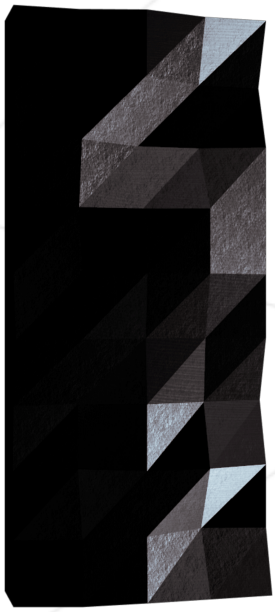


Hack Stack

Find help on stackoverflow!

Pragathi N pragathi.code@gmail.com
Ted Tran 42hourai@gmail.com

Summary: This PDF is an introduction to **`stacks`** *and* **`queues`** *in Python.*

Contents

Chapter I

Foreword

Leptodactylus fallax, commonly (and deceptively) known as the mountain chicken or giant ditch frog, is a species of frogs that is native to the Caribbean islands of Dominica and Montserrat. The population has declined 81% in the last ten years and this species is now critically endangered. In 2004 it was estimated that the population possibly was as low as 8,000 individuals. One of the main threats is human consumption.

Being deliciously chicken-tasting is dangerous.

Chapter II

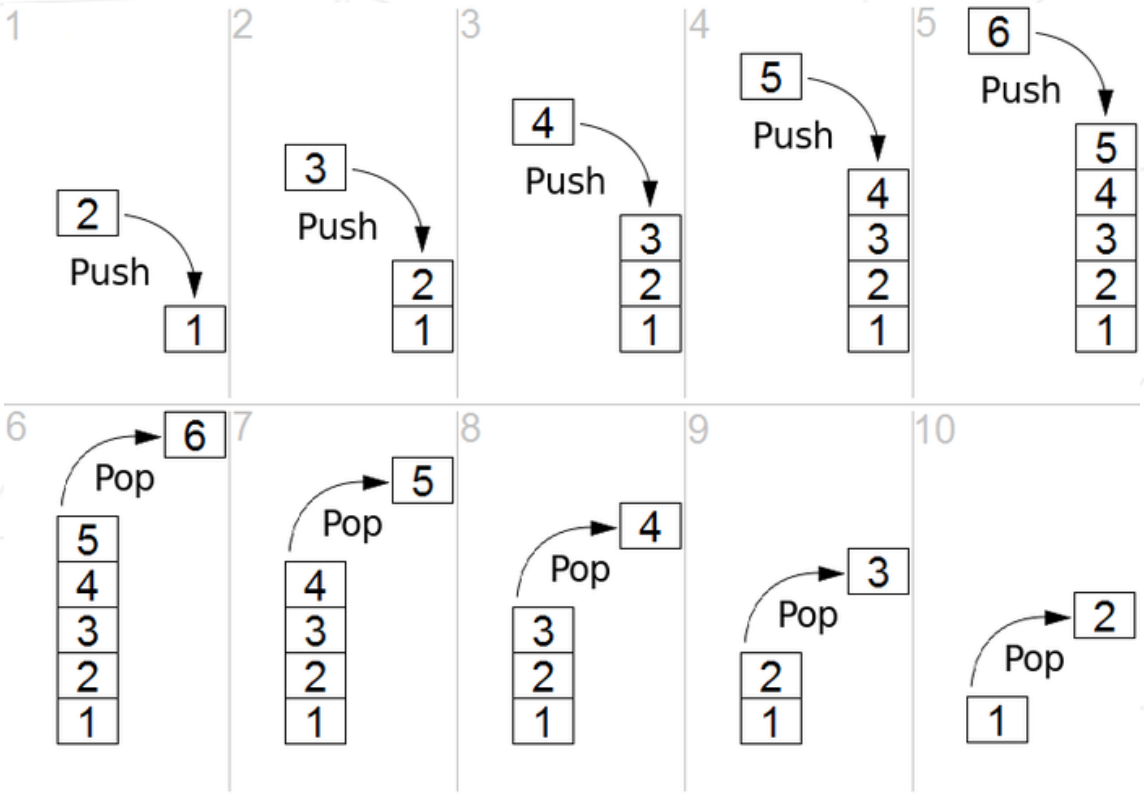
Goals

- What is a Stack?
- What is a Queue?

Chapter III

Introduction

Prepare to work with stacks! Stacks are a LIFO data structure with two main operations– pop and push.
If you don't know what LIFO means, google it! Or bother Pragathi, her number is 1-888-447-5594. If you want someone nicer, call me at 605-475-6966!



