**Game of Nim**

Professor Minhthong Nguyen

Students: Umar Khan, LeslieMarie Soliven, Anthony Diaz, Bianette Soriano, Gabriel Rodriguez

CECS 174

7th December 2016

**Introduction to the Program:**

This is the program that has designed the game of nim. The game of nim has been played between the user and the computer. There are two modes that the computer can play this game. The mode that the computer play smart or with the strategy is 1 and the dumb or without the strategy is 0. The game has a basic idea of picking up the cards from the piles which varies from 1 to half of the size of the pile. Anything beyond that limit shoots out an error to the user, where the computer never crosses this limit as the computer take has the strategy set to it in smart mode and random pick in dumb mode. In the end, whoever has to pick up the last card from the pile loses the game. The game in the end decides the winner and displays the winner on the screen.

**Pseudocode for the Program:**

**Methods.py Pseudocode**

Import random class

Function pileSize ()

Declare integer a

Set integer a = random.randint (10,100)

Return integer a

End function

Function firstPlayer ()

Declare integer b

Set integer b = random.randint (0,1)

Return integer b

End function

Function difficulty ()

Declare integer c

Set integer c = random.randint (0,1)

Return integer c

End function

Function computer\_take(pileSize)

Set integer comp\_take = 0

If difficulty = 1

If pileSize > 63 and pileSize <= 100

Set integer comp\_take = pileSize – 63

End if

Else if pileSize > 31 and pileSize <= 62

Set integer comp\_take = pileSize – 31

End else if

Else if pileSize > 15 and pileSize <= 30

Set integer comp\_take = pileSize -15

End else if

Else if pileSize > 7 and pileSize <= 14

Set integer comp\_take = pileSize – 7

End else if

Else if pileSize > 3 and pileSize <= 6

Set integer comp\_take = pileSize – 3

End else if

Else if pileSize > 1 and pileSize <= 2

Set integer comp\_take = pileSize – 1

End else if

Else

Set integer comp\_take = random.randint (1, pileSize//2)

End else

End if

Else

Set integer comp\_take = random.randint (1, pileSize//2)

End else

Return integer comp\_take

End function

**Game.py Pseudocode**

**From methods importing pileSize, firstPlayer, difficulty, computer\_take**

Declare integer pileSize

Set integer pileSize = function pileSize ()

Declare integer firstPlayer

Set integer firstPlayer = function firstPlayer ()

Declare integer difficulty

Set integer difficulty = function difficulty ()

Print (‘Pile size:’, pileSize, ‘First player:’, firstPlayer, ‘Difficulty:’, difficulty)

If firstPlayer == 0

Print (‘Computer goes first’)

While True

Print (‘Computers turn’)

Set integer pileSize = pileSize – function comp\_take(pileSize)

Print (‘New Pile size:’, pileSize)

If pileSize == 1

Print (‘Computer Wins’)

Break the condition

Set integer user\_take = eval (input (‘Enter how many you will take:’)

While (user\_take > pileSize//2) or user\_take <= 0

Set integer user\_take = eval (input (‘Please enter a valid enter how many you will take’)

End while

Set integer pileSize = pileSize – user\_take

Print (‘New pile size:’, pileSize)

If pileSize == 1

Print (‘You Win!’)

Break the condition

End while

End if

Else if firstPlayer == 1

Print (‘You go first’)

While True

Set integer user\_take = eval (input (‘Enter how many you will take’)

While (user\_take > pileSize //2) or user\_take <= 0

Set integer user\_take = eval (input (‘Please enter a valid enter how many you will take:’))

End while

Set integer pileSize = pileSize – user\_take

Print (‘New pile size:’, pileSize)

If pileSize == 1

Print (‘You Win!’)

