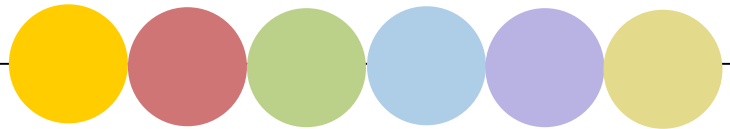
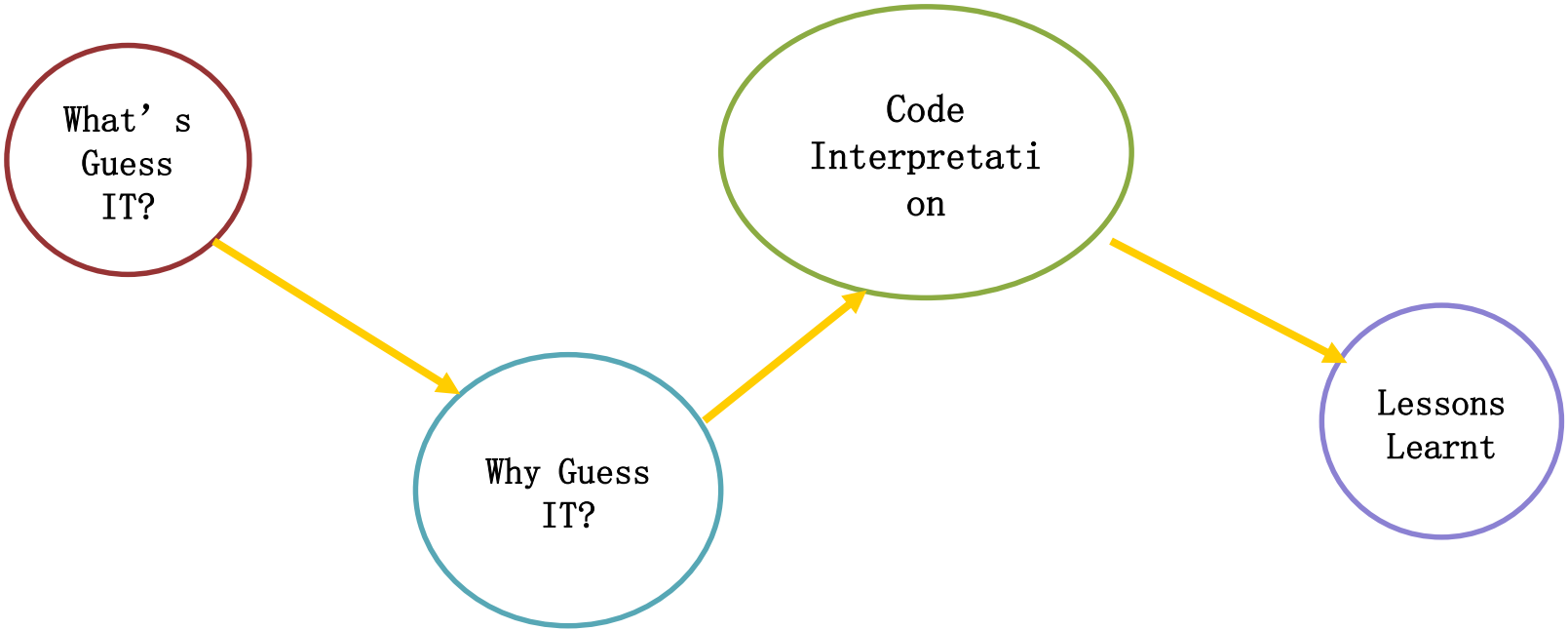
Guess IT
The Pythonic Alternative



Chia Jun Jie, Claris Chong Ka Shing, Igor Ivankin, Shane Bradley

Contents


What’ s Guess IT?
Why Guess IT?
Code Interpretation
Lessons Learnt

1

What’s Guess IT?

What’s Guess IT?

- A logic puzzle that focusses on deductive reasoning.
- Player gets 8 consecutive guesses to deduce the correct color pattern that was computer generated.

2

Why Guess IT?

Why Guess IT?

- Excellent Logic Puzzle
- Scarcely heard of among Chinese students
- Simple yet effective use of the Pygame library

3 Game Demonstration

4 Code Interpretation

class Button(pygame.sprite.Sprite)

- Gives the ability to work with the images
- Set the positions for the images
- Defines when certain image is pressed


```
class Button(pygame.sprite.Sprite):
    def __init__(self,image):
        pygame.sprite.Sprite.__init__(self)
        self.image, self.rect = load_image(image)
    def setCords(self,x,y):
        self.rect.topleft = x,y
        screen.blit(self.image, (x,y))
    def pressed(self,mouse):
        if mouse[0] > self.rect.topleft[0]:
            if mouse[1] > self.rect.topleft[1]:
                if mouse[0] < self.rect.bottomright[0]:
                    if mouse[1] < self.rect.bottomright[1]:
                        return True
                    else: return False
                else: return False
            else: return False
        else: return False
```

class Board

```
class Board:
    def __init__(self):
        self.board = ["e  e  e  e",
                      "e  e  e  e",
                      "e  e  e  e",
                      "e  e  e  e",
                      "e  e  e  e",
                      "e  e  e  e",
                      "e  e  e  e",
                      "e  e  e  e",]
        self.bwboard = ["e  e  e  e",
                        "e  e  e  e",
                        "e  e  e  e",
                        "e  e  e  e",
                        "e  e  e  e",
                        "e  e  e  e",
                        "e  e  e  e",
                        "e  e  e  e",]
        self.guess = ["e", "e", "e", "e"]
        self.gx = 270
        self.gy = 260
        self.gby = 420
        self.bluebut = Button('/Users/ivankinigor/image/bluepeg.png')
        self.redbut = Button('/Users/ivankinigor/image/redpeg.png')
        self.yelbut = Button('/Users/ivankinigor/image/yellowpeg.png')
        self.orbut = Button('/Users/ivankinigor/image/orangepeg.png')
        self.purpbut = Button('/Users/ivankinigor/image/purppeg.png')
        self.greenbut = Button('/Users/ivankinigor/image/greenpeg.png')
        self.submitbut = Button('/Users/ivankinigor/image/submit.png')
        self.font = pygame.font.SysFont('agencyfb', 18)
        self.font2 = pygame.font.SysFont('agencyfb', 24)
```

- Board = main game board
- Bwboard = hint board
- This sets the location of the left and right parts of the board, and uploads the images onto the board.

def drawboard(self)

```
def drawboard(self):
    self.guessline = 1
    self.bx = 30
    self.by = 100
    for row in self.board:
        self.g = str(self.guessline)
        self.text = self.font.render(self.g, 1, (10, 10, 10))
        screen.blit(self.text, (self.bx - 15, self.by))
        self.guessline += 1
        pygame.display.update()
        for col in row:
            if col == "e":
                screen.blit(empty_peg, (self.bx, self.by))
            elif col == "r":
                screen.blit(red_peg, (self.bx, self.by))
            elif col == "b":
                screen.blit(blue_peg, (self.bx, self.by))
            elif col == "g":
                screen.blit(green_peg, (self.bx, self.by))
            elif col == "p":
                screen.blit(purple_peg, (self.bx, self.by))
            elif col == "y":
                screen.blit(yellow_peg, (self.bx, self.by))
            elif col == "o":
                screen.blit(orange_peg, (self.bx, self.by))
            else:
                continue
            self.bx += 35
        self.by += 35
        self.bx = 30
        pygame.display.flip()
```

- Sets the drawboard location according to x, y.
- Defines the movement of the colours from the current guess to the display board

def drawbw(self) & def colourbin(self)

```
def drawbw(self):
    self.bwx = 175
    self.bwy = 110
    for row in self.bwboard:
        for col in row:
            if col == "e":
                screen.blit(bw_empty, (self.bwx, self.bwy))
            elif col == "b":
                screen.blit(bw_black, (self.bwx, self.bwy))
            elif col == "w":
                screen.blit(bw_white, (self.bwx, self.bwy))
            else:
                continue
            self.bwx += 18
        self.bwy += 35
        self.bwx = 175
        pygame.display.flip()


  def colorbin(self):
      pygame.draw.rect(screen, BLACK, (self.gx + 3, self.gy + 3, 90,110))
      pygame.draw.rect(screen, GREY, (self.gx,self.gy,90,110))
      self.redbut.setCords(self.gx+10,self.gy+5)
      self.orbut.setCords(self.gx+50,self.gy+5)
      self.yelbut.setCords(self.gx+10, self.gy +40)
      self.greenbut.setCords(self.gx+50, self.gy+40)
      self.bluebut.setCords(self.gx+10, self.gy+75)
      self.purpbut.setCords(self.gx+50, self.gy+75)
      pygame.display.update()
```

