**“Mini Compiler in Gujarati Language”**

**Special Assignment Report**

*Submitted in Partial Fulfillment of the Requirements for completion of*

Course on

# 2CS701 Compiler Construction

By

**Charmi Parmar (18BCE032) Dixit Umaretiya (18BCE060)**



# Department of Computer Science and Engineering, Institute of Technology

## Project Description:

A mini-compiler for a programming language written in Gujarati. This mini-compiler has its own lexical, syntactic and semantic analyzer. It is based mostly on Python, which is also used as an intermediate language to generate the machine language by using python ply.

## Use:

You can start using it by executing the following commands:

* Python parser.py for interpreted mode
* Python parser.py file.dr to compile a file.

## Keywords and equivalent in python:

|  |  |
| --- | --- |
| Keyword | Python |
| Chappo | print |
| Mahiti | input |
| Jo | if |
| Athva | else |
| Karo | do |
| Jya | while |
| Mate | for |
| Bahar | break |
| Chalu | continue |
| Ane | and |
| Ya | or |
| Khotu | false |
| Sachu | true |
| Nahi | not |
| Banavo | def |
| Moklo | return |
| Prayas | try |
| Sivay | except |

|  |  |
| --- | --- |
| Jodo | append |

|  |  |
| --- | --- |
| Hatavo | pop |
| Kul | length |

**Features:**

* Variables
* Comments
* Control flow statements
  + Loop ( while, do-while, for)
  + Conditional statements
  + Function calls
  + Exceptions
* Operators
  + Arithmetic: +, -, \*, /, %, ^
  + Logical: and, or, not

o Comparison: >, >=, <, <=, ==, !=

* + Assignment: =
  + Unary: ++, --

## Data Types:

* Numbers:

You can use either integers or floating points numbers in this language.

* + int: 123
  + float: 1.23
* Strings:

You can choose either 'string' or "string" to represent a string. We can use the slicing operator [ : ] to extract an item or parts of the string:

* + a= "hello world!" chhapo(a[0:5]) hello
* Lists:

A list is created by placing all the items (elements) inside square brackets [ ], separated by commas.

o a = [5,10,15,20,25,30,35,40]

chhapo(a[0:3])

Output: [5, 10, 15]

o List slicing:

my\_list = ['p','r','o','g','r','a','m','i','z']

chhapo(my\_list[2:5]) Output: elements 3rd to 5th

chhapo(my\_list[:-5])

Output: elements beginning to 4th

chhapo(my\_list[5:]) Output: elements 6th to end

chhapo(my\_list[:])

Output: elements beginning to end

o List indexing:

my\_list = ['p', 'r', 'o', 'b', 'e']

chhapo(my\_list[0]) Output: p

chhapo(my\_list[2])

Output: o

n\_list = ["Happy", [2, 0, 1, 5]]

## Operators:

* Arithmetic operators:
  + x = 15 y = 4

chhapo(x+y) Output: 19

chhapo(x-y) Output: 11

chhapo(x\*y) Output: 60

chhapo(x/y)

Output: 3.75

chhapo(x%y) Output: 3

chhapo(x^y) Output: 50625

* Comparison operators
  + x = 10 y = 12

chhapo(x>y) Output: khotu

chhapo(x<y) Output: sachu chhapo(x==y) Output: khotu

chhapo(x!=y) Output: sachu

chhapo(x>=y) Output: khotu

chhapo(x<=y) Output: sachu

* Logical Operators:
  + x = sache y = khotu

chhapo(x ane y) Output: khotu

chhapo(x ya y) Output: sache

## Flow Control:

* jo..athva (if..else):
  + The jo..athva statement evaluates test expression and will execute the body of jo only when the test condition is sachu (True).
  + If the condition is khotu(false), the body of athva is executed.
    - num=5 #or num=-5 jo(num >= 0){

chhapo('The number is positive')

}

athva{

chhapo('The number is negative')

}

* mate (for loop): The mate loop is used to iterate over a sequence.

o mate(i=0;i<5;i++){ chhapo("Iteration:",i)

}

* jya(while loop): The jya loop is used to iterate over a block of code as long as the test expression (condition) is true.
  + n = 10 sum = 0 i=1 jya(i<n){

sum = sum + i i = i+1

}

chhapo("The sum is", sum) Output: The sum is 45

* bahar (break): The bahar statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop.
  + str="string"

mate (a=0;a<tol(str);a++){ jo(str[a] == "i"){

bahar

}

chhapo(str[a])

}

chappo("The end") Output:

#s #t #r

#The end

* chalu(continue): The chalu statement is used to skip the rest of the code inside a loop for the current iteration only. Loop does not terminate but continues on with the next iteration.
  + str="string" mate(a=0;a<tol(str);a++){

jo(str[a] == "i"){ chalu

}

chhapo(str[a])

}

chhapo("The end") Output:

#s #t #r #n #g

#The end

* banavo (function def): A function is a group of related statements that performs a specific task.

## Screenshots:

