01076103, 01076104

Programming Fundamental

Programming Project

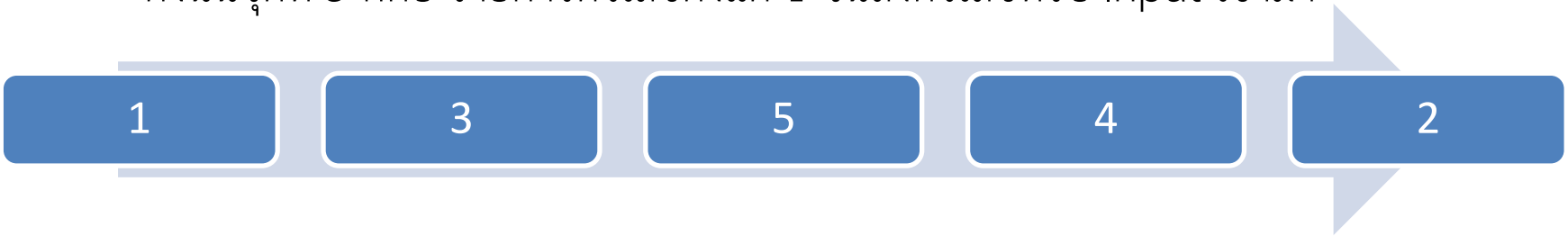
Problem Solving

Problem Solving

- การเขียนโปรแกรม จะต้องมีทักษะการแก้ปัญหาที่ดี ทักษะการแก้ปัญหา ก็คือทักษะการคิดอย่างหนึ่งที่ต้องฝึกฝน โดยมีรูปแบบการคิดดังนี้
  - การคิดตามลำดับตั้งแต่ต้นจนจบ เป็นการคิดที่เราทราบกระบวนการทำงานที่แน่นอน ตั้งแต่ 1 จนถึงขั้นตอนสุดท้าย
  - การคิดแบบเชื่อมต่อจุดหมายย่อย ในกรณีที่การทำงานมีหลายขั้นตอนและซับซ้อน อาจคิดแบบเรียงลำดับไม่ได้ ให้ทำลักษณะของการต่อ jigsaw คือเริ่มจากงานที่มองออกว่าต้องมี และเขียนเป็นส่วนๆ ให้เสร็จแล้วค่อยๆ นำไปสู่งานที่สมบูรณ์
  - การคิดแบบโจทย์แบบง่าย ในกรณีที่งานมีขนาดใหญ่ เราอาจจะเริ่มจากงานที่มีขนาดเล็กที่คล้ายๆ กัน เช่น หากมีสินค้า 20 ประเภท เราอาจทำเพียงประเภทเดียวก่อน แล้วค่อยขยายออกไปอีกที
  - การคิดแบบย้อนกลับจากหลังมาหน้า บางครั้งเราอาจนึกวิธีการไม่ออก เช่น มีโจทย์ที่ถามว่า ตัวเลขที่รับมาเป็นจำนวนเฉพาะหรือไม่ ก็ต้องถามว่าจำนวนเฉพาะคืออะไร ถ้ารู้ว่าจำนวนเฉพาะคืออะไร ก็จะหาคำตอบได้

Problem Solving

- **วิเคราะห์โจทย์** ตัวเลขที่รับมาเป็นจำนวนเฉพาะหรือไม่
  - จำนวนเฉพาะ คือ จำนวนที่ไม่มีตัวเลขใดหารลงตัว นอกจาก 1 และตัวมันเอง
  - จุดที่ 1 คือ จุดเริ่มต้นของการทำงาน
  - จุดที่ 2 คือ จุดสิ้นสุดของการทำงาน ซึ่งจะได้คำตอบว่า ตัวเลขเป็นจำนวนเฉพาะหรือไม่
  - จุดที่ 3 การรับข้อมูล เพื่อนำมาเข้ากระบวนการ
  - จุดที่ 4 คือ จุดที่เป็นกระบวนการคิดย้อนกลับ เนื่องจากจุดที่ 2 คือ ต้องรู้ว่าตัวเลขที่รับมาเป็นจำนวนเฉพาะหรือไม่ ดังนั้นจุดนี้ต้องทราบว่าตัวเลขที่รับมามีอะไรหารลงตัวบ้าง
  - ดังนั้นจุดที่ 5 ก็คือ รายการตัวเลขตั้งแต่ 1 จนถึงตัวเลขที่รับ Input เข้ามา



