Crash Course in Python

Shmuel Jacobs

May 24, 2018

Design Process of Rock, Paper, Scissors game.

Topics:

- Introduction to Functions
- Abstraction (implicitly covered)
- Typical Definition of Functions
- Keyword and Default arguments

Outline the game

Let's start with a very rough idea of all the steps needed in programming this game. I'm writing comments in a .py file, so technically, this is python. That way, I'll be able to work in the same file and it will run.

```
# Print an introduction. Mention typing q to quit.

# Get some user input

# In each round (until the user types q to quit):

    # Generate the computer's move. Make use of some kind of random behavior.


    # Use some logic to determine who wins.


    # Let the player know who wins.


    # Any long term score keeping needs to happen here if it's happening at all.


    # Get next user input.


# After user quits, print some goodbye message.
```

If you like, you can write between my comments to think through how you'll handle stuff. However, before you try to tackle the whole thing, go to the next page, because I want to show you how to procrastinate to improve your productivity.

Let’s put off some tasks. Look at the tasks inside the loop. I know I’ll figure out how to handle them at some point. Let’s just name the process of handling each task, so that I can write it somewhere else.

```
# Print an introduction. Mention typing q to quit.


# Get some user input


# In each round (until the user types q to quit):


      # Generate the computer's move. Make use of some kind of random behavior.
      computerIn = computerPlay( [] )


      # Use some logic to determine who wins.
      winner = chooseWinner(userIn, computerIn)


      # Let the player know who wins.



      # Any long term score keeping needs to happen here if it's happening at all.



      # Get next user input.



# After user quits, print some goodbye message.
```

Now, let’s write a dummy to do something where we plugged in those functions. Put the following **Function Declarations** at the top of your code: Use the keyword `def`, and then the function name followed by inputs, called **arguments**, or **parameters**.

```
def computerPlay( handsList ):
      return ‘r’
def chooseWinner(userScore, computerScore):
      return 1
```

Python knows that when it hits those function names later, it should jump back up to run the code in the definions, until it hits a ‘return’ statement. Then, it takes the return value plugs it into the code where you called the function. Effectively, code that says “computerPlay” will be wiped out and replaced with “ ‘r’ ”.

```
def computerPlay( handsList ):
      return 'r'
def chooseWinner(userScore, computerScore):
      return 1
```

Let’s work on running the game until the user quits. Use a ‘while’ loop, just like parrot.py in the last packet.

```
# Print an introduction. Mention typing q to quit.

# Get some user input
userIn = input()
# In each round (until the user types q to quit):
while(userIn != 'q'):
      # Generate the computer's move. Make use of some kind of random behavior.
      computerIn = computerPlay( [] )

      # Use some logic to determine who wins.
      winner = chooseWinner(userIn, computerIn)

      print('You chose: ', userIn, '\nComputer chose: ', computerIn)
      # Let the player know who wins.


      # Any long term score keeping needs to happen here if it's happening at all.


      # Get next user input.
      userIn = input()

# After user quits, print some goodbye message.
```

This should actually run. It doesn’t do much. It just takes input and chooses a counter, until you quit.

Let’s write some more reasonable game logic into the functions above.

Expanded functions. I wrote all of this, so feel free to ask me about it. I've commented heavily, so I'll remember what everything does.

```
# Include some code that knows how to choose randomly.
import random
# define list of options
handBeats = {'r':'s', 'p':'r', 's':'p'}
hands = list(handBeats.keys())
intro = 'Welcome to Rock, Paper, Scissors. Type q at any time to quit.'
roundInstr = 'Type r, p or s to play a round of Rock, Paper, Scissors.\nrps> '

# Define procedure for letting computer choose one play from a list of options.
def computerPlay(moveChoices):
      # moveChoices is the list of choices to for computer to choose from.

      computerHand = random.choice(moveChoices)

      return computerHand
      print('This line will never print. After "return", we leave the function')

'''Define procedure for choosing winner based on two inputs, assuming first
input is player's choice and second input is computer's. Return 1 for player wins,
0 for tie, -1 for computer wins, -2 for invalid choice.'''
def chooseWinner(playerChoice, computerChoice):
      if(playerChoice == computerChoice):
            return 0
      # If we reach next line, choices don't match.
```

Why don't I need 'else' in this section?

```
      if( handBeats[playerChoice] == computerChoice ):
            return 1
      if( playerChoice not in hands ):
            # invalid choice -- return error code: -2
            return -2

      # The only way to reach this is if player doesn't win and doesn't tie.
```

I've left a bug in here. What if user types 'a'? Can you fix it?

Let's put in some long term score keeping. I like the way those functions work, so I'm just looking at the main gameplay after the function definitions.

```
showPrompt(intro)
userIn = input(roundInstr)
playerWins = 0
computerWins = 0

# In each round (until the user types q to quit):
while(userIn != 'q'):
      # Generate the computer's move. Make use of some kind of random behavior.
      computerIn = computerPlay( hands )

      # Use some logic to determine who wins.
      winner = chooseWinner(userIn, computerIn)

      print('You chose: ', userIn, '\nComputer chose: ', computerIn)
      # Let the player know who wins, and keep score.
      if(winner == 1):
            print('You win')
            playerWins += 1
      elif(winner == -1):
            print('You lose')
            computerWins += 1
      elif(winner == 0):
            print('You tie. Let\'s not count that one')
      else:
            print('Maybe you misunderstood the choices')

      # Get next user input.
      userIn = input()
```

Of course, I wrote a few other functions to simplify stuff, and to show how to get data into and out of functions. Check out this one:

```
def showStandings(playerScore, ComputerScore):
      print('You\'re ', playerScore, ' for ', playerScore + ComputerScore, '.')
```

My full code will be distributed later. Try putting some pieces together.