграция различных компонентов
- Embedding. Встраивание Python в C++ программы
  - упрощение разработки
  - дополнительных возможности



НАШ ПУТЬ

WARGAMING.NET
LET'S BATTLE



bigWORLD

O PYTHON

СПРАВКА

WARGAMING.NET
LET'S BATTLE

Python — это свободный интерпретируемый объектно-ориентированный расширяемый встраиваемый язык программирования очень высокого уровня.

- свободный — все исходные тексты интерпретатора и библиотек доступны для любого, включая коммерческое, использования;
- интерпретируемый — поэтому кроссплатформенный, имеет рефлексию;
- объектно-ориентированный — классическая ОО модель, включая множественное наследование;
- расширяемый — имеет API для создания модулей, типов и классов на C, C++;
- встраиваемый — имеет API для встраивания интерпретатора в другие программы;
- очень высокого уровня — динамическая типизация, встроенные типы данных высокого уровня, классы, модули, механизм исключений.

СПРАВКА

WARGAMING.NET
LET'S BATTLE

Python ориентирован на повышение производительности разработчика и читаемости кода.

В Python есть дзен (*import this*), описывающий общие подходы к разработке.

Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python активно развивается и имеет огромное community.

К недостаткам относят низкую производительность и отсутствие реальной многопоточности(GIL)

СПРАВКА

WARGAMING.NET
LET'S BATTLE

Python — язык универсальный, он широко используется во всем мире для самых разных целей — базы данных и обработка текстов, встраивание интерпретатора в игры, программирование GUI и быстрая разработка приложений(RAD).

И, конечно же, Python используется для программирования web-приложений — серверных, клиентских, web-серверов и серверов приложений.

Python и приложения, написанные на нем, используют самые известные и крупные фирмы — IBM, Yahoo!, Google.com, Hewlett Packard, NASA, Red Hat, Microsoft.

СРАВНЕНИЕ C++ и PYTHON

WARGAMING.NET
LET'S BATTLE

C++	Python
компиляция	интерпретация
статическая типизация	динамическая типизация
строгая типизация	строгая типизация
сложный для изучения?	лёгкий для изучения?
эффективное управление памятью	сборщик мусора
есть стандартная библиотека	мощная стандартная библиотека
C++xx…;)	рефлексия и интроспекция
высокая производительность кода	высокая производительность разработчика

СРАВНЕНИЕ C++ И PYTHON

WARGAMING.NET
LET'S BATTLE

```cpp
#include <iostream>
#include <vector>
using namespace std;

vector<int>::const_iterator binarySearch(const vector<int>& container, int element)
{
    const auto endIt = end(container);
    auto left = begin(container);
    auto right = endIt;

    if (container.size() == 0
        || container.front() > element
        || container.back() < element) {
        return endIt;
    }

    while (distance(left, right) > 0) {
        const auto mid = left + distance(left, right) / 2;
        if (element <= *mid)
            right = mid;
        else
            left = mid + 1;
    }

    if (*right == element)
        return right;

    return endIt;
}

int main()
{
    const vector<int> vec = { 0,1,2,3,4,5,6,7,8,9,10 };
    const int element = 12;
    auto foundIt = binarySearch(vec, element);
    if (foundIt != vec.end())
        cout << *foundIt << endl;

    return 0;
}
```

```python
def binarySearch(lst, x):
    l = 0
    r = len(lst)
    while r - l > 1:
        m = (l + r) / 2
        if x < lst[m]:
            r = m
        else:
            l = m
    return l if lst[l] == x else None

def main():
    lst = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    x = 8
    print(binarySearch(lst, x))

if __name__ == "__main__":
    main()
```

ПОЛОЖИТЕЛЬНЫЕ СТОРОНЫ

WARGAMING.NET
LET'S BATTLE

1. C++:
   a. Производительность
   b. Эффективная работа с ресурсами
2. Python:
   a. Скорость разработки
   b. Разнообразие доступных решений
   c. Лёгкость изучения

WARGAMING.NET
LET'S BATTLE

ИНТЕГРАЦИЯ

РЕШЕНИЯ

- Python
- ctypes – библиотека Python для импорта функций из внешних библиотек
- cffi(C Foreign Function Interface) - механизм для импорта функций из внешних библиотек, основанный использовании на C-подобных деклараций

