1] Guess TheNumber.py

```
import random print('NAME: MANOJ R \n USN:1AY24AI068 \n
SECTION: O') number = random.randint(1, 20) for
guessesTaken in range(1, 7):
   guess = int(input("Take a guess: "))
if guess < number:         print("Your
guess is too low.")     elif guess >
number:
      print("Your guess is too high.")
   else:
break  if guess ==
number:
   print(f"Good job! You guessed my number in {guessesTaken} guesses!")
else:
   print(f"Nope. The number I was thinking of was {number}.")
```

OUTPUT:

NAME: MANOJ R

USN:1AY24AI068

SECTION: O

Take a guess: 7

Your guess is too low.

Take a guess: 10

Your guess is too low.

Take a guess: 13

Your guess is too low.

Take a guess: 14

Good job! You guessed my number in 4 guesses!

2] RockPaperScissors.py

```
import random print('NAME: MANOJ R \n USN:1AY24AI068
```

```
\n SECTION: O')  moves = ['rock',
'paper', 'scissors'] while
True:     player = input("Enter rock, paper, scissors (or quit):
").lower()
   if player == 'quit':
      break
   if player not in moves:
print("Invalid     move.")       continue
computer = random.choice(moves)
print(f"Computer chose {computer}")     if
player == computer:
      print("It's a tie!")     elif (player == 'rock' and computer
== 'scissors') or \
       (player == 'paper' and computer == 'rock') or \
(player == 'scissors' and computer == 'paper'):
print("You win!")     else:
print("You lose.")
```

OUTPUT:

NAME: MANOJ R

USN:1AY24AI068

SECTION: O

Enter rock, paper, scissors (or quit): rock

Computer chose paper

You lose.

Enter rock, paper, scissors (or quit): scissor Invalid move.

Enter rock, paper, scissors (or quit): paper

Computer chose scissors

You lose.

Enter rock, paper, scissors (or quit): quit

3] ZigZag.py

```
import time, sys print('NAME: MANOJ R \n USN:1AY24AI068

\n SECTION: O') indent = 0 indentIncreasing = True


try:    while

True:

      print(' ' * indent + '* * * *')       time.sleep(0.1)

if

indentIncreasing:

         indent += 1

if indent == 20:

            indentIncreasing = False

      else:

         indent -= 1

if indent == 0:

            indentIncreasing = True except

KeyboardInterrupt:

   sys.exit()
```

OUTPUT:

NAME: MANOJ R

USN:1AY24AI068

SECTION: O

* * * *

* * * *

* * * *

* * * *

* * * *

* * * *

* * * *

* * * *

```
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
* * * *
```

4]CollatzSequence.py

```
print('NAME: MANOJ R \n USN:1AY24AI068 \n SECTION: O')
```

```
def collatz(number):     print(number)     if number == 1:
return
    elif number % 2 == 0:
        return collatz(number // 2)
    else:
        return collatz(3 * number + 1)


try:
    n = int(input("Enter a number: "))
    collatz(n)
except ValueError:
    print("Please enter an integer.")
```

OUTPUT:

NAME: MANOJ R

USN:1AY24AI068

SECTION: O

Enter a number: 2

2

1

5]ConWaysGameOfLife.py

```
import random, time, copy print('NAME: MANOJ R \n
USN:1AY24AI068 \n SECTION: O')
WIDTH = 60
```

```
HEIGHT = 20

nextCells = {} for x in range(WIDTH):
for y in range(HEIGHT):
      nextCells[(x, y)] = random.choice([True, False])

while True:
print('\n' * 5)
   cells = copy.deepcopy(nextCells)    for
y in range(HEIGHT):       for x
in range(WIDTH):
         if cells[(x, y)]:
print('#', end='')          else:
print(' ', end='')       print()

   for x in range(WIDTH):       for
y in range(HEIGHT):          left =
(x - 1) % WIDTH          right = (x
+ 1) % WIDTH          up = (y - 1)
% HEIGHT
down = (y + 1) % HEIGHT

         neighbors = 0          for nx, ny in [(left, up),
(x, up), (right, up),                    (left, y),
(right, y),
                  (left, down), (x, down), (right, down)]:
if cells[(nx, ny)]:                neighbors += 1          if cells[(x,
y)] and (neighbors == 2 or neighbors == 3):
```

```
            nextCells[(x, y)] = True            elif not
cells[(x, y)] and neighbors == 3:
nextCells[(x, y)] = True
          else:
             nextCells[(x, y)] = False
   time.sleep(1)
```

