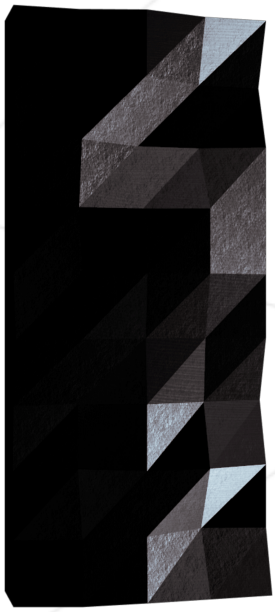

Interpreter101

Create your own language instead of learning one!

Ivan Kozlov ikozlov@student.42.us.org
Kai Drumm kai@42.us.org

Summary: Summary: The aim of this project is to code a simple interpreter. (Don't worry. It's easier then it seams.)

Contents

I Foreword **2**

II Introduction **4**

III Goals **5**

IV General instructions **6**

V Mandatory part **7**

V.1 Small steps in the right direction 8

V.2 Interpreter 8

V.3 Program entry 8

VI Bonus part **10**

VII Turn-in and peer-evaluation **11**

Chapter I

Foreword

In case you missed Geography in school or just want to learn some exciting facts about the world you live in - here are 20 facts about Poland. Read away...

1. The most popular dog's name in Poland is "Burek", which is the Polish word for a brown-grey colour.

2. In Poland, people generally peel bananas from the blossom end and not the stem end.

3. Poland shares its borders with no less than seven countries: Russia, Lithuania, Belarus, Slovakia, Ukraine, the Czech Republic and Germany.

4. Poland has had many capital cities in its time. These have included Gniezno, Poznan, Krakow and Warsaw. Lublin has been the capital twice – after both World Wars.

5. Geographically, Poland is not actually in Eastern Europe – it is in fact in the very centre of Europe(this inaccuracy personally drives me crazy)

6. In Poland, one's "Name Day" – imieniny – is considered a far more important occasion than one's birthday.

7. Poland's national symbol is the white-tailed eagle.(get that US)

8. The name "Poland" originates from the name of the tribe Polanie. This tribe used to inhabit the Western part of what we now call Poland, and originally meant "people living in open fields".

9. The most "World's Strongest Man" winners are from Poland.

10. People of Polish descent have won 17 noble prizes, including four for peace and five for literature.

11. One third of Poland is covered with forest, with 50

12. Rysy, in the Tatra Mountains, is the highest point in Poland, at a height of 2,499 m.

13. Most Poles are Roman Catholics.

14. Roman Catholicism is so popular that Poland has a TV channel dedicated to the Pope.

15. In 1989, Poland held its first free elections in more than 40 years.

16. Bigos is the country's most popular traditional dish. It is a type of stew made from Polish sauerkraut, fresh cabbage, different types of meat, sausage, prunes, dried mushrooms, onions and spices. This is cooked over several days and served with potatoes and bread.

17. Marzanna is a Polish tradition where people weave straw dolls, which they then decorate with ribbons. These dolls represent winter, so when the snow starts to melt the Marzanna dolls are thrown into a river, symbolizing the 'killing' of winter.

18. The Polish are generally well educated, with 90

19. The Polish born astronomer Nicolaus Copernicus is considered to be the first person to propose the theory that the earth was not the centre of the universe.

20. In Poland, pizza bases are not topped with Napolitana or a tomato-based sauce. These are generally served separately and resemble what we would consider to be ketchup.

Usually foreword has no connection to subject whatsoever. But not in this case. What is so polish about this topic? Can you read these facts in reverse?

Chapter II

Introduction

Programming languages were around since 1942. First language was Plankalkül or `Plan Calculus`. It was a simple(not really) programming language which mostly focused on algebra and made calculations a lot easier.

Since then languages have developed to be able to do complex things(such as the operation system you are using right now). But computers haven't changed much. There is still a need to translate the language to commands a machine can understand.

There are two types of languages: **compiling** and **interpreted**. We will be focusing on the second one for several reasons(you don't want to learn assembly, do you?)

Chapter III

Goals

The goal of this project is to introduce you to inner workings of programming languages. Sounds fun? No? Too bad.