РЕШЕНИЯ

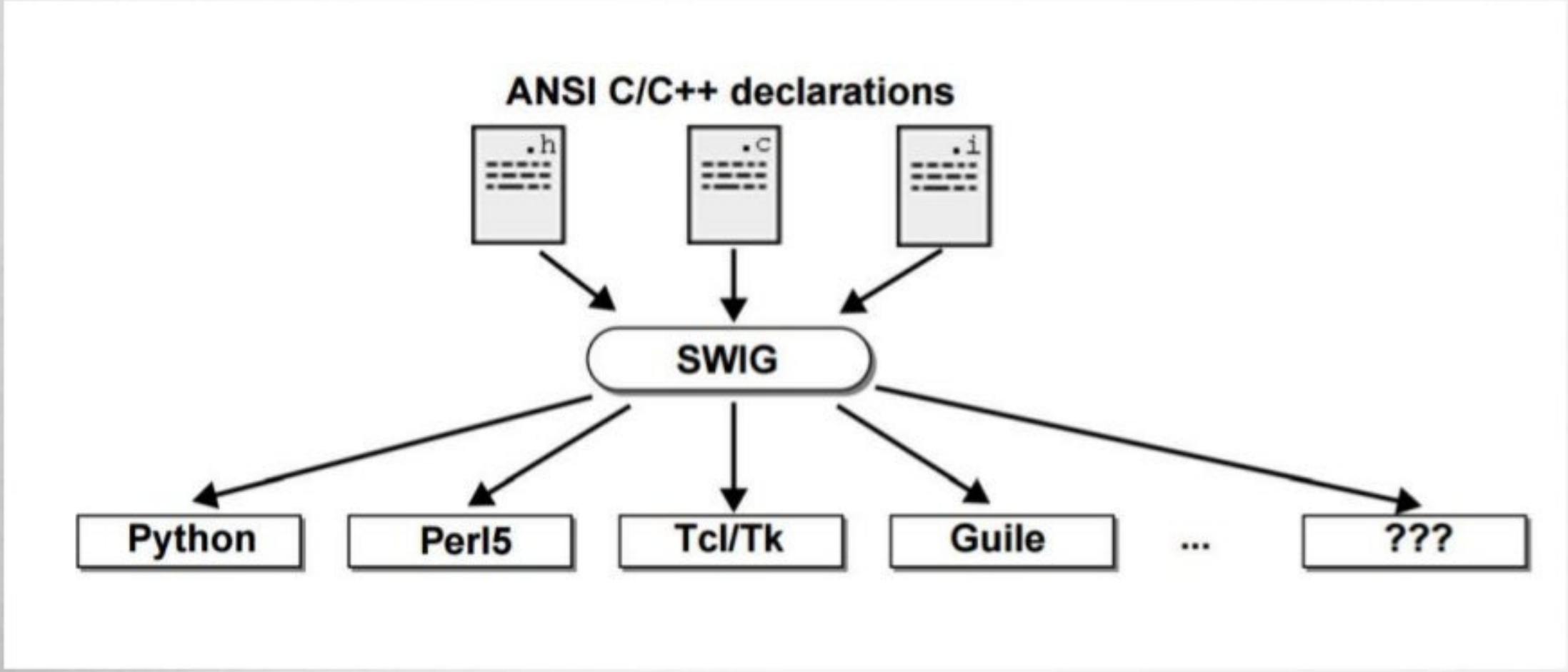
- Не совсем Python
- Cython- преобразуется в C/C++ код для последующей компиляции и впоследствии может использоваться как расширение стандартного Python или как независимое приложение со встроенной библиотекой выполнения Cython.
- RPython (Restricted Python) - ограниченное подмножество Python, статически типизирован, транслируется в языки C, Java, CIL

РЕШЕНИЯ

WARGAMING.NET
LET'S BATTLE

- SWIG(**S**implified **W**rapper and **I**nterface **G**enerator)
- Без зависимостей (для запуска нужен один .exe), не нужны целевые языки
- Доступен на многих платформах
- Поддерживает структуры C и классы C++

РЕШЕНИЯ



РЕШЕНИЯ

WARGAMING.NET
LET'S BATTLE

- C & C++
- Python C API
- Boost.Python
- Pybind11

Python C API

- Самая тесная интеграция с Python из всех возможных
- Лучшая производительность
- Нужно все время считать ссылки
- Нужно хорошо знать C
- Хорошо документирован
- Объемный
- Опасно использовать не понимая

Python C API (пример)

WARGAMING.NET
LET'S BATTLE

```c
#include <Python.h>

static PyObject* module_function(PyObject *self, PyObject *args) {
    float a, b, c;
    if (!PyArg_ParseTuple(args, "ff", &a, &b))
        return NULL;
    c = a + b;
    return Py_BuildValue("f", c);
}
static PyMethodDef MyMethods[] = {
    { "add", module_function, METH_VARARGS, "Adds two numbers" },
    { NULL, NULL, 0, NULL }
};

PyMODINIT_FUNC initmymath(void) {
    (void)Py_InitModule3("mymath", MyMethods,
        "My doc of mymath");
}
```

```python
import mymath
mymath.add(1, 2)
```