OUTPUT:

#

##

##

6]CommaCode.py

```
print('NAME: MANOJ R \n USN:1AY24AI068 \n SECTION: O')
def commaCode(items):
   if len(items) == 0:
return ''      elif
len(items) == 1:
return items[0]
   else:
      return ', '.join(items[:-1]) + ', and ' + items[-1]
print(commaCode(['apples', 'bananas', 'tofu', 'cats']))
```

OUTPUT:

NAME: MANOJ R

USN:1AY24AI068 SECTION: O

apples, bananas, tofu, and cats

7]CoinFlipStreaks.py

```
import random print('NAME: MANOJ R \n USN:1AY24AI068
\n SECTION: O') streaks = 0  for
experimentNumber in range(10000):
flips = [random.choice(['H', 'T']) for _ in
```

```
range(100)]     for i in range(94):         if
all(f == flips[i] for f in flips[i:i+6]):
            streaks += 1
break


print(f"Chance of streak: {streaks / 100}%")
```

OUTPUT:

NAME: MANOJ R

USN:1AY24AI068

SECTION: O

Chance of streak: 79.95%

8]CharacterPictureGrid.py

```
print('NAME: MANOJ R \n USN:1AY24AI068 \n SECTION: O')
grid = [['.', '.', '.', '.', '.', '.'],
['.', 'O', 'O', '.', '.', '.'],
        ['O', 'O', 'O', 'O', '.', '.'],
        ['O', 'O', 'O', 'O', 'O', '.'],
        ['.', 'O', 'O', 'O', 'O', 'O'],
        ['O', 'O', 'O', 'O', 'O', '.'],
        ['O', 'O', 'O', 'O', '.', '.'],
        ['.', 'O', 'O', '.', '.', '.'],
        ['.', '.', '.', '.', '.', '.']]


for x in range(len(grid[0])):
for y in range(len(grid)):
print(grid[y][x], end='')     print()
```

OUTPUT:

NAME: MANOJ R

USN:1AY24AI068

SECTION: O ..OO.OO..

.OOOOOOO.

.OOOOOOO.

..OOOOO..

...OOO...

....O....

9]ChessDictionaryValidator.py

```
print('NAME: MANOJ R \n USN:1AY24AI068 \n SECTION: O')
def isValidChessBoard(board):     piecesCount = {}     whiteKing
= blackKing = 0


   for pos, piece in board.items():         if pos[0] not in
'abcdefgh' or pos[1] not in '12345678':
return False         if piece not in ['wking', 'bking',
'wqueen', 'bqueen',
                    'wrook', 'brook', 'wbishop', 'bbishop',
'wknight', 'bknight', 'wpawn', 'bpawn']:            return False


       piecesCount[piece] = piecesCount.get(piece, 0) + 1


   if piecesCount.get('wking', 0) != 1 or piecesCount.get('bking', 0) != 1:
return False     return True


# Example usage board
= {
   '1h': 'bking', '6c': 'wqueen', '2g': 'bbishop',
   '5h': 'bqueen', '3e': 'wking'
}
print(isValidChessBoard(board))
```

OUTPUT:

NAME: MANOJ R

USN:1AY24AI068

SECTION: O

False

10]FantasyGameInventory.py

```
print('NAME: MANOJ R \n USN:1AY24AI068 \n SECTION: O')

def displayInventory(inventory):

print("Inventory:")     total =

0     for k, v in

inventory.items():

       print(f"{v} {k}")

total += v      print(f"Total number of

items: {total}")


def addToInventory(inventory, addedItems):     for

item in addedItems:

       inventory[item] = inventory.get(item, 0) + 1     return

inventory


inv = {'gold coin': 42, 'rope': 1}  dragonLoot = ['gold coin',

'dagger', 'gold coin', 'gold coin', 'ruby'] inv = addToInventory(inv,

dragonLoot) displayInventory(inv)
```

