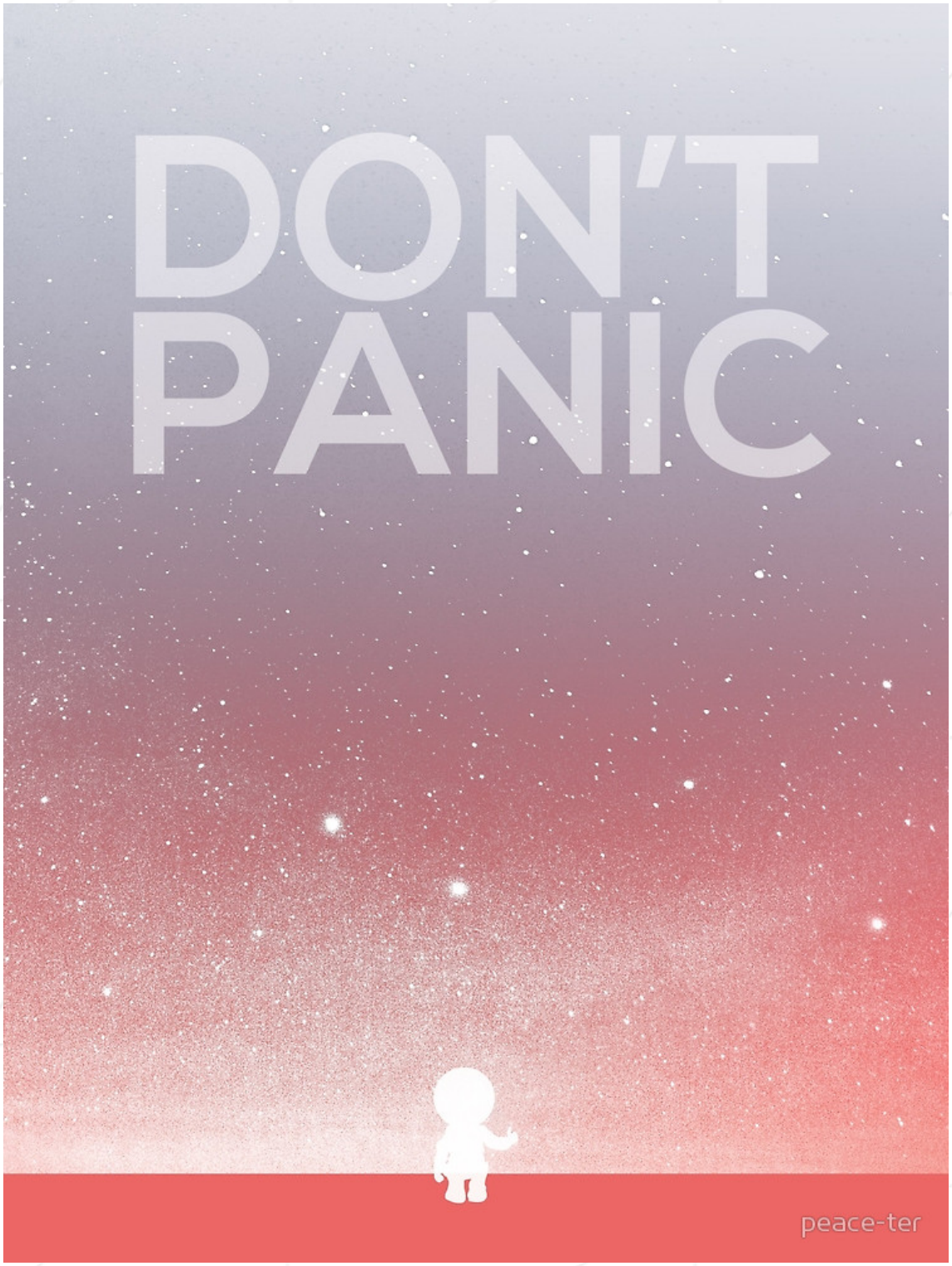
Parseltongue Piscine - Part02

Arrays; Numbers; Booleans

Kai kai@42.us.org

Summary: Learn about conditionals, booleans, lists, indexing, for loops, ARGV, and sorting functions

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

Conditionals

II.1 FOPP Chapter 8

Go to Runestone: Fundamentals of Python and complete section 8 before the next exercise.