Problem Solving

- จากผังความคิดทั้ง 5 จุด จะนำมาทำงานต่อเพื่อสร้างโค้ดเทียม
  - เริ่มจากจุด 1 ไปยังจุด 3 เป็นการรับข้อมูล ดังนั้นโค้ดเทียมคือ

    รับข้อมูล (a)
  - จากจุด 3 ไปยังจุด 5 คือ การนำข้อมูล a ที่รับมาไปสร้างลำดับตัวเลขตั้งแต่ 1 – a ดังนั้นโค้ดเทียมคือ

    กำหนดตัวนับ i=1

    ทำงานเมื่อ (i <= a)

    i = I +1

Problem Solving

- จากผังความคิดทั้ง 5 จุด จะนำมาทำงานต่อเพื่อสร้างโค้ดเทียม
  - จากจุด 5 ไปจุด 4 คือ งานที่มีตัวเลขจาก 1 ถึง a จนทราบว่าตัวเลขใดที่หาร a ลงตัวบ้าง โดยต้องมีการนับว่าตัวเลขที่หารลงตัวมีจำนวนกี่ตัว เนื่องจากเป็นกระบวนการที่คล้ายเดิม จึงใช้วิธีนำโค้ดเทียมเดิมมาแก้ไข

กำหนดตัวนับ i=1 เพื่อเป็นรายการตัวเลขตั้งแต่ 1-a

กำหนดตัวนับ c เพื่อนับตัวเลขที่หาร a ลงตัว เริ่มต้น c=0

ทำงานเมื่อ (i <= a)

ถ้า (a/i) ลงตัว c = c+1 เพื่อแสดงว่าตัวเลขที่หาร a ลงตัวเพิ่มขึ้น 1

i = I +1

Problem Solving

- จากผังความคิดทั้ง 5 จุด จะนำมาทำงานต่อเพื่อสร้างโค้ดเทียม
  - ดังนั้นจากที่จุด 4 ไป 2 ก็จะได้ตัวแปร c ซึ่งเป็นจำนวนที่หาร a ลงตัว ถ้า c = 2 ก็จะบอกได้ว่าเป็นจำนวนเฉพาะ

    ถ้า (c=2) แสดงผลว่า a คือ จำนวนเฉพาะ

Problem Solving

- เมื่อรวมโค้ดเทียมทั้งหมด จะได้ดังนี้

```
รับข้อมูล (a)
กำหนดตัวนับ i=1 เพื่อเป็นรายการตัวเลขตั้งแต่ 1-a
กำหนดตัวนับ c เพื่อนับตัวเลขที่หาร a ลงตัว เริ่มต้น c=0
ทำงานเมื่อ (i <= a)
    ถ้า (a/i) ลงตัว c = c+1 เพื่อแสดงว่าตัวเลขที่หาร a ลงตัวเพิ่มขึ้น 1
    i = I +1
ถ้า (c=2) แสดงผลว่า a คือ จำนวนเฉพาะ
```

Problem Solving

- นำมาเขียนเป็นโปรแกรมได้ดังนี้

main.py

```
1  a = int(input("Enter Number : "))
2  c = 0
3  i = 1
4  while (i<=a):
5      if (a % i) == 0:
6          c += 1
7      i += 1
8
9  if c==2:
10     print("Prime")
11 else:
12     print("Not prime")
13
```