OUTPUT:

NAME: MANOJ R

USN:1AY24AI068

SECTION: O Inventory:

45 gold coin

1 rope

1 dagger

1 ruby

Total number of items: 48

11]TablePrinter.py

```
print('NAME: MANOJ R \n USN:1AY24AI068 \n SECTION: O') def printTable(tableData):
colWidths = [max(len(item) for item in col) for col in tableData]

   for row in range(len(tableData[0])):        for
col in range(len(tableData)):
          print(tableData[col][row].rjust(colWidths[col]), end=' ')
      print()

tableData = [['apples', 'oranges', 'cherries', 'banana'],
           ['Alice', 'Bob', 'Carol', 'David'],
['dogs', 'cats', 'moose', 'goose']]
printTable(tableData) OUTPUT:
NAME: MANOJ R
 USN:1AY24AI068
SECTION: O    apples
Alice  dogs   oranges
Bob  cats  cherries
Carol moose
banana David goose
```

12]ZombieDiceBots.py

```
print('NAME: MANOJ R \n USN:1AY24AI068 \n SECTION: O')
import random

class ZombieDiceBot:
```

```
   """Base class for a Zombie Dice bot."""
def __init__(self, name):        self.name =
name

   def should_roll(self, brain_count, shotguns_count,
turn_rolls_history):        raise NotImplementedError("Subclasses
must implement the should_roll method.")

   def __str__(self):
return self.name

class BasicBot(ZombieDiceBot):
   def should_roll(self, brain_count, shotguns_count,
turn_rolls_history):
       return brain_count < 1

class RiskyBot(ZombieDiceBot):
   def should_roll(self, brain_count, shotguns_count,
turn_rolls_history):
       return shotguns_count < 3

class CautiousBot(ZombieDiceBot):
   def should_roll(self, brain_count, shotguns_count,
turn_rolls_history):
       return brain_count < 2

class RandomBot(ZombieDiceBot):
   def should_roll(self, brain_count, shotguns_count,
turn_rolls_history):
       return random.choice([True, False])

class BrainGreedyBot(ZombieDiceBot):
   def should_roll(self, brain_count, shotguns_count,
turn_rolls_history):
       return shotguns_count < 3

def roll_dice():
   dice_colors = ['green'] * 6 + ['yellow'] * 4 + ['red'] * 3
   rolled_dice = random.sample(dice_colors, 3)
results = []     for color in rolled_dice:        if
color == 'green':
          outcomes = ['brain'] * 3 + ['shotgun'] * 1 + ['runner'] * 2
elif color == 'yellow':
```

```
        outcomes = ['brain'] * 2 + ['shotgun'] * 2 + ['runner'] * 2
else:  # red
        outcomes = ['brain'] * 1 + ['shotgun'] * 3 + ['runner'] * 2
      results.append(random.choice(outcomes))
   return tuple(results)

def play_turn(bot):    print(f"\n--- {bot.name}'s turn --
-")    brains_this_turn = 0    shotguns_this_turn = 0
turn_rolls_history = []    while shotguns_this_turn < 3
and bot.should_roll(brains_this_turn,
shotguns_this_turn, turn_rolls_history):
      input(f"{bot.name} decides to roll. Press Enter to roll...")
      roll_result = roll_dice()
turn_rolls_history.append(roll_result)
print(f"{bot.name} rolled: {', '.join(roll_result)}")
for result in roll_result:           if result == 'brain':
            brains_this_turn += 1           elif result ==
'shotgun':              shotguns_this_turn += 1
print(f"Brains this turn: {brains_this_turn}")
print(f"Shotguns this turn: {shotguns_this_turn}")
if shotguns_this_turn >= 3:
        print(f"{bot.name} got zombied out!")          return
0    print(f"{bot.name} decided to stop. Total brains this
turn:
{brains_this_turn}")
   return brains_this_turn

def run_game(bots, num_turns=5):
scores = {bot.name: 0 for bot in bots}
for turn in range(1, num_turns + 1):
for bot in bots:
        brains_earned = play_turn(bot)
scores[bot.name] += brains_earned
print(f"{bot.name}'s total score: {scores[bot.name]}")
      print(f"\n--- End of Turn {turn} ---")
print("Current Scores:")       for name,
score in scores.items():
        print(f"{name}: {score}")
print("\n--- Game Over ---")    print("Final
Scores:")
   for name, score in scores.items():
      print(f"{name}: {score}")
```

```
if __name__ == "__main__":     print(' Name: MANOJ R\n
USN: 1AY24AI068 \n Section: O')     bot1 = BasicBot("Basic
Bot")     bot2 = RiskyBot("Risky Bot")     players = [bot1,
bot2]     run_game(players, num_turns=3)
```

