Steven Bernas
Prof. Dabish
CS 420
Assignment 8 - Final Report

- **Programming Language Name**
    - Codeopoly
- **Programming Language Purpose & Philosophy**
    - **Purpose**
        - Created Codeopoly to make coding easy for beginners, especially kids and has board game twist using phrases and play action from the classic board game Monopoly
        - Using familiar elements of Monopoly make coding fun, interactive and prevents and solves challenges like frustrations and boredom while programming
    - **Philosophy**
        - Codeopoly programming is simple is a pimple using “CAPS LOCK”, no indents, code blocks, braces and spaces do not matter that could cause syntax errors, which will cause frustration to the brand new coder
        - Overall, making coding fun and engaging using the Monopoly elements
- **Programming Language Style**
    - **Design:**
        - Syntax:
        - Variables
            - To declare and initialize Thing and Name using keywords MY and IS
                - MY [THING] IS [NAME]
                - MY GAME PIECE IS DOG
                - MY [THING] IS [VALUE]
                - MY MONEY IS 100 DOLLARS
            - To add $200 to MONEY keyword is
                - COLLECT PASS GO
        - Control Structures
            - Conditionals
                - To declare if statement keyword is IF compares current MONEY to MONEY to player with highest MONEY then prints you WIN OR LOSE
                - Comparison keyword operators use BIGGER THAN or SMALLER THAN
            - Print output
                - SHOW keyword

- **Semantics:**
    - Data types
    - String
        - THING
            - GAME PIECE
            - MONEY
        - NAME
            - CAR
            - DOG
            - THIMBLE
    - Integer
        - VALUE
            - 0
            - 1125
            - 3502
- **Familiar Elements**
    - Implemented “MY” [GAME PIECE] is “CAR” since kids would usually say that is “My Dog” or “My Car”
    - Implemented “MY” [MONEY] is [1500], since kids would say I’m winning how much do you have
    - Implemented “COLLECT PASS GO” MY” [MONEY] again kids would give me my money

- **Intuitive Design**
    - Designed for kids to prevent boredom, frustration and giving up on programming in addition to an interactive learning experience

- **Go over your Language philosophy and why you created it**
    - Accessibility
        - Even though Codeopoly is designed for kids, the design is accessible by all ages who are familiar with Monopoly phrases, rules, etc. If deployment goes well version 2 will focus on intermediate to advance coders
    - Simplicity
        - Simple syntax using CAPS LOCK without code blocks, braces or indents eliminates room for error with traditional code like Python, C++ and Java
    - Engagement
        - Codeopoly creates a fun and learning environment for new learners to enjoy while learning programming

- **Explain your Three (3) programs and explain how they work**
    - GAME PIECE
        - interpreter()
            - State dict() is looking for keywords “MY”, “IS”, “SHOW”, “MOVE”, “DOLLARS”, “GAME PIECE” and “SPACES”
            - Finds match from grammar.tx in state dict() to perform operations

game_piece.tx file

```
import textx

grammar = '''
Program:
  commands*=Command
;

Command:
  PrintCommand | AssignCommand | MoveCommand | ShowSpacesCommand | ShowMoveCommand
;

PrintCommand:
  'SHOW' 'GAME PIECE'
;

AssignCommand:
  'MY' ('MONEY' 'IS' amount=INT 'DOLLARS' | 'GAME PIECE' 'IS' identifier=ID)
;

MoveCommand:
  'MOVE' 'GAME PIECE' number=INT 'SPACES'
;

ShowSpacesCommand:
  'SHOW'
;

ShowMoveCommand:
  'GAME PIECE' number=INT 'SPACES'
;

Comment:
  /\/\/.*$/
;
'''
```

game_piece.cdpy -> interpreter.py

```
def interpreter(program):
    try:
        varmap = {}  # Initialize variable map
        codeopoly_mm = textx.metamodel_from_str(grammar)
        codeopoly_model = codeopoly_mm.model_from_str(program)

        for c in codeopoly_model.commands:
            class_name = c.__class__.__name__
            if class_name == "AssignCommand":
                varmap[c.variable] = c.value
                print("ASSIGNED {} DOLLARS TO {} VARIABLE".format(c.value, c.variable))
            elif class_name == "PrintCommand":
                if c.variable in varmap:
                    print("VALUE OF {} VARIABLE IS {} DOLLARS".format(c.variable, varmap[c.variable]))
                else:
                    print("'{}' PLEASE ASSIGN VALUE USING KEYWORD MY".format(c.variable))
            elif class_name == "AddValueCommand":
                if c.variable in varmap:
                    varmap[c.variable] += c.value
                    print("{} DOLLARS ADDED TO {} VARIABLE \nNEW MONEY VARIABLE VALUE IS {} DOLLARS".format(c.value, c.variable, varmap[c.variable]))
                else:
                    print("'{}' VALUE NOT ASSIGNED".format(c.variable))
    except:
                print("IF YOU SEE THIS MESSAGE PLEASE CLICK HELP AND SAMPLE PROGRAMS BUTTON")
```

