Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления
Кафедра Интеллектуальных информационных технологий

ОТЧЁТ

по лабораторной работе №1
по дисциплине “Проектирование программного обеспечения в информационных системах”

Выполнил:
Р. В. Липский, гр. 121701
Проверил:
Бутрин С. В.

Минск 2023

Постановка задачи

Цель: Изучить основные возможности языка Python для разработки программных систем с интерфейсом командной строки (CLI)

Вариант: реализовать интерпретатор стекового языка программирования.

Реализация

Language reference

Литералы.

Integer — это последовательность десятичных цифр, которая может начинаться с дефиса (-) для обозначения отрицательного целого числа. Когда встречается целое число, оно помещается в стек.

String – строка представляет собой последовательность байтов между двумя двойными кавычками (`"`). Не допускается перевод строки, имеется поддержка Unicode. Доступны следующие управляющие последовательности:

- \n – переход на новую строку
- \r – возврат каретки
- \" – двойные кавычки

Когда интерпретатор встречает строку, она помещается в стек.

Также имеется поддержка литералов для boolean (True/true, False/false), float (5.6, 12.3, …). При обнаружении таких литералов, значение также помещается в стек.

Встроенные операции:

- для манипуляций со стеком:

Имя	**Сигнатура**	**Описание**
`dup`	`a -- a a`	дублирует элемент на вершине стека
`swap`	`a b -- b a`	меняет местами два последних элемента стека
`drop`	`a b -- a`	удаляет элемент на вершине стека
`rot`	`a b c -- c b a`	меняет местами 1-ый и 3-ий элементы стека
`size`	`a -- a s`	помещает на вершину стека количество элементов в нём

- операции сравнения:

Имя	**Сигнатура**	**Описание**
=	`a b -- [a == b : bool]`	помещает в стек результат сравнения последних двух элементов

!=	`a b -- [a != b : bool]`	
>	`a b -- [a > b : bool]`	
<	`a b -- [a < b : bool]`	
>=	`a b -- [a >= b : bool]`	
<=	`a b -- [a <= b : bool]`	

- Вызов Python кода

Имя	**Сигнатура**	**Описание**
pyimport	b [a: string] -- b	импортирует Python модуль
pycall	b [a: string] -- b [c: Object]	вызывает Python код и помещает результат в стек

Управление потоком

If:

```
// -- snip --
<condition> if
   <body>
fi
// -- snip --
```

While:

```
// -- snip --
<start condition> while
   ...contents
<continue condition> do
// -- snip --
```

Include

```
// -- snip --
include package.file
// -- snip --
```

Macros:

```
// -- snip --
macro print-hello-world
   "Hello, world!\n" std.io.popln!
end
// -- snip --
macro main
```

```
   print-hello-world!
end
```

Constants:

```
// -- snip --
const ONEHUNDRED 100
// -- snip --
```

Примеры программ на языке ***slang***

Hello, world:

```

include std.io

macro main using std.io namespaces
   "Hello, world!\n" popln!
end

```

Вывод последовательности Фибоначчи:

```

include std.io

const CAP 0x90

macro main using std.io namespaces

   1 1
   cycle:
       dup rot +
       dup putx!
       " (" pop! dup puti! ")" pop!
       dup CAP! >
       if
           ", " pop!
           cycle~
       fi
   drop drop
   "\n" pop!

end

```

Нахождение корней уравнения методом Ньютона:

```
include std.math
include std.io
include std.list

const _A    -2.
const _B    5.
const _EPS  0.001
const _STP  0.4

macro function using std.math namespaces
   $param swap &
   3 $param @ pypow!       // x^3
   2 $param @ pypow! 5 *   // 5x^2
   12                      // 12
   swap rot + -
end

macro derivative using std.math namespaces
   $param swap &
   2 $param @ pypow! 3 *   // 3x^2
   $param @ 10 *           // 10x
   swap -
end

macro find-root using std.math namespaces
   $near swap &
   $near @ function! abs! _EPS! > if
       $near @ return
   fi
   $near @ derivative! $near @ function! /
   $near @ -
   find-root!
end

macro round-to-eps using std.math namespaces
   10 _EPS! log! abs! round!
   swap dround!
end

macro main using std std.io namespaces

   $i _A! &
```

```
    $counter 0 &

    $answers list.new! &
    True while
        $newroot $i @ find-root! round-to-eps! &
        $newroot @ $answers @ list.contains! not if
            $newroot @ $answers @ list.append!
            $counter dup @ ++ &
        fi

        $i dup @ _STP! + &

        $i @ _B! >=
    do
    $newroot $i delete delete

    $answers @ dup list.sort! list.reverse!
    "D = {" pop!
    True while
        $answers @ list.pop! pop!
        $counter dup @ -- &
        $counter @ 0 < dup
        if ", " pop! fi
    do
    "}, where D - set of roots of x^3-5x^2+12" popln!


end
```