- **def drawbin(self):**

Sets the hint board location
Gives the results of the guesses according to the main board based on your guesses

- **def colorbin(self):**

Sets the colour board colours

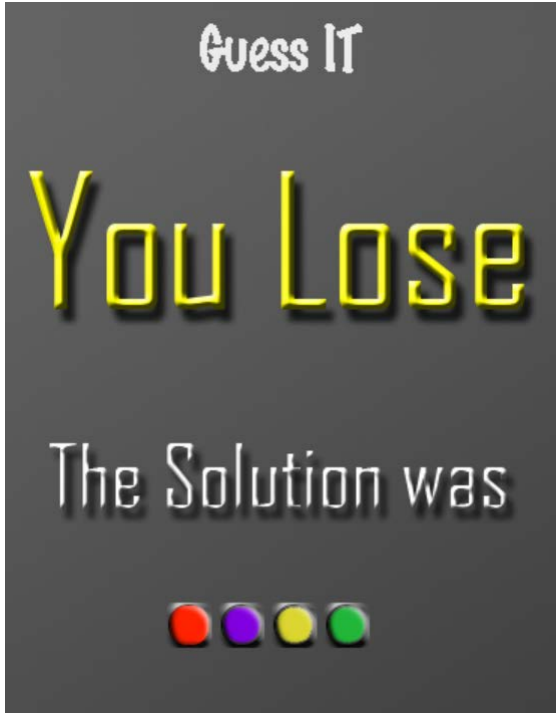
def guessdisplay(self) & def pegcheck(self, guess)


```
def guessdisplay(self):
    self.bx = 30
    for row in self.guess:
        if row == "e":
            screen.blit(empty_peg, (self.bx, self.gby))
        elif row == "r":
            screen.blit(red_peg, (self.bx, self.gby))
        elif row == "b":
            screen.blit(blue_peg, (self.bx, self.gby))
        elif row == "g":
            screen.blit(green_peg, (self.bx, self.gby))
        elif row == "p":
            screen.blit(purple_peg, (self.bx, self.gby))
        elif row == "y":
            screen.blit(yellow_peg, (self.bx, self.gby))
        elif row == "o":
            screen.blit(orange_peg, (self.bx, self.gby))
        else:
            continue
        self.bx += 35
        pygame.display.flip()
```

```
def pegcheck(self, guess):
    self.strikes1 = []
    self.strikes2 = []
    self.blackpeg=0
    self.whitepeg=0
    self.bwcount = []
    for i in range(len(guess)):
        if guess[i] == solution[i]:
            self.blackpeg += 1
            self.strikes1.append(i)
            self.strikes2.append(i)
            self.bwcount.append("b")
    for x in range(len(solution)):
        for y in range(len(solution)):
            if x not in self.strikes1 and y not in self.strikes2:
                if guess[x] == solution[y]:
                    self.whitepeg += 1
                    self.strikes1.append(x)
                    self.strikes2.append(i)
                    self.bwcount.append("w")
        self.bwboard[turn] = self.bwcount
```

def lose(self)

```
def lose(self):
    screen.blit(losebg, (0,0))
    self.bx = 115
    for row in solution:
        if row == "e":
            screen.blit(empty_peg, (self.bx, 400))
        elif row == "r":
            screen.blit(red_peg, (self.bx, 400))
        elif row == "b":
            screen.blit(blue_peg, (self.bx, 400))
        elif row == "g":
            screen.blit(green_peg, (self.bx, 400))
        elif row == "p":
            screen.blit(purple_peg, (self.bx, 400))
        elif row == "y":
            screen.blit(yellow_peg, (self.bx, 400))
        elif row == "o":
            screen.blit(orange_peg, (self.bx, 400))
        else:
            continue
        self.bx += 35
    pygame.display.update()
    pygame.time.delay(5000)
    exit()
```



def win(self)

```
def win(self):
    screen.blit(winbg, (0,0))
    pygame.display.flip()
    pygame.time.delay(5000)
    exit()
```

