**UNIVERSITY OF PUERTO RICO**
**MAYAGÜEZ CAMPUS**
**FACULTY OF ENGINEERING**
**DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING**

# Phase III: Final Report


**By:**
José F. Rodríguez (802-09-7552)
Karla M. Valcárcel Martínez (802-12-8095)
Tamara González Feliciano (801-11-2765)
Ricardo A. Casares Rivas (841-11-1110)

**Dr. Wilson Rivera Gallego**
wilson.riveragallego@upr.edu
ICOM 4036 – Programming Languages

**Due Date**
2 July 2017

# Introduction

The purpose of this project was to create a programming language which university students may use as a helpful tool for some science or engineering classes. With “ConvieLte”, users are able to convert measuring units from one system of measurement to another or convert within the same system, converting between scales of the same type of measurement, following the proper rules of syntax. This may help students do this calculations with ease and efficiency. Students may also use specific functions which represent some equations widely used in the fields of science and engineering. These may be equations to calculate distance, velocity, force, work or voltage, among other types of measurements. Hopefully this programming language is actually useful to help students in the courses they must take.

# “ConvieLte” Tutorial

To be able to use “ConvieLte”, the user must have Python 3.6.0 installed in his or her computer and the python lexer & yacc (PLY) libraries must be inside the project folder. Using any Python compiler (we recommend JetBrains Pycharm), open the folder containing the project and open the Convielte.py file. Start running the program once it opens (Run -> Run Convielte). Once the program starts the user can start performing all of the operations available. These will be explained in the Reference Manual. The user can also type the ‘help’ command in the program to get an overview of each operation and how to use them. The ‘convielte’ function can be used to convert a value measured with a certain type of unit to a value measured with another type of unit (i.e. liters to ounces). It can also be used to change the prefix of a value’s unit (i.e. from kilometers to meters). Other functions are ‘ohmsLaw’, ‘forceEquation’, ‘freefallEquation’, etc., which can be used to calculate certain measures. The user may also be able to assign values to variables and perform simple mathematical operations.

## Reference Manual

The possible types of conversion are:

- convielte<prx>(NUMBER,PREFIX) --> Any number in base format conversion to FEMTO(f), PICO (p), NANO (n), MICRO (u), MILLI (m), CENTI (c), DECI (d), DEKA (da), HECTO (h), KILO (k), MEGA (M), GIGA (G), TERA (T)
- convielte<L>(VALUE,TYPE) --> Any number in base L to oz, gal, m^3
- convielte<oz>(VALUE,TYPE) --> Any number in base oz to L, gal, m^3
- convielte<m>(VALUE,TYPE) --> Any number in base m to in, ft, yd
- convielte<in>(VALUE,TYPE) --> Any number in base in to m, cm, mm, ft, yd
- convielte<ft>(VALUE,TYPE) --> Any number in base ft to in, m, yd
- convielte<yd>(VALUE,TYPE) --> Any number in base yd to in, ft, m
- convielte<rad>(VALUE,TYPE) --> Any number in rad to deg
- convielte<deg>(VALUE,TYPE) --> Any number in deg to rad
- convielte<km>(VALUE,TYPE) --> Any number in km to mi
- convielte<mi>(VALUE,TYPE) --> Any number in mi to km
- convielte<F>(VALUE,TYPE) --> Any number in F to C, K
- convielte<C>(VALUE,TYPE) --> Any number in C to F, K
- convielte<K>(VALUE,TYPE) --> Any number in K to F, C
- convielte<g>(VALUE,TYPE) --> Any number in g to mg, kg, lb
- convielte<lb>(VALUE,TYPE) --> Any number in lb to g, kg, t
- convielte<t>(VALUE,TYPE) --> Any number in t to kg, lb
- convielte<s>(VALUE,TYPE) --> Any number in s to hr, min, day
- convielte<hr>(VALUE,TYPE) --> Any number in hr to s, min, day
- convielte<min>(VALUE,TYPE) --> Any number in min to s, hr, day
- convielte<day>(VALUE,TYPE) --> Any number in day to s, min, hr

The available formulas are:

- ohmsLaw<V>(VALUE OF CURRENT,VALUE OF RESISTANCE) --> Calculates voltage using Ohm's Law (V = i * R)
- ohmsLaw<I>(VALUE OF VOLTAGE,VALUE OF RESISTANCE) --> Calculates current using Ohm's Law (i = V / R)
- ohmsLaw<V>(VALUE OF CURRENT,VALUE OF RESISTANCE) --> Calculates voltage using Ohm's Law (V = i * R)