Chapter IV

Exercise 00 : Stack Creation

	Exercise 00
Stack Creation	
Turn-in directory : *ex*00/	
Files to turn in : `stack.py`	
Allowed functions : `None`	
Notes : `n/a`	

Make use of the Node class for the elements of the stack

```
class Node:
    def __init__(self, value, next = None):
        self.data = value
        self.next = next
```

Implement a stack class with the following methods:

def ___init___(self): Initialize the stack class with required variables

def isEmpty(self): Checks if the stack is empty or not (return None if empty)

def push(self, data): Adds an element to the stack

def pop(self): removes the top element from the stack

def peek(self): retrieves the value of the top element of the stack

def size(self): returns the size of the stack

def ___str___(self): prints the elements of the stack

```python
class Stack:
    def __init__(self):
        """ Write your code """
        pass

    def push(self, value):
        """ Write your code """
        pass

    def pop(self):
        """ Write your code """
        pass

    def peek(self):
        """ Write your code """
        pass

    def isEmpty(self):
        """ Write your code """
        pass

    def size(self):
        """ Write your code """
        pass

    def __str__(self):
        """ Write your code """
        pass
```

Chapter V

Exercise 01 : Basic Arithmetic

	Exercise 01
Basic Arithmetic	
Turn-in directory : $ex01/$	
Files to turn in : `math.py`	
Allowed functions : `None`	
Notes : `n/a`	

Input a list and create a stack of the elements in the given list and implement the following operations on the stack. Start from the top of the stack when doing operations! Print the values in the same order as functions

- Calculate the total number of elements in the stack.
- Calculate the sum of all the elements in the stack.
- Multiply all the elements in the stack.
- Find the mean of the stack
- Lastly, find the maximum and minimum elements of the stack.

```
?> python math.py
Enter the numbers: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Total count = 10
Sum = 55
Product = 3628800
Mean = 5
Min = 1
Max = 10
```

Chapter VI

Exercise 02 : Queue Creation

	Exercise 02
Queue Creation	
Turn-in directory : *ex*02/	
Files to turn in : `queueClass.py`	
Allowed functions : `None`	
Notes : `n/a`	

Make use of the Node class for the elements of the queue

```
class Node:
    def __init__(self, value, next = None):
        self.data = value
        self.next = next
```

Implement a queue class with the following methods:

def ___init___(self): Initialize the queue class with required variables

def isEmpty(self): Checks if the queue is empty or not (return None if empty)

def enqueue(self, data): Inserts an element to end of the queue

def dequeue(self): Removed an element from the beginning of the queue

def size(self): Returns the length of the queue

def ___str___(self): prints the elements of the queue

Chapter VII

Exercise 03 : Reverse String

	Exercise 03
Reverse String	
Turn-in directory : *ex*03/	
Files to turn in : `strrev.py`	
Allowed functions : `None`	
Notes : `n/a`	

Ask the user to enter the string and call the strrev method to reverse tht input string.

- Function prototyped as `def strrev(input_string)`
- "!ihtagarP" returns "Pragathi!"

`How do you read the last character as the first character (LIFO)?`

```
?> python strrev.py
Enter the string to be reversed: !ihtagarP
Reversed String: Pragathi!
```

Chapter VIII

Exercise 04 : Queue question

	Exercise 04
Queue question	
Turn-in directory : *ex*04/	
Files to turn in : `printingTasks.py`	
Allowed functions : `None`	
Notes : `n/a`	

Ask the user to enter the string and call the strrev method to reverse tht input string.

- Function prototyped as `def strrev(input_string)`
- "!ihtagarP" returns "Pragathi!"

`How do you read the last character as the first character (LIFO)?`

```
?> python strrev.py
Enter the string to be reversed: !ihtagarP
Reversed String: Pragathi!
```

Chapter IX

Exercise 05 : Balance Parentheses

	Exercise 05
Balance Parentheses	
Turn-in directory : $ex05/$	
Files to turn in : `balanceCheck.py`	
Allowed functions : `None`	
Notes : `n/a`	

Ask the user to enter the sequence and check whether the input string is a balanced parentheses.

- Function prototyped as `def isBalanced(input_string)`
- Valid sequence will consist a combination of "([{}])" or an empty string
- Function returns True for balanced sequence, False for not balanced ssequence

```
?> python balanceCheck.py
Enter the sequence:
True
Enter the sequence: [{}]
True
Enter the sequence: (((((]
False
```

Chapter X

Exercise 06 : Base Converter

	Exercise 06
Base Converter	
Turn-in directory : $ex06/$	
Files to turn in : `baseConverter.py`	
Allowed functions : `None`	
Notes : `n/a`	

Ask the user to enter a decimal number and a base. Call the function baseConverter to convert the given decimal number to it's equivalent number in the give base system.

- Function prototyped as `def baseConverter(decNum, base)`
- We shall take care of just positive integers and base is between 2 to 16

```
?> python baseConverter.py
Enter the decimal number: 25
Enter the base: 16
19
```

`decNum is repeatedly divided by the base and pushed onto stack and then can be read from the top`

Chapter XI

Bonus Part

We are very proud of you for reaching till here! Now, are you up for the challenge?

XI.0.1 Evaluate Expression

	Exercise 07
Evaluate Expression	
Turn-in directory : *ex*07/	
Files to turn in : `evalExpr.py`	
Allowed functions : `None`	
Notes : `n/a`	

Get ready to build a simple mathematical expression parser!

- Function prototyped as `def evalExpr(expr)`
- expr is the mathematical expression of the form, for example (5 + 3) * 7 / (2 + 3), which is passed to the functionand 11 is returned as the result
- Ensure that you handle brackets as well!

```
?> python evalExpr.py
Enter the expression: (5 + 3) * 7 / (2 + 3)
11
```

XI.0.2 Queue Bonus

	Exercise 08
Evaluate Expression	
Turn-in directory : *ex*08/	
Files to turn in : `evalExpr.py`	
Allowed functions : `None`	
Notes : `n/a`	