- **COLLECT PASS GO**
    - interpreter3()
        - State dict() is looking for keywords “COLLECT PASS GO”, “IS” “SHOW”and “DOLLARS”
        - Finds match from grammar.tx in state dict() to perform operations

collect_pass_go.tx file

```
import textx

grammar = '''
Program:
  commands*=Command
;

Command:
  PrintCommand | AssignCommand | AddValueCommand
;

PrintCommand:
  'SHOW' variable=ID
;

AssignCommand:
  'MY' variable=ID 'IS' value=INT
;

AddValueCommand:
  'COLLECT PASS GO' value=INT 'DOLLARS TO' variable=ID
;

Comment:
  /\/\/.*$/
;
'''
```

collect_pass_go.cdpy -> interpreter2.py

```
def interpreter(program):
    try:
        varmap = {}  # Initialize variable map
        codeopoly_mm = textx.metamodel_from_str(grammar)
        codeopoly_model = codeopoly_mm.model_from_str(program)

        for c in codeopoly_model.commands:
            class_name = c.__class__.__name__
            if class_name == "AssignCommand":
                varmap[c.variable] = c.value
                print("ASSIGNED {} DOLLARS TO {} VARIABLE".format(c.value, c.variable))
            elif class_name == "PrintCommand":
                if c.variable in varmap:
                    print("VALUE OF {} VARIABLE IS {} DOLLARS".format(c.variable, varmap[c.variable]))
                else:
                    print("'{}' PLEASE ASSIGN VALUE USING KEYWORD MY".format(c.variable))
            elif class_name == "AddValueCommand":
                if c.variable in varmap:
                    varmap[c.variable] += c.value
                    print("{} DOLLARS ADDED TO {} VARIABLE \nNEW MONEY VARIABLE VALUE IS {} DOLLARS".format(c.value, c.variable, varmap[c.variable]))
                else:
                    print("'{}' VALUE NOT ASSIGNED".format(c.variable))
    except:
                print("IF YOU SEE THIS MESSAGE PLEASE CLICK HELP AND SAMPLE PROGRAMS BUTTON")
```

- WIN OR LOSE
    - (interpreter3())
        - State dict() is looking for keywords “IF”, “BIGGER THAN’, ‘LESS THAN”, “DOLLARS’, “MY”, “IS’, “SHOW”, “MONEY’ and “CONGRATULATIONS! YOU WIN!”,

- Finds match from grammar.tx in state dict() to perform operations

win_or_lose.tx

```
import textx

grammar = '''
Program:
  commands*=Command
;

Command:
  PrintCommand | AssignCommand | ShowCongratulationsCommand
;

PrintCommand:
  'SHOW' message=STRING
;

AssignCommand:
  'MY' 'MONEY' 'IS' amount=INT 'DOLLARS'
;

ShowCongratulationsCommand:
  'SHOW' 'CONGRATULATIONS! YOU WIN!' 'IF' 'MY' 'MONEY' 'IS' 'BIGGER THAN' threshold=INT 'DOLLARS'
;

Comment:
  /\/\/.*$/
;
'''
```

