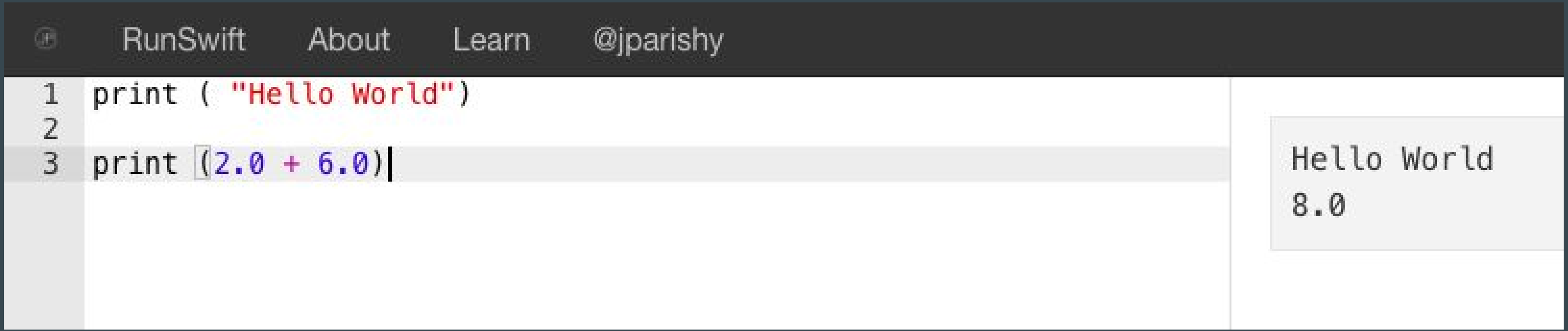
# Programming Languages Final Project

Nich Osborn, Francisco Robleto, Luis Tierrablanca

# Swift Language Feature - Print()

- Print() will output what is inside the parenthesis depending on the input
  - If input is a string surrounded by double quotes, it will print what's inside the quotes
  - If input is numbers and math symbols, it will print the result of the expression

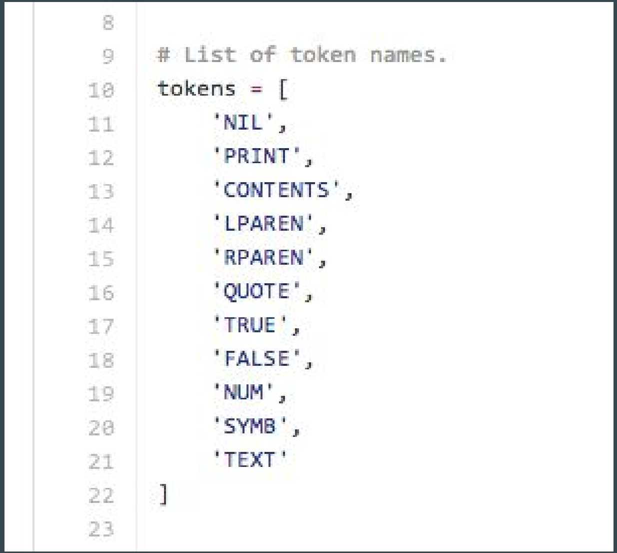


```swift
print ( "Hello World")

print (2.0 + 6.0)
```

```
Hello World
8.0
```

## Overall Process - Setting up

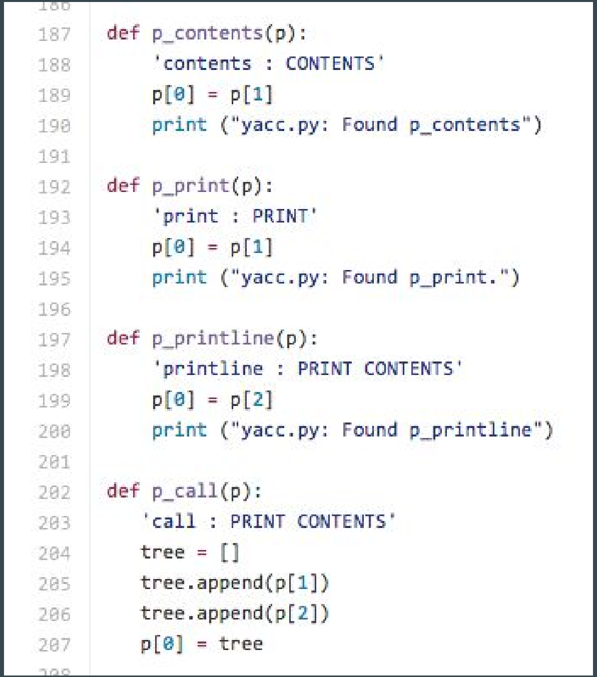
- We are reimplementing Swift’s print() in PLY for strings and mathematical expressions
- To do this, we need to:
    - Create new tokens within Lex.py:



```python

# List of token names.
tokens = [
    'NIL',
    'PRINT',
    'CONTENTS',
    'LPAREN',
    'RPAREN',
    'QUOTE',
    'TRUE',
    'FALSE',
    'NUM',
    'SYMB',
    'TEXT'
]