II.2 Exercise 0: Daenerys

	Daenerys
Topics to study : `If/Else, Equality Testing`	
Files to turn in : `00_daenerys.py`	
Notes : `Control Flow, Conditionals, Booleans`	

- Create a program called `00_daenerys.py` which asks your name and only greets you if your name is “Daenerys of the House Targaryen, the First of Her Name, The Unburnt, Queen of the Andals, the Rhoynar and the First Men, Queen of Meereen, Khaleesi of the Great Grass Sea, Protector of the Realm, Lady Regnant of the Seven Kingdoms, Breaker of Chains and Mother of Dragons” or "DHTFHNUQARFMQMKGSPRLRSKBCMD" for short.
- Otherwise, if your name is "Dany", the program replies "Dany who?".
- For any other name, the program replies "Move along, now."

```
?> python3 00_daenerys.py
Who goes there?
?> DHTFHNUQARFMQMKGSPRLRSKBCMD
Welcome, Daenerys.
```

```
?> python3 00_daenerys.py
Who goes there?
?> Dany
Dany who?
```



```
?> python3 00_daenerys.py
Who goes there?
?> Jaqen H'gar
Move along, now.
```


Chapter III

Sequences and Iteration

III.1 FOPP Chapter 6 & 7

Go to Runestone: Fundamentals of Python and complete sections 6 and 7 before the next exercise.

III.2 Exercise 1: Brainstorm

	Brainstorm
Topics to study : `ASCII Control Codes`, `Lists`	
Files to turn in : `01_brainstorm.py`	
Notes : `Lists`, `Append List Method`, `Output Formatting`	

Create a program called `01_brainstorm.py` which imitates a version of the game Scattegories. Here are some goals for it:

- Your game should have a built in list or tuple which contains a selection of categories. When the game begins, choose one of the categories from the list randomly. (Research: How to choose a random number between 0 and n?)
- Once the random category is chosen, the program displays it and loops 10 times to collect input from the user.
- Every answer the user gives is added to another list, the answers lists.
- After 10 entries, display each answer that was given.
- The answers should be displayed neatly. Print them in a column that is centered in the middle of the terminal.
- Bonus: Time how long it takes the user to enter 10 items for that category, and display the elapsed time at the end.

- Bonus: Draw a nice box around the answer table output.

III.3 Exercise 2: George Bool

	George Bool
Topics to study : `Booleans, iPython`	
Files to turn in : `02_bool.py`	
Notes : `Constants`, `Types`, `Booleans`, `Python Interpreter`	

Using the three provided lists:

[False,True,True,None,True,None,None,False,False,None,True,False]
["or","or","or","==","!=","==","and","==","!=","and","==","or"]
[False,False,None,None,True,True,False,True,None,False,True,None]

Write a program which tests boolean logic by constructing logical statements from the provided lists and evaluating their results.

You should have the lists shown above, or something similar, as variables in your code. The program will iterate through each index for the length of these lists.

For each index, combine the values from the three list at that index into one syntactically valid string. Example: at index 0, "False" + "or" + "False" should be combined with spaces to the expression "False or False".

Use the Python built-in function "eval()"" to find the result of the Boolean expression, and print it out.

```
?> python3 02_bool.py
False or False => False
True or False => True
True != True => False
...
```

You can invoke Interactive Python in the terminal by typing "Python". In here, anything you type will be evaluated as Python code and it will show you the result - so you can type "True or None" to see what the answer is. Type exit() to exit.

III.4 Exercise 3: Palindrome

	Palindrome
Topics to study : `Index, incrementing, for loops, string manipulation`	
Files to turn in : `03_palindrome.py`	
Notes : `The goal of this project is to learn to work through strings by accessing individual letters through their index.` `For Loops`	

Create a program called `03_palindrome.py` which takes in a sentance, word, or number and tells you if it is a palindrome or not.
Here's a few things to think about:

- The phrase "A man, a plan, a canal: Panama." is a valid palindrome. You don't want your program to ignore because of the spaces and punctuation marks, do you? You will have to find a way to handle spaces, capitalization, and punctuation.
- How can you run tests of equality on specific letters in a string?
- Your program will be nicer quality if it prints out a prompt, like "Enter the text which may be a palindrome:", and then a readable answer, like "So cool! This text IS a palindrome." At the bare minimum, it must print out 'true' or 'false'.

Chapter IV

Sorting

IV.1 FOPP Chapter 16

Go to Runestone: Fundamentals of Python and complete section 16 before the next exercise.

IV.2 Exercise 4: ARR Matey

	ARR, Matey
Topics to study : `ARGV`	
Files to turn in : `04_arrmatey.py`	
Notes : `What is ARGV?`, `Increment and Decrement`	

- Create a script `04_arrmatey.py` which takes a sentence worth of command-line arguments, splits them into an array, and then prints them each out on a different line along with the corresponding index of the array.

- Next, sort the array by word length and reverse it, printing just the words in descending order of length.

```
?> python3 04_arrmatey.py docs.python.org/3/ has official info regarding every module :\)
Argv of 0 is 01_arrmatey.py
Argv of 1 is docs.python.org/3/
Argv of 2 is has
Argv of 3 is official
Argv of 4 is info
Argv of 5 is regarding
Argv of 6 is every
Argv of 7 is module
Argv of 8 is :)
docs.python.org/3/
01_arrmatey.py
regarding
official
module
every
info
has
```

```
:)
?>
```