OUTPUT:

```
Name: MANOJ
 USN: 1AY24AI068
 Section: O

--- Basic Bot's turn ---
Basic Bot decides to roll. Press Enter to roll...
Basic Bot rolled: brain, runner, shotgun
Brains this turn: 1
Shotguns this turn: 1
Basic Bot decided to stop. Total brains this turn: 1
Basic Bot's total score: 1

--- Risky Bot's turn ---
Risky Bot decides to roll. Press Enter to roll...
Risky Bot rolled: brain, brain, runner
Brains this turn: 2
Shotguns this turn: 0
Risky Bot decides to roll. Press Enter to roll...
Risky Bot rolled: shotgun, brain, runner
Brains this turn: 3
Shotguns this turn: 1
Risky Bot decides to roll. Press Enter to roll...
Risky Bot rolled: shotgun, runner, shotgun
Brains this turn: 3
Shotguns this turn: 3 Risky
Bot got zombied out!
Risky Bot's total score: 0

--- End of Turn 1 --- Current
Scores:
Basic Bot: 1
Risky Bot: 0

--- Basic Bot's turn ---
Basic Bot decides to roll. Press Enter to roll...
Basic Bot rolled: shotgun, runner, brain
Brains this turn: 1
Shotguns this turn: 1
Basic Bot decided to stop. Total brains this turn: 1
Basic Bot's total score: 2

--- Risky Bot's turn ---
Risky Bot decides to roll. Press Enter to roll...
Risky Bot rolled: brain, shotgun, runner
Brains this turn: 1
Shotguns this turn: 1
Risky Bot decides to roll. Press Enter to roll...
Risky Bot rolled: brain, shotgun, runner
Brains this turn: 2
Shotguns this turn: 2
Risky Bot decides to roll. Press Enter to roll...
```

Risky Bot rolled: shotgun, brain, runner
Brains this turn: 3
Shotguns this turn: 3 Risky
Bot got zombied out!
Risky Bot's total score: 0

--- End of Turn 2 --- Current
Scores:
Basic Bot: 2
Risky Bot: 0

--- Basic Bot's turn ---
Basic Bot decides to roll. Press Enter to roll...
Basic Bot rolled: brain, runner, shotgun
Brains this turn: 1
Shotguns this turn: 1
Basic Bot decided to stop. Total brains this turn: 1
Basic Bot's total score: 3

--- Risky Bot's turn ---
Risky Bot decides to roll. Press Enter to roll...
Risky Bot rolled: shotgun, runner, brain
Brains this turn: 1
Shotguns this turn: 1
Risky Bot decides to roll. Press Enter to roll...
Risky Bot rolled: brain, shotgun, runner
Brains this turn: 2
Shotguns this turn: 2
Risky Bot decides to roll. Press Enter to roll...
Risky Bot rolled: shotgun, brain, runner
Brains this turn: 3
Shotguns this turn: 3 Risky
Bot got zombied out!
Risky Bot's total score: 0

--- End of Turn 3 --- Current
Scores:
Basic Bot: 3
Risky Bot: 0

--- Game Over --- Final
Scores:
Basic Bot: 3 Risky
Bot: 0