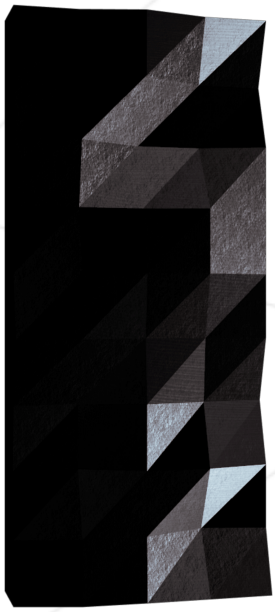

Parseltongue Piscine - Part03

Transforming Strings, While loops, Dictionaries

Kai kai@42.us.org
Ruchen ruchen@42.us.org

Summary:

Contents

`Eat, Sleep, Code, Repeat.`

Chapter I

Don't Panic!

Remember the three commandments of 42:

1. Ask the person on the left of you
2. Ask the person on the right of you
3. Read the Manual (i.e. the documentation)!

Chapter II

Transforming Sequences

II.1 FOPP Chapter 9

Go to Runestone: Fundamentals of Python and complete section 9 before the next exercise.

II.2 Exercise 0: PaReNtHeSiS

	PaReNtHeSiS
Topics to study : `Modulo, String manipulation, accumulator pattern`	
Files to turn in : `00_parentheses.py`	
Notes : `String`	

Write a progam which performs three tasks in a row on the phrase given as a command line argument.

First, it prints out the phrase with eVeRy OtHeR lEtTeR cApItAlIzEd.

Then, print out the same string with every capitalized vowel replaced by an asterisk.

Next, run a check_parentheses function which determines whether every open paren "(" in the phrase is balanced with a close paren ")". Print out "Balanced? True" or "Balanced? False" accordingly.

Don't worry about the order of the parentheses, just the number of them. (It's an advanced problem if you check for correct order of parentheses too. Go ahead and try if you want, for a bonus!)

```
?> python3 00_parentheses.py "(whats going on()?)"
(WhAtS gOiNg On()?)
(Wh*tS g*iNg *n()?)
Balanced? True
```

```
?> python3 00_parentheses.py "()()()()(((a)b)b)b"
()()()()(((A)b)B)b
()()()()(((*)b)B)b
Balanced? True
```

```
?> python3 00_parentheses.py "((this doesn't match)"
((ThIs DoEsN't MaTcH)
((Th*s Do*sN't MaTcH)
Balanced? False
```

Chapter III

Modules and the While loop

III.1 FOPP Chapters 4 & 14

Go to Runestone: Fundamentals of Python and complete sections 4 and 14 before the next exercise.

III.2 Exercise 1: Guessing Game

	Guessing Game
Topics to study : `Random library, while loop`	
Files to turn in : `01_guess.py`	
Notes : `Control Flow, While Loop, Break and Continue`	

Create a program which contains an array of five-letter words.
When the program begins, if should pick a random word from the array (make sure to "import random" to help with this!). Then it tells the user what letter the word starts with and invites the user to guess.

Use an infinite while loop to receive guesses from the user. Make it so that it exits if the user correctly guesses the word, or guesses wrong 10 times.

The program says different things depending on the input:

- If the user does not type a word but presses Enter, the program informs them, "You wasted a guess =P"
- If the user types a word that is longer or shorter than five letters long, the program helpfully prints out "0, 1, 2, 3, 4 that's how we count to five!".
- If the user types a word that is five letters long but does not start with the correct letter, the program helpfully prints out the full alphabet, and does not count it as a wrong guess.

- If the user types a word that is five letters long and starts with the correct letter but is not the secret word, the program informs them how many guesses they have left.

- If the user guesses the word correctly, the program prints out, "Good Job! You are one with the Source."

- If the user spends all 10 guesses and does not get the question right, exit the program.

```
?> python3 01_guess.py
The secret word begins with a D.
GUESS: delighted
0, 1, 2, 3, 4 that's how we count to five!
GUESS: rigor
ABCDEFGHIJKLMNOPQRSTUVWXYZ
GUESS:
You wasted a guess!
GUESS: dolma
You have 7 guesses left!
GUESS: dough
Good Job! You are one with the Source.
```

Chapter IV

Dictionaries

IV.1 FOPP Chapter 11

Go to Runestone: Fundamentals of Python and complete section 11 before the next exercise.

IV.2 Exercise 2: State Capitols

	State Capitols
Topics to study : `Infinite while loop, Exit`	
Files to turn in : `02_capitols.py`	
Notes : `Use the capitols.txt file provided on the project page. You do not need to turn that file in, but you can include it in your repository.` `Input/Output`, `Text Files in Python`, Python `dictionaries`	

- Create a script `02_capitols.py` which reads in the provided comma-delimited file of US States and capitals and stores this information in a hashtable.
- Next, on an infinite loop, print "Ready: " and wait for the user to enter the name of a state or capital. For each query print out the associated capital or state and go back to Ready state.
- The program exits when the user types "Done". If the input is invalid, answer "nil".

```
?> python3 02_capitols.py capitals.txt
Ready: Arizona
Phoenix
Ready: Montana
Helena
Ready: MacaroniAndCheese
nil
Ready: Pierre
South Dakota
Ready: Done
?>
```