README.md

# Intro

*Цель задания*: у всех заданий общая цель, необходимо поборать страх "копания" в исходниках, приобрести навык сборки и правки крупных проектов, закрепить знания, полученные на лекции. В результате это поможет быстрее разбираться в чужом коде, освежит знания C, подготовит к написаю extension'ов, даст представление о том, где искать источник нетривиальных проблем при разработке.

*Критерии успеха*: критерий успеха у заданий тоже один, должен работать заданный функционал. То есть если это задание про икремент, то в итоге должен собраться интерпретатор, где можно сделать i++.

# Задание

## Opcode

*Задание*: добавляем опкод, совмещающий в себе несколько других опкодов. Взглянем на дизассемблер простой функции, котрая вычисляет числа Фибоначчи.

```
[root@4e71999b346e cpython]$ python
Python 2.7.5 (default, Nov  6 2016, 00:28:07)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> def fib(n): return fib(n - 1) + fib(n - 2) if n > 1 else n
...
>>> import dis
>>> dis.dis(fib)
  1           0 LOAD_FAST                0 (n)
              3 LOAD_CONST               1 (1)
              6 COMPARE_OP               4 (>)
              9 POP_JUMP_IF_FALSE       40
             12 LOAD_GLOBAL              0 (fib)
             15 LOAD_FAST                0 (n)
             18 LOAD_CONST               1 (1)
             21 BINARY_SUBTRACT
             22 CALL_FUNCTION            1
             25 LOAD_GLOBAL              0 (fib)
             28 LOAD_FAST                0 (n)
             31 LOAD_CONST               2 (2)
             34 BINARY_SUBTRACT
             35 CALL_FUNCTION            1
             38 BINARY_ADD
             39 RETURN_VALUE
        >>   40 LOAD_FAST                0 (n)
             43 RETURN_VALUE
```

LOAD_FAST и LOAD_CONST так часто идут вместе

```
LOAD_FAST                0
LOAD_CONST               1
```

или вот

```
LOAD_FAST                0
LOAD_CONST               2
```

Давайте "склеим" их, сэкономим на размере байткода, а может даже и по времени исполнения (стоит проверить). Для этого давайте сделаем свой opcode! В итоге получится как-то так:

```
[root@4e71999b346e cpython]$ ./python
Python 2.7.13+ (default, Jul 14 2017, 16:25:35)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> def fib(n): return fib(n - 1) + fib(n - 2) if n > 1 else n
...
[44740 refs]
>>> import dis
[46032 refs]
>>> dis.dis(fib)
  1           0 LOAD_OTUS               1
              3 COMPARE_OP              4 (>)
              6 POP_JUMP_IF_FALSE       31
              9 LOAD_GLOBAL             0 (fib)
             12 LOAD_OTUS               1
             15 BINARY_SUBTRACT
             16 CALL_FUNCTION           1
             19 LOAD_GLOBAL             0 (fib)
             22 LOAD_OTUS               2
             25 BINARY_SUBTRACT
             26 CALL_FUNCTION           1
             29 BINARY_ADD
             30 RETURN_VALUE
        >>   31 LOAD_FAST               0 (n)
             34 RETURN_VALUE
[46059 refs]
>>>
```

*Подсказка*: придется поменять Include/opcode.h, Lib/opcode.py, Python/peephole.c, Python/ceval.c, opcode_targets.h. В peephole.c нужно найти место, где можно в случае если мы видим LOAD_FAST с аргументом 0 со следующим за ним LOAD_CONST, заменить последний на наш opcode, а пространство до этого забить NOP'ами.

### Until

*Задание*: while и for недостаточно, давайте добавим until! Для этого нужно воспроизвести самостоятельно вот эту статью http://eli.thegreenplace.net/2010/06/30/python-internals-adding-a-new-statement-to-python/.

### Increment/Decrement

*Задание*: после выполнения других заданий в интерпретаторе не хватает, кажется, только инкремента и декремента (++/--). Делаем по материалам этой статьи https://hackernoon.com/modifying-the-python-language-in-7-minutes-b94b0a99ce14

### Ограничения:

- cpython 2.7
- centos 7
  - рекомендую docker, см. code sample ниже

### С чего начать

Пробовать что-то сделать проще и удобнее в докере, чтобы не сломать ничего на своей тачке. В нижепреведенном скрипте настраивается окружение и запускается сборка интерпретатора (в шапке даны инструкции по запуску докера). Процесс разработки такой: меняете код, запускаете make, проверяете.

```
# скачиваем image с 7кой: docker pull centos
# запускаем контейнер и заходим: docker run -ti --rm -v
/Users/s.stupnikov/Coding/docker/cpython:/tmp/bin centos /bin/bash
# контейнер при выходе убьется (--rm), монтируем к нему мапочку с этим скриптом (-v ...) в папку
```

```
/tmp/bin внутри контейнера

#!/bin/bash
set -x
set -e

yum clean all
yum install -y\
    git\
    make\
    gcc-c++\
    vim\
    ssh\

cd /opt
git clone https://github.com/python/cpython.git
cd cpython
git checkout 2.7
./configure --with-pydebug --prefix=/tmp/python
make -j2
```

### Что сдавать:

После того, как функционал заработал, делаем патч https://www.devroom.io/2009/10/26/how-to-create-and-apply-a-patch-with-git/ и коммитим его в гит. Имя патча в зависимости от задания: new_opcode. patch, unitl.patch, inc.patch.

## Deadline

Задание нужно сдать через неделю. То есть ДЗ, выданное в понедельник, нужно сдать до следующего занятия в понедельник. Код, отправленный на ревью в это время, рассматривается в первом приоритете. Нарушение делайна (пока) не карается, но может повлиять на ранжирование при выборе топа студентов при окончании курса, пытаться сдать ДЗ можно до конца курсы. Но код, отправленный с опозданием, когда по плану предполагается работа над более актуальным ДЗ, будет рассматриваться в более низком приоритете без гарантий по высокой скорости проверки

## Обратная связь

Студент коммитит все необходимое в свой github/gitlab репозитарий (пример структуры репозитария https://github.com/s-stupnikov/otus-python-0717 , http://docs.python-guide.org/en/latest/writing/structure/#structure-of-the-repository). Далее необходимо зайти в ЛК, найти занятие, ДЗ по которому выполнялось, нажать “Чат с преподавателем” и отправить ссылку. После этого ревью и общение на тему ДЗ будет происходить в рамках этого чата.