def getanswer() & def drawbg()

```
def getanswer():
    #generates the solution which the player must guess
    availcolors = ("r", "o", "y", "g", "b", "p")
    answer = [random.choice(availcolors) for i in range(4)]
    return answer
```

```
def drawbg():
    background, bg_rect = load_image('/Users/ivankinigor/image/mmbg2.jpg')
    screen.blit(background, (0,0))
    pygame.draw.line(screen, BLACK, (30, 55), (350, 55), 2)
    pygame.draw.line(screen, DKGREY, (32,57), (352, 57), 2)
    pygame.draw.line(screen, BLACK, (30, 380), (350, 380), 2)
    pygame.draw.line(screen, DKGREY, (32,382), (352, 382), 2)
    heading_text = font.render("Guesses", 1, (10, 10, 10))
    heading_textpos = (67, 65)
    current_guess_text = font.render('Current Guess', 1, (10,10,10))
    current_guesspos = (44, 385)
    screen.blit(current_guess_text, current_guesspos)
    screen.blit(heading_text, heading_textpos)
    pygame.display.update()
```

- **Def getanswer():**

Radom set of colour combination is generated

- Def drawbg():

The design set up for color board, guessed board and main game board

Loading/ Setting of Images


```
empty_peg, empty_rect = load_image('/Users/ivankinigor/image/mmempty.png')
red_peg, red_peg_rect = load_image('/Users/ivankinigor/image/redpeg.png')
blue_peg, blue_peg_rect = load_image('/Users/ivankinigor/image/bluepeg.png')
green_peg, green_peg_rect = load_image('/Users/ivankinigor/image/greenpeg.png')
purple_peg, purple_peg_rect = load_image('/Users/ivankinigor/image/purppeg.png')
yellow_peg, yellow_peg_rect = load_image('/Users/ivankinigor/image/yellowpeg.png')
orange_peg, orange_peg_rect = load_image('/Users/ivankinigor/image/orangepeg.png')
bw_empty, bw_empty_rect = load_image('/Users/ivankinigor/image/bwempty1.png')
bw_white, bw_white_rect = load_image('/Users/ivankinigor/image/bwwhite.png')
bw_black, bw_black_rect = load_image('/Users/ivankinigor/image/bwblack.png')
winbg, winbg_rect = load_image('/Users/ivankinigor/image/mmbgwin.jpg')
losebg, losebg_rect = load_image('/Users/ivankinigor/image/mmbglose.jpg')
```

- Depending on what the user selects, the according image will be loaded on the screen

Game Play

- Sets the game to be solved for 8 steps
- Sets system to recognize the color pressed with the corresponding guess
- Sets the parameters for he guess can only be 4 colors

```
while turn <= 8:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        elif turn == 8:
            board.lose()
        elif event.type == MOUSEBUTTONDOWN and guessnum < 4:
            pos = pygame.mouse.get_pos()
            if board.bluebut.pressed(pos) == True:
                board.guess[guessnum] = "b"
                board.guessdisplay()
                guessnum += 1
            elif board.yelbut.pressed(pos) == True:
                board.guess[guessnum] = "y"
                board.guessdisplay()
                guessnum += 1

            elif board.orbut.pressed(pos) == True:
                board.guess[guessnum] = "o"
                board.guessdisplay()
                guessnum += 1

            elif board.purpbut.pressed(pos) == True:
                board.guess[guessnum] = "p"
                board.guessdisplay()
                guessnum += 1

            elif board.greenbut.pressed(pos) == True:
                board.guess[guessnum] = "g"
                board.guessdisplay()
                guessnum += 1

            elif board.redbut.pressed(pos) == True:
                board.guess[guessnum] = "r"
                board.guessdisplay()
                guessnum += 1
            else:
                continue
        elif guessnum == 4:
            board.board[turn] = board.guess
            board.drawboard()
            board.pegcheck(board.guess)
            board.drawbw()
            board.guess = ["e", "e", "e", "e"]
            turn += 1
            board.guessdisplay()
            board.drawbw()
            guessnum = 0
            if board.blackpeg == 4:
                board.win()
                break
        else:
            continue
```

5

Lessons Learnt & Challenges Faced

Lessons Learnt

Importance of defining variables

- Should the variable be also defined as a class, it will cause an error when the code is ran.
- Especially after importing Pygame, some terms are part of the pygame library and cannot be defined as other variables.

Efficient codes are easier to spot error

- When our codes were longer it was more difficult to pinpoint error whenever it is ran.

Challenges Faced

Pygame

- As we had no experience learning Pygame in class, a lot of our Pygame knowledge had to be learnt from online resources.

OS Problems

- The coding process was difficult as some codes were not able to be ran on the Mac OS (or Windows) and a lot of time was spent troubleshooting.

Thanks!

Any questions ?