Python C API (пример)

```
static PyTypeObject noddy_NoddyType = {
    PyObject_HEAD_INIT(NULL)
    0, /*ob_size*/
    "noddy.Noddy", /*tp_name*/
    sizeof(noddy_NoddyObject), /*tp_basicsize*/
    0, /*tp_itemsize*/
    0, /*tp_dealloc*/
    0, /*tp_print*/
    _ /*MANY MORE*/
    0, /*tp_str*/
    0, /*tp_getattro*/
    0, /*tp_setattro*/
    0, /*tp_as_buffer*/
    Py_TPFLAGS_DEFAULT, /*tp_flags*/
    "Noddy objects", /* tp_doc */
};


static PyObject *
Noddy_new(PyTypeObject *type, PyObject *args, PyObject
    *kwds) {
    Noddy *self;
    self = (Noddy *)type->tp_alloc(type, 0);
    if (self != NULL) {
        self->first = PyString_FromString("");
        if (self->first == NULL) {
            Py_DECREF(self);
            return NULL;
        }
        self->last = PyString_FromString("");
        if (self->last == NULL) {
            Py_DECREF(self);
            return NULL;
        }
        self->number = 0;
    }
    return (PyObject *)self;
}
```

Python C API (встраивание)

WARGAMING.NET
LET'S BATTLE

```
#include <Python.h>

int main(int argc, char *argv[])
{
    Py_SetProgramName(argv[0]);
    //set PYTHONHOME=C:/python27
    Py_SetPythonHome("C:/Python27");
    Py_Initialize();

    PyRun_SimpleString("print 'Hello World'");

    Py_Finalize();
    return 0;
}
```

Boost.Python

- Позволяет небольшими усилиями экспортировать типы из C++ в Python
- Без сторонних утилит, только C++ компилятор
- Подходит для оборачивания сторонних библиотек без изменения их кода)
- Extensions and embedding

Boost.Python (пример)

WARGAMING.NET
LET'S BATTLE

```
char const* greet(){
       return "hello, world";}
```

```
#include <boost/python.hpp>

BOOST_PYTHON_MODULE(hello_ext) {
    using namespace boost::python;
    def("greet", greet);
}
```

```
>>> import hello_ext
>>> print hello_ext.greet()
   hello, world
```

Boost.Python (пример)

WARGAMING.NET
LET'S BATTLE

```
struct World {
    World(std::string msg): msg(msg) {} // added constructor
    void set(std::string msg_) { this->msg = msg_; }
    std::string greet() { return msg; }
    std::string msg;
};
```

```
class_<World>("World", init<std::string>())
        .def(init<double, double>())
        .def("greet", &World::greet)
        .def("set", &World::set);
```

Pybind11

- Аналог Boost.Python
- Поддерживает STL и callback
- header only

Pybind11

- Аналог Boost.Python
- Активно развивается
- Поддерживает STL и callback
- header only

pybind11

WARGAMING.NET
LET'S BATTLE

```cpp
//pybind11_math.cpp
#include <pybind11/pybind11.h>

int add(int i, int j) {
    return i + j;
}

namespace py = pybind11;

PYBIND11_PLUGIN(pybind11_math) {
    py::module m("pybind11_math");
    m.def("add", &add);
    return m.ptr();
}
```

```cpp
#include <pybind11/pybind11.h>
struct Food {
    float quantity;
};
struct Water {
    float amount;
};
namespace py = pybind11;
PYBIND11_PLUGIN(pybind11_math) {
    py::module m("pybind11_math");
    py::class_<Food>(m, "Food")
        .def(py::init<>())
        .def_readwrite("quantity", &Food::quantity);
    py::class_<Water>(m, "Water")
        .def(py::init<>())
        .def_readwrite("amount", &Water::amount);
}
```

```
>>import mymodule
>>food = mymodule.Food()
>>food.quantity = 3.5
>>print food.quantity
```

ВЫВОДЫ

ВЫВОДЫ

WARGAMING.NET
LET'S BATTLE

- Python можно расширять C++ модулями для увеличения производительности
- Python можно встраивать в C++ программы для упрощения разработки
- Существует библиотеки для выполнения этих действий

СПАСИБО ЗА ВНИМАНИЕ!

ANY QUESTIONS?

IGOR SADCHENKO

software developer

igor.sadchenko@gmail.com

+375 33 642 92 91

https://www.facebook.com/WargamingMinsk

https://www.linkedin.com/company/wargaming-net

wargaming.com