Console Shell

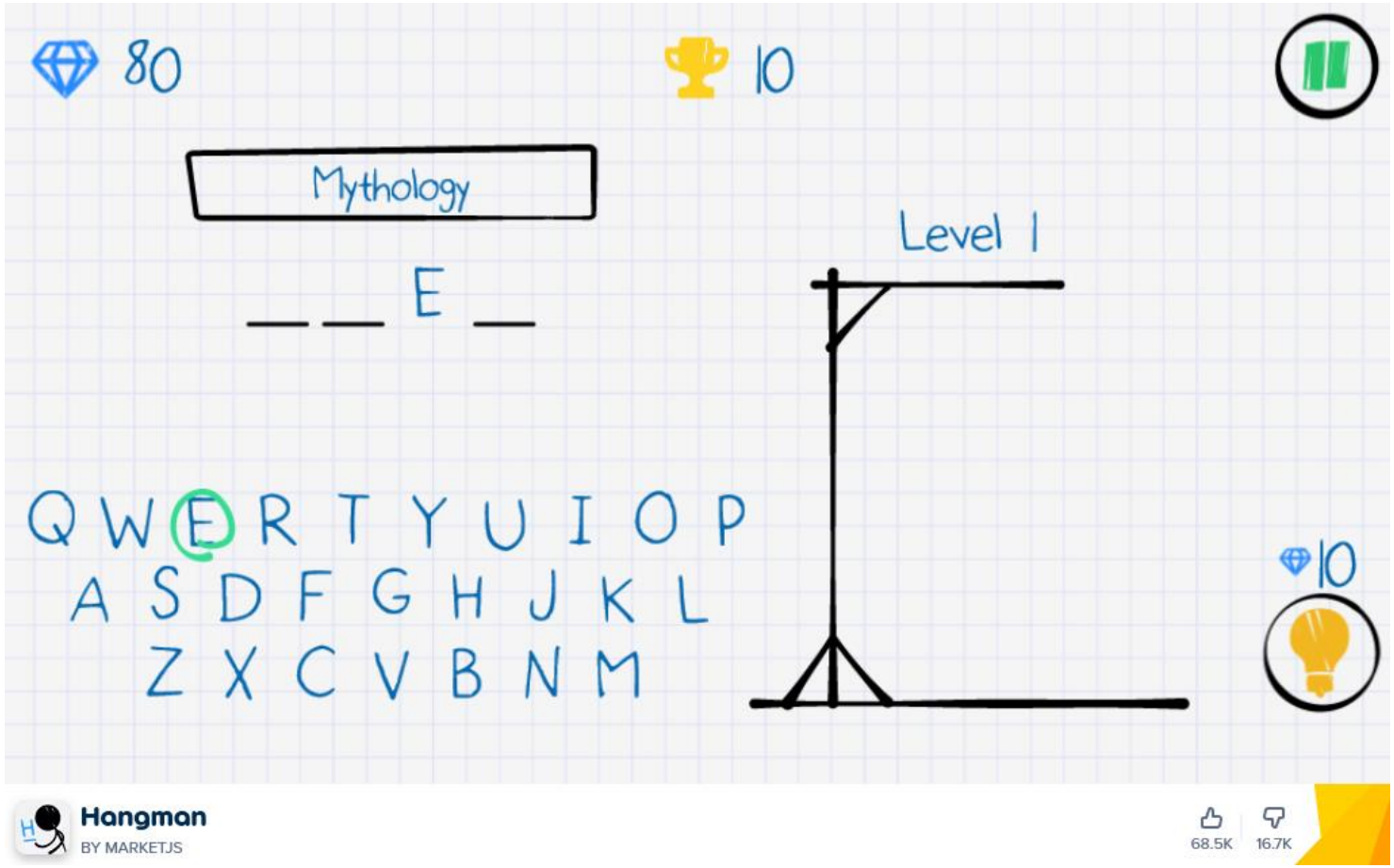
```
Enter Number : 13
2
Prime
```

Problem Solving

- **Exercise** จากโจทย์ต่อไปนี้ ลองพิจารณาดูว่า เราใช้วิธีใดในการแก้ไขปัญหา
  - **4.1** 2520 คือ ตัวเลขที่น้อยที่สุด ที่สามารถหารด้วยตัวเลขทุกตัวตั้งแต่ 1-10 จงหาจำนวนเต็มบวกที่น้อยที่สุดที่หารด้วยตัวเลขทุกตัวตั้งแต่ 1-20 (ในการทดสอบใช้แค่ 15 ก็พอ เพราะรันนาน)
  - **4.2** จำนวนเฉพาะ (Prime Number) คือตัวเลขที่มีแต่ 1 กับตัวมันเองที่หารลงตัว โดยจำนวนเฉพาะ 6 ตัวแรกคือ 2, 3, 5, 7, 11, 13 โดยจำนวนเฉพาะตัวที่ 6 คือ 13 จงหาจำนวนเฉพาะตัวที่ 101
-

