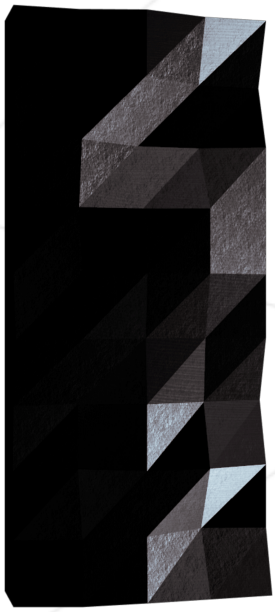

# Parseltongue Piscine - Day03

## keep practicing and build more complex projects

Kai kai@42.us.org

*Summary:*

# Contents


`Eat, Sleep, Code, Repeat.`

# Chapter I

# Don’t Panic!

Confused about how to begin? Not sure what the PDF means?

Don’t worry, we are not perfect :)

And the PDFs are challenges - not walkthroughs. Team up with your group, your partner, your mentor and your other peers to decipher what to do.

You are the master of your own destiny! Go forward and code the world !

# Chapter II

# Format your Code

Each 42 challenge you turn in must adhere to the following format:

```python
#!/usr/bin/env python3

# Write your name at the top, and any helpful comments you have for people
# running your program.
# By <userid>

import sys

def function_a:
    # code

def function_b:
    # code

def main(argv):
    # main method
    function_a
    function_b

main(sys.argv)
```

or:

```ruby
#!/usr/bin/env ruby

# Write your name at the top, and any helpful comments you have for people
# running your program.
# By <userid>

def function_a
 # code
end

def function_b
 # code
end

def main(ARGV)
 # main method
 function_a
 function_b
end

main(ARGV)
```

- Always begin with the "#!/usr/bin/env ruby" statement. This tells your terminal to run the program using Ruby. In python, the first line is "#!/usr/bin/env python3".

- Always add a comment stating what this program is for, some hints to help others use or understand it, and your name or intra ID.

- Do not write any code outside of functions except for one line, at the end of your program, which calls the main() function.

# Chapter III

# Exercise 0: Rainbow

|  | Rainbow |
|---|---|
| Topics to study : `Functions and parameters, ANSI escape codes` | |
| Files to turn in : `00_rainbow.rb or 00_rainbow.py` | |
| Notes : `Credit to StackOverflow for the demonstration code` | |

Did you know that it is possible to print colorful text in the terminal?

Here is a sample Python program, which if you save and run it, will show you many options for color combinations that you can get without installing any special libraries.

```
def print_format_table():
    """
    prints table of formatted text format options
    """
    for style in range(8):
        for fg in range(30,38):
            s1 = ''
            for bg in range(40,48):
                format = ';'.join([str(style), str(fg), str(bg)])
                s1 += '\x1b[%sm %s \x1b[0m' % (format, format)
            print(s1)
        print('\n')

print_format_table()
```

Run this program, study this code and think about what is "says", how it does what it does.

Write a program called `00_rainbow.py` which includes a function called write_colorful_text. The function will take in one parameter for the text to be written, one parameter for the "style" (in the range of 0 to 8), one parameter for the "background" (in the range of 40 to 48), and one parameter for the "foreground" (in the range from 30 to 38). The first line looks like this:

```
def print_colorful_text(string, style, foreground, background):
```

Call the write_colorful_text function several times, with different parameters. Your function should print out the word "R A I N B O W" with each letter a different color, in rainbow order. You can decide what color combination is best for rainbow order.

### III.0.1 Bonus

Use global variables to label each number with the color or style it represents. You should then be able to call your function like this:

```
print_colorful_text("I am Groot", BOLD, FG_GREEN, BG_DARKBLUE)
```

...or similarly with any other color labels that are useful to you.

This is not the most common color encoding system, by the way. Later on - in Game Design class - we will use RBG format where each color in R, G, and B has a value between 0 and 255.

# Chapter IV

# Exercise 1: PaReNtHeSiS

|  | PaReNtHeSiS |
|---|---|
| Topics to study : | |
| Files to turn in : `01_parentheses.rb or 01_parentheses.py` | |
| Notes : Ruby String, Regular Expressions Python String, Regular Expressions | |

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!)

```
?> python 01_parentheses.py "(whats going on()?)"
(WhAtS gOiNg On()?)
(Wh*tS g*iNg *n()?)
Balanced? True
```

```
?> python 01_parentheses.py "()()()()(((a)b)b)b"
()()()()(((A)b)B)b
()()()()(((*)b)B)b
Balanced? True
```

```
?> python 01_parentheses.py "((this doesn't match)"
((ThIs DoEsN't MaTcH)
((Th*s Do*sN't MaTcH)
Balanced? False
```

`Use gsub or re.sub. Study a little about regex. Look up what string interpolation means & use it all the time.`

# Chapter V

# Exercise 2: Math Class

|  | Math Class |
|---|---|
| Topics to study : | |
| Files to turn in : `02_mathclass.rb or 02_mathclass.py` | |
| Notes : `Ruby` Array, `Python` List, More on Lists | |

Take in a set of numbers as command line arguments. Store them as an array and print out the min, max, mean, median, mode and range of the set.

You may not use "import statistics". You may use "import math".

```
?> python 02_mathclass.py 142 6 13 36 54 4 9 78 78 102
Min: 4
Max: 142
Mean: 52.2
Median: 45
Mode: 78
Range: 138
```

Each mathematical task should take place in its own function. Write a function to find the min, one to find the max, one to find the mean, and so on.

### V.0.1 Bonus

As a bonus, but not the main part, you may turn in a version of this program which demonstrates how much easier this is using the Statistics library. :)

# Chapter VI

# Exercise 3: Guessing Game

|  | Guessing Game |
|---|---|
| Topics to study : | |
| Files to turn in : `03_guess.rb or 03_guess.py` | |
| Notes : `Ruby` Control Expressions `Python` Control Flow | |

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.

```
?> 03_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 6 guesses left!
GUESS: dough
Good Job! You are one with the Source.
```