Break the condition

Set integer pileSize = pileSize – function comp\_take(pileSize)

Print (‘New pile size:’, pileSize)

If pileSize == 1

Print (‘Computer Wins’)

Break the condition

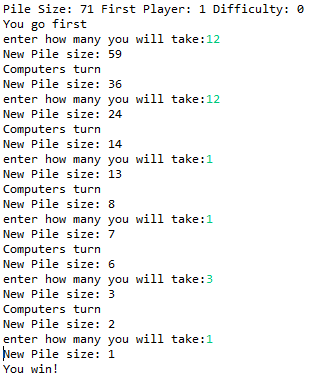
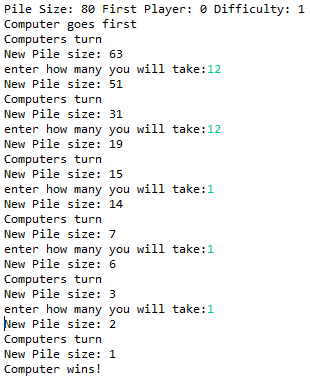
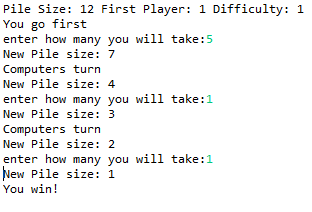
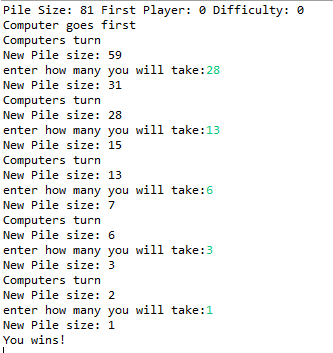
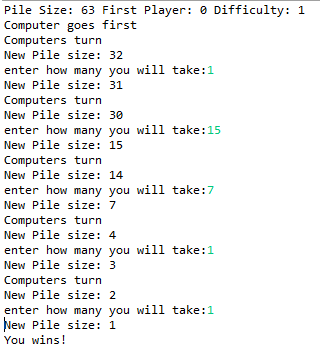
End while

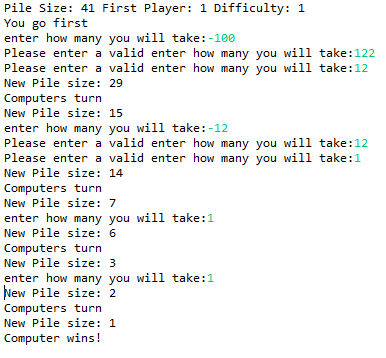
End else if

**Methods:**

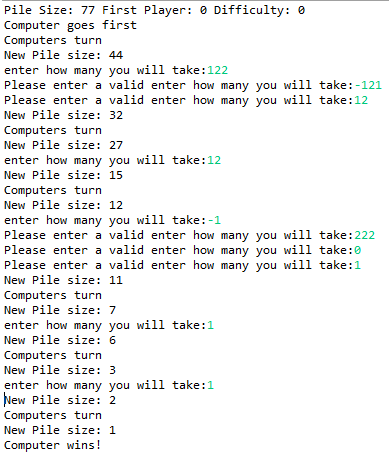
* **Def pileSize ():** This method generates a random number between 10 and 100, this number determines the size of the pile.
* **Def firstPlayer ():** This method generates a random integer between 0 and 1 to decide whether the computer or the user takes the first turn.
* **Def difficulty ():** This method generates a random integer between 0 and 1 to whether the computer plays smart or stupid. 1 is for the smart mode and 0 is for the dumb mode.
* **Def computer\_take (pileSize):** This method determines the number of cards taken by the computer depending on the level of difficulty generated by the difficulty function. If the game is in smart mode and the pile size is greater than 63 and less than or equal to 100 it will take away at most 63. If the pile size is between 31 and 62 the computer can take 31 at most. If the pile size is between 14 and 7 the computer can take 7 at most. If the pile size is between 7 and 3 you can take 3 at most. If the pile size is between 1 and 2, computer can take only 1 from the pile and the it will be the last else statement where the computer will determine randomly since there will be only one card left, and the program will decide the winner. If the game is played in dumb mode, the computer takes a random number between 1 and the pile size divided by 2. It follows the same strategy throughout the code.
* **Def main ():** The main for this function has been created as a tester for the game. The methods have been used in the game tester. The methods were imported into the game.py where we used the combination of the methods, loops, and if-else statements to test the game.