win_or_lose.cdpy -> interpreter3.py

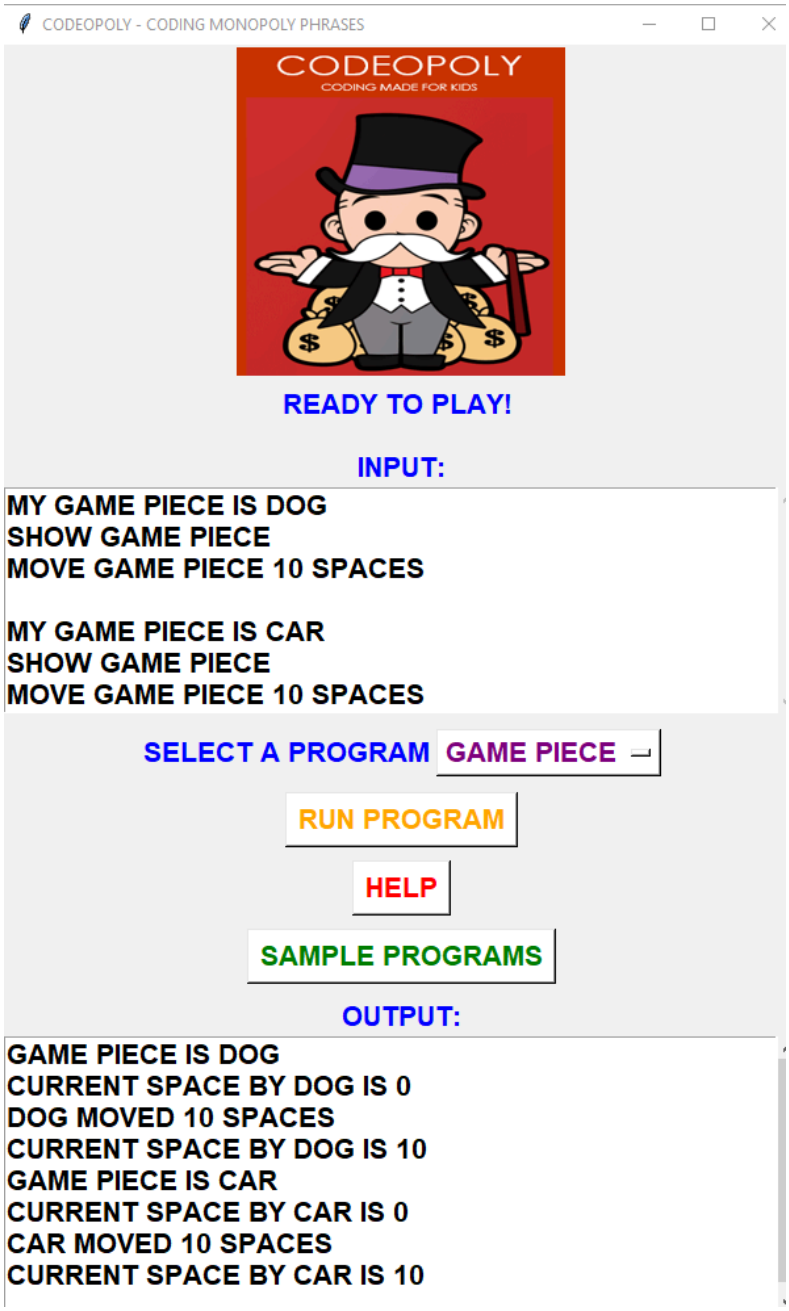
```
def interpreter(program):
    try:
        varmap = {'MONEY': 0}
        codeopoly_mm = textx.metamodel_from_str(grammar)
        codeopoly_model = codeopoly_mm.model_from_str(program)

        for c in codeopoly_model.commands:
            class_name = c.__class__.__name__
            if class_name == "AssignCommand":
                varmap['money'] = c.amount
                print(f"Assigned {c.amount} dollars to MONEY")
            elif class_name == "PrintCommand":
                print(c.message)
            elif class_name == "ShowCongratulationsCommand":
                if 'money' in varmap:
                    if varmap['money'] > c.threshold:
                        print("CONGRATULATIONS! YOU WIN!")
                    else:
                        print("YOU HAVEN'T WON YET")
                else:
                    print("MONEY NOT ASSIGNED YET")
    except:
                print("IF YOU SEE THIS MESSAGE PLEASE CLICK HELP AND SAMPLE PROGRAMS BUTTON")
```

- **Demonstrate your interpreter running**
    - Using tkinter GUI located in main.py
    - Input window
    - Output window
    - Drop down menu
    - Help menu
    - Sample program menu