- ohmsLaw<R>(VALUE OF VOLTAGE,VALUE OF CURRENT) --> Calculates resistance using Ohm's Law (R = V / i)
- electricPotencyVI<P>(VALUE OF VOLTAGE,VALUE OF CURRENT) --> Calculates power (P = V * i)
- electricPotencyVI<I>(VALUE OF POWER,VALUE OF VOLTAGE) --> Calculates current (i = P / V)
- electricPotencyVI<V>(VALUE OF POWER,VALUE OF CURRENT) --> Calculates voltage (V = P / i)
- electricPotencyRV<P>(VALUE OF VOLTAGE SQUARED,VALUE OF RESISTANCE) --> Calculates power (P = V^2 / R)
- electricPotencyRV<R>(VALUE OF VOLTAGE SQUARED,VALUE OF POWER) --> Calculates resistance (R = V^2 / P)
- electricPotencyRI<P>(VALUE OF CURRENT SQUARED,VALUE OF RESISTANCE) --> Calculates power (P = i^2 / R)
- electricPotencyRI<R>(VALUE OF POWER,VALUE OF CURRENT SQUARED) --> Calculates resistance (R = P / i^2)
- electricPotencyEt<P>(VALUE OF ENERGY,VALUE OF TIME) --> Calculates power (P = E / t)
- electricPotencyEt<E>(VALUE OF POWER,VALUE OF TIME) --> Calculates energy (E = P * t)
- electricPotencyEt<t>(VALUE OF ENERGY,VALUE OF POWER) --> Calculates time (t = E / P)
- forceEquation<force>(VALUE OF ACCELERATION,VALUE OF MASS) --> Calculates force (F = a * m)
- forceEquation<acceleration>(VALUE OF FORCE,VALUE OF MASS) --> Calculates acceleration (a = F / m)
- forceEquation<mass>(VALUE OF FORCE,VALUE OF ACCELERATION) --> Calculates mass (m = F / a)
- positionEquation<position>(VALUE OF INITIAL POSITION,VALUE OF VELOCITY*TIME) --> Calculates horizontal position (Ph = Pi + V*t)
- positionEquation<time>(VALUE OF HORIZONTAL POSITION,VALUE OF INITIAL POSITION,VALUE OF VELOCITY) --> Calculates time (t = Ph-Pi / V)
- positionEquation<velocity>(VALUE OF HORIZONTAL POSITION,VALUE OF INITIAL POSITION,VALUE OF TIME) --> Calculates velocity (V = Ph-Pi / t)

- freefallEquation<height>(VALUE OF VELOCITY,VALUE OF TIME,VALUE OF TIME SQUARED) --> Calculates height (h = V*t - 9.8*(t^2) / 2)
- freefallEquation<velocity>(VALUE OF HEIGHT,VALUE OF TIME SQUARED,VALUE OF TIME) --> Calculates height (V = h + 9.8*(t^2) / 2) / t)
- workEquation<W>(VALUE OF FORCE,VALUE OF DISPLACEMENT) --> Calculates work (W = F * x)
- workEquation<F>(VALUE OF WORK,VALUE OF DISPLACEMENT) --> Calculates force (F = W / x)
- workEquation<displacement>(VALUE OF WORK,VALUE OF FORCE) --> Calculates displacement (x = W / F)
- kineticEquation<E>(VALUE OF MASS, VALUE OF VELOCITY SQUARED) --> Calculates the kinetic energy (Ec = (m * V^2) / 2)
- kineticEquation<mass>(VALUE OF KINETIC ENERGY,VALUE OF VELOCITY SQUARED) --> Calculates the mass (m = 2 * (Ec / V^2))
- potentialEquation<E>(VALUE OF MASS,VALUE OF HEIGHT) --> Calculates the potential energy (Ep = m * h * 9.8)
- potentialEquation<mass>(VALUE OF POTENTIAL ENERGY,VALUE OF HEIGHT) --> Calculates the mass (m = Ep / (9.8 * h))
- potentialEquation<height>(VALUE OF POTENTIAL ENERGY,VALUE OF MASS) --> Calculates the height (h = Ep / (9.8 * m))

* The ‘help’ command in the program will show all of these descriptions.

## “ConvieLte” Development

### Translator architecture and design

This project was developed using Python 3.6.0. Using Lex from PLY, the input from the user is analyzed so the correct syntax and the correct tokens are used during the execution of the program. Then Yacc from PLY was used so the program could compile the input given and perform the proper actions according to the given input. Finally the desired calculations and instructions are executed and the result of the entered function or operation is shown.

### Software environment and test methodology

The software development environment used for this project was Pycharm. This is a compiler which can be used to work with Python files. For the testing methodology during the development, the lexer was the first thing to test by verifying each token represented what it was meant to represent and by giving the program test inputs with the lexer to make sure the tokens worked properly. After testing the lexer, the parser was tested by requesting an input and verifying if said input could be parsed. These are some examples of the testing methodology:

Lexer test:

```
lexer= lex.lex()

#test case del lexer:

lexer.input("1.888M")

while True:
    tok = lexer.token()
    if not tok:
        break
    print(tok)
```

Parser test:

```
while True:
    try:
        s = input('>> ')
    except EOFError:
        break
    parser.parse(s)
```

## Conclusion

During the development of this project, we had the opportunity to work with Pyhton to create a basic programming language. This allowed us to have better understanding of the inner processing of a programming language and the many things it needs to execute each instruction. This was also an opportunity to learn about lexers and parsers using PLY. We were able to implement the grammar of the language and how the parser should analyze the given input and how it should execute the required instructions. With this project we were able to obtain useful skills which may be useful in the industry and a more in depth understanding of what a programming language really is.