```

## Overall Process - Setting Up

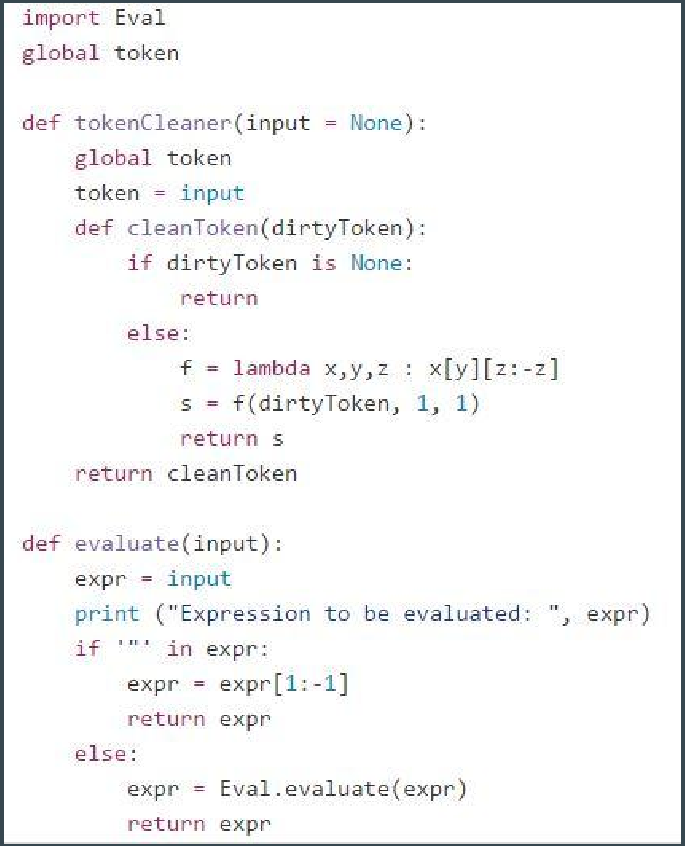
- Create new Grammar Rules within Yacc.py:



```
def p_contents(p):
    'contents : CONTENTS'
    p[0] = p[1]
    print ("yacc.py: Found p_contents")

def p_print(p):
    'print : PRINT'
    p[0] = p[1]
    print ("yacc.py: Found p_print.")

def p_printline(p):
    'printline : PRINT CONTENTS'
    p[0] = p[2]
    print ("yacc.py: Found p_printline")

def p_call(p):
   'call : PRINT CONTENTS'
   tree = []
   tree.append(p[1])
   tree.append(p[2])
   p[0] = tree
```

## Overall Process - Handling the input

- After the command is input and matches up with its grammar rule:
  - It's Abstract Syntax Tree (AST) is sent to a token cleaner function:
    - This will isolate the input without it's parenthesis for further inspection
    - If the remaining input has quotes surrounding it, we output directly to the console and are done
    - Otherwise...



```python
import Eval
global token

def tokenCleaner(input = None):
    global token
    token = input
    def cleanToken(dirtyToken):
        if dirtyToken is None:
            return
        else:
            f = lambda x,y,z : x[y][z:-z]
            s = f(dirtyToken, 1, 1)
            return s
    return cleanToken

def evaluate(input):
    expr = input
    print ("Expression to be evaluated: ", expr)
    if '"' in expr:
        expr = expr[1:-1]
        return expr
    else:
        expr = Eval.evaluate(expr)
        return expr
```

## Overall Process - Evaluating expressions

- Upon finding a mathematical expression within the parenthesis, the program will call Eval.java
    - This allows us to evaluate the 4 common arithmetic expressions (+,-,*,/)
- This Java file will return the result as a double for us to output to the screen

## Bugs We Ran Into

1. Not having PLY folder in directory
2. Not setting jython as SDK from start
3. Grammar Rule not being recognized