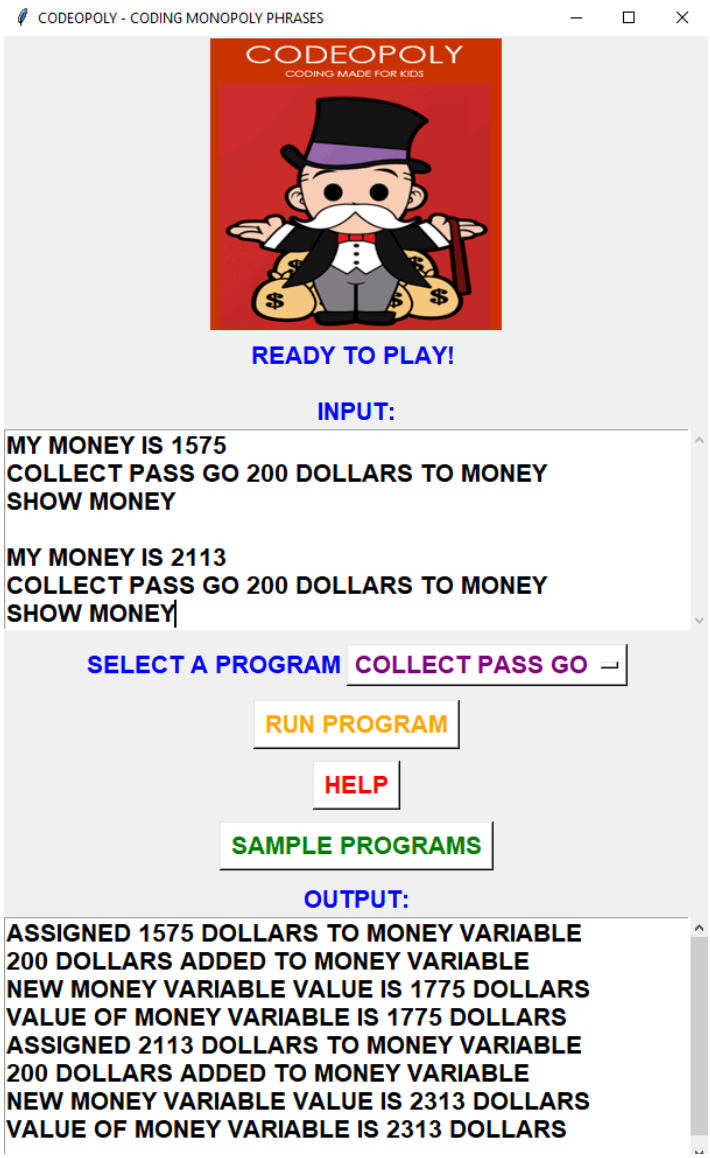
```python
from tkinter import scrolledtext
from tkinter import *
from PIL import ImageTk, Image
from interpreter import interpreter as GAME_PIECE
from interpreter3 import interpreter as interpreter3
from interpreter2 import interpreter as interpreter2

def interpret_program():
    program = code_editor.get("1.0", tk.END)
    interpreter = interpreter_selection.get()

    sys.stdout = output_buffer = StringIO()

    if interpreter == "GAME PIECE":
        GAME_PIECE(program)
    elif interpreter == "COLLECT PASS GO":
        interpreter3(program)
    elif interpreter == "WIN OR LOSE":
        interpreter2(program)

    output_text.delete("1.0", tk.END)
    output_text.insert(tk.END, output_buffer.getvalue().upper())
    sys.stdout = sys.__stdout__

def open_help_file():
    with open("help.txt", "r") as file:
        help_content = file.read()
        output_text.delete("1.0", tk.END)
        output_text.insert(tk.END, help_content.upper())

def open_sample_programs_file():

    file = open("sample_programs.txt", "r")


    sample_programs_content = file.read()

    output_text.delete("1.0", tk.END)

    output_text.insert(tk.END, sample_programs_content.upper())



root = tk.Tk()
root.title("CODEOPOLY - CODING MONOPOLY PHRASES")

font_style = ("Mulish", 16, "bold")
```

```python
font_colors = {
    "label": "blue",
    "button1": "orange",
    "button2": "red",
    "button3": "green",
    "dropdown": "purple"
}

opened_image = Image.open("codeopoly.png")
resized_image = opened_image.resize((250, 250))
logo_image = ImageTk.PhotoImage(resized_image)
logo_label = tk.Label(root, image=logo_image)
logo_label.pack()

code_frame = tk.Frame(root)
code_frame.pack(pady=5)

code_label = tk.Label(code_frame, text="READY TO PLAY! \n\n INPUT: ".upper(), font=font_style, fg=font_colors["label"])
code_label.pack()

code_editor = scrolledtext.ScrolledText(code_frame, width=75, height=7, font=font_style)
code_editor.pack()

interpreter_frame = tk.Frame(root)
interpreter_frame.pack(pady=5)

interpreter_label = tk.Label(interpreter_frame, text="SELECT A PROGRAM".upper(), font=font_style, fg=font_colors["label"])
interpreter_label.pack(side=tk.LEFT)

interpreters = ["GAME PIECE", "COLLECT PASS GO", "WIN OR LOSE"]
interpreter_selection = tk.StringVar(root)
interpreter_selection.set(interpreters[0])
interpreter_dropdown = tk.OptionMenu(interpreter_frame, interpreter_selection, *interpreters)
interpreter_dropdown.config(font=font_style, fg=font_colors["dropdown"], bg="white")
interpreter_dropdown["menu"].config(font=font_style)
interpreter_dropdown.pack(side=tk.LEFT)

interpret_button = tk.Button(
    root,
    text="RUN PROGRAM".upper(),
    command=interpret_program,
    font=font_style,
    fg=font_colors["button1"],
    bg="white"
)

interpret_button.pack(pady=5)


sample_programs_button = tk.Button(
    root,
    text="SAMPLE PROGRAMS".upper(),
    command=open_sample_programs_file,
    font=font_style,
    fg=font_colors["button3"],
    bg="white"
)

sample_programs_button.pack(pady=5)

output_frame = tk.Frame(root)
output_frame.pack(pady=5)

output_label = tk.Label(output_frame, text="Output:".upper(), font=font_style, fg=font_colors["label"])
output_label.pack()

output_text = scrolledtext.ScrolledText(output_frame, width=75, height=10, font=font_style)
output_text.pack()

root.mainloop()
```