**Test Cases:**

* Test Case 1: When the computer is playing in dumb mode, and the user goes first 
* Test Case 2: When the computer is playing in smart mode, and it goes first. 
* Test Case 3: When the computer is playing in smart mode, and the user goes first so if the user plays with smart strategy, the user canwin the game. 
* Test Case 4: When the computer is playing in dumb mode, and the user goes first. For this case, there is a high chance that the user can win if he knows the strategy. 
* Test Case 5: When the computer is playing in smart mode, and the pileSize lands on a strategic number so the user has the chance to win unless he doesn’t plays smart. 
* Test Case 6: When the user goes first, testing for the input validation.Input validation works throughout the code.



* Test Case 7: When the computer goes first, testing for input validation. Input validation works throughout the code.



**Code of the Method.py:**

import random

#Generate a random integer between 10 and 100 to denote the initial size of the pile.

def **pileSize**():

a = random.randint(10,100)

return a

#Generate a random integer between 0 and 1 to decide whether the computer or the human takes the first turn.

#1 for player going first and 0 for computer going first

def **firstPlayer**():

b = random.randint(0,1)

return b

#Generate a random integer between 0 and 1 to decide whether the computer plays smart or stupid.

#1 for it being playing smart and 0 for playing dumb

def **difficulty**():

c = random.randint(0,1)

return c

#Computes how much computer has to take

def **computer\_take**(pileSize):

comp\_take=0

if difficulty == 1: #smart mode

if pileSize > 63 and pileSize <= 100:

comp\_take= pileSize- 63

elif pileSize > 31 and pileSize <= 62:

comp\_take= pileSize- 31

elif pileSize > 15 and pileSize <= 30:

comp\_take= pileSize- 15

elif pileSize > 7 and pileSize <= 14:

comp\_take= pileSize- 7

elif pileSize > 3 and pileSize <= 6:

comp\_take= pileSize- 3

elif pileSize > 1 and pileSize <=2:

comp\_take= pileSize - 1

else:

comp\_take = random.randint(1,pileSize//2)

else:# Stupid mode

comp\_take = random.randint(1,pileSize//2)

return comp\_take

**Code of the Game.py**

from methods import pileSize, firstPlayer, difficulty, computer\_take

pileSize = pileSize()

firstPlayer = firstPlayer()

difficulty = difficulty()

# 1 = smart -- 0 = dumb

print(*"Pile Size:"*, pileSize, *"First Player:"*, firstPlayer, *"Difficulty:"*, difficulty)

#When computer goes first

if firstPlayer == 0:

print(*"Computer goes first"*)

while True: #making sure pile is not less than 1

print(*"Computers turn"*)

pileSize = pileSize - computer\_take(pileSize)

print(*"New Pile size:"*, pileSize)

if pileSize == 1:

print(*'Computer wins!'*)

break

user\_take = int(input(*"enter how many you will take:"*))

while (user\_take > pileSize // 2) or user\_take <= 0:

user\_take = int(input(*"Please enter a valid enter how many you will take:"*))

pileSize = pileSize - user\_take

print(*"New Pile size:"*, pileSize)

if pileSize == 1:

print(*'You wins!'*)

break

#When the user goes first

elif firstPlayer == 1:

print(*"You go first"*)

while True: #making sure pile is not less than 1

user\_take = int(input(*"enter how many you will take:"*))

while (user\_take > pileSize // 2) or user\_take <= 0:

user\_take = int(input(*"Please enter a valid enter how many you will take:"*))

pileSize = pileSize - user\_take

print(*"New Pile size:"*, pileSize)

if pileSize == 1:

print(*'You win!'*)

break

print(*"Computers turn"*)

#print(computer\_take(pileSize))

pileSize = pileSize - computer\_take(pileSize)

print(*"New Pile size:"*, pileSize)

if pileSize == 1:

print(*'Computer wins!'*)

break