In this project you will create you own language. It won't do much, but you have to start somewhere, right? At the very end your language will calculate a mathematical expression, which may include variables.

You also will get familiar with the `Stack` data structure. This is very a common way to organize an array of elements used both in real life and in software development. There are many real life examples of a stack. Consider the simple example of plates stacked over one another in the restaurant. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottom most position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO/FILO order.

```
LIFO stands for Last In First Out
```

For the reasons of common sense and countless others, stack structure t must be implemented using OOP.

```
'P' in OOP stands for programming :)
```

The stack is not much use by itself, but it can be very useful tool to solve certain problems.

Through this project, you will use a stack to convert an expression to a form that a machine will understand and then calculate. Simple? We'll see about that...

Chapter IV

General instructions

- You will use Python 3 in this project(New is always better. True story)
- You are not allowed use any modules except `sys`
- You are free to organize and name your files as you wish, but within the constraints listed here.
- Besides the files you need, you must turn in three files named `stack.py`, `interpreter.py` and `main.py`
- `stack.py` must have an implementation of class Stack
- `interpreter.py` is the main class of your program. It will get the code as text, parse and execute it
- `main.py` must be a main executable, which will calculate an expression and print the result on stdout
- All files must be placed at root of the folder
- In no way you program should exit unexpectedly. You are allowed to use custom exception errors.
- May the Force be with you. Always.

Chapter V

Mandatory part

Your language will have simple and straightforward syntax. Here is the example

```
a = 1
b = 2
c = 3
a + b + c
```

Your test file will contain `n` lines of code. The first `n - 1` lines of this file will be variables. These lines should follow simple syntax:

```
name = value
```

Where `name` is a letter of Latin alphabet and `value` is an integer.

The last line must be an mathematical expression which may contain variables, integers, and math operations. Just like in normal mathematical expression. No tricks.

Your program must support following operations:

- addition - represented by `+`
- subtraction - represented by `-`
- modulo - represented by `%`
- division - represented by `/`
- multiplication - represented by `*`
- power - represented by `^`
- don't forget about parenthesis :)

`Think about order of operations. Multiplication always goes before addition but parenthesis can change the order.`

V.1 Small steps in the right direction

Although creating an interpreter might seem overwhelming and it is. You can start by creating class **Stack**. Trust me you will need it at the end.

`stack.py` must have an implementation of the class `Stack` with following methods:

- push - pushes value on top of the stack
- pop - returns value of top element of the stack and removes it
- peek - returns value of top element of the stack **without** removing it
- isEmpty - returns `true` if stack is empty and `false` otherwise

Here is a skeleton for you:

```
class Stack:
    def __init__(self):
        pass

    def push(self):
        pass

    def pop(self):
        pass

    def peek(self):
        pass

    def isEmpty(self):
        pass
```

V.2 Interpreter

The interpreter is the main class of your program. It will parse your code line by line, store all variables and then execute the last line as a mathematical expression. The variables will be replaced by their values.

V.3 Program entry

The program will take the 1st argument that is passed. This argument represents a path to the file with your program code. In case there are too many arguments(or none) your program should display its usage:

```
> python3 main.py | cat -e
Usage: python3 main.py [path_to_file]
```

All operators and operands **must** be separated by spaces. The only exceptions are negative values like -2.

The program will be correct and **will not** contain any errors.

Here is an example of how you program should work:

```
> cat ../examples/test1
a = 1
b = 2
c = 3
a + b + c

> python main.py ../examples/test1
6
```

Chapter VI

Bonus part

We all like bonuses, don't we? Evaluating an expression is just the first step. You can evaluate everything! Python, for example, is an interpreted language. That means that a special program called an interpreter runs the code and evaluates your commands step by step. You can think about it as a complex expression with support for more operations than mathematical. Here are some ideas to give you a chance to improve your program.

Errors: people aren't perfect but your program can be! Make it check if the input is correct. It can give an error message to a user, explaining what is wrong. You can add colors!

```
> ./main.py error_file | cat -e
[ERROR] Variable 'f' is not defined
```

More operations: Basic math operations are fun but not SUPERFUN. Add a new operation to your program. Like binary operators, or absolute values.

Chapter VII

Turn-in and peer-evaluation

Submit your work on your **git** repository as usual. Only the work on your repository will be graded.