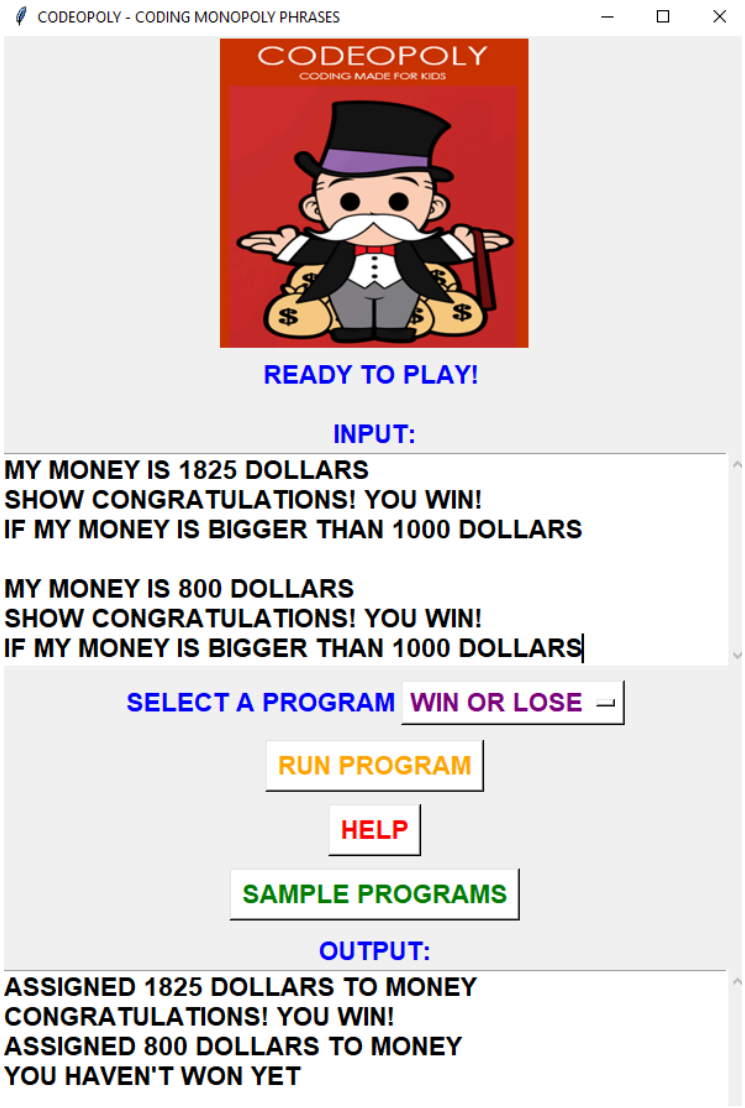
- GAME PIECE
    - interpreter()
        - Select a program from drop down menu
        - Choose GAME PIECE
        - Enter code in input window
        - Click run program
        - Check output window



- COLLECT PASS GO
    - (interpreter2()
        - Select a program from drop down menu
        - Choose COLLECT PASS GO
        - Enter code in input window
        - Click run program
        - Check output window



- WIN OR LOSE
    - (interpreter3())
        - Select a program from drop down menu
        - Choose WIN OR LOSE
        - Enter code in input window
        - Click run program
        - Check output window



Honor Code - Please do not forget the honor code on your Module:

"*I have neither given nor received unauthorized aid in completing this work, nor have I presented someone else's work as my own.*"