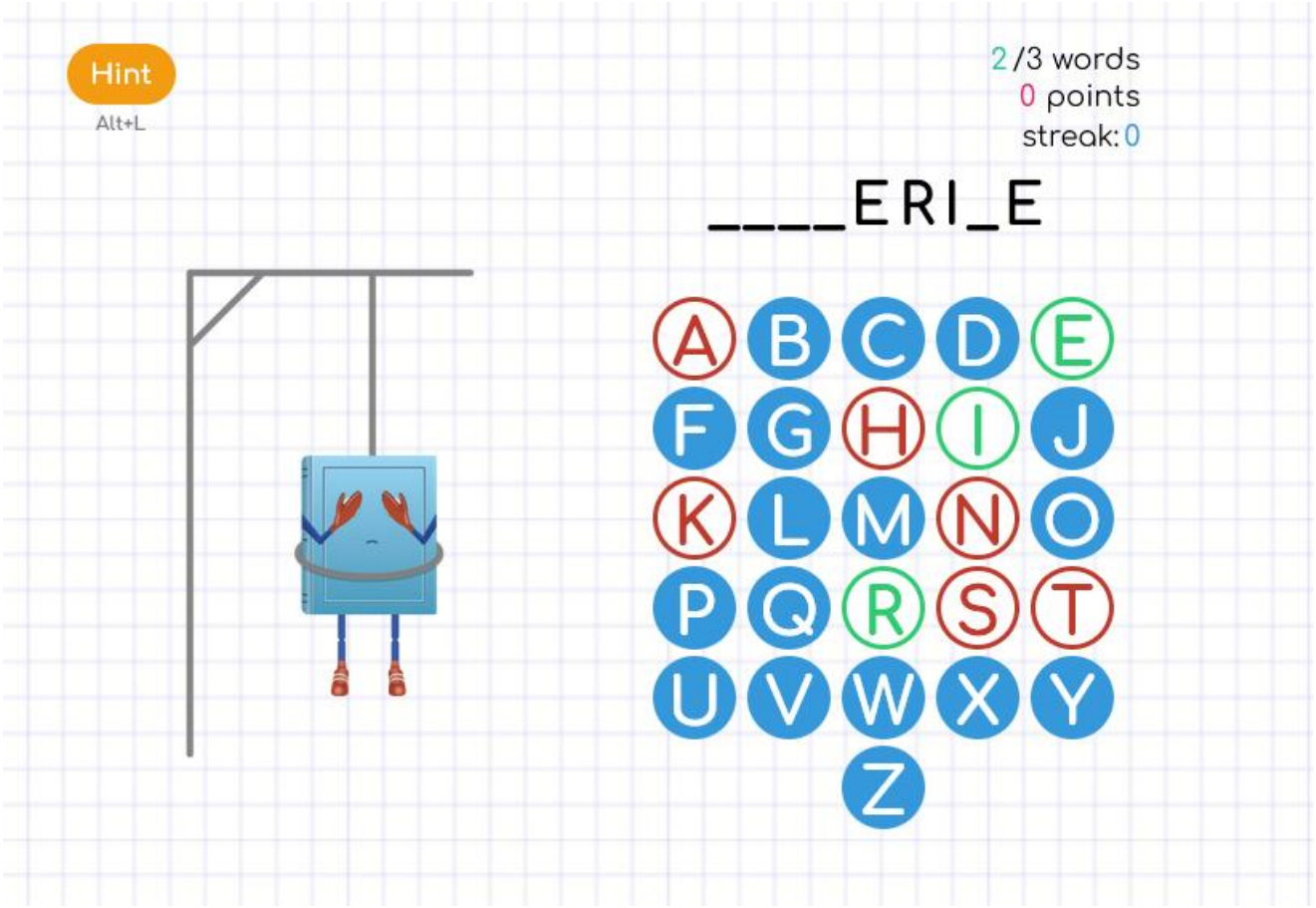
Problem Solving

- ตัวอย่าง โปรแกรม Hangman https://poki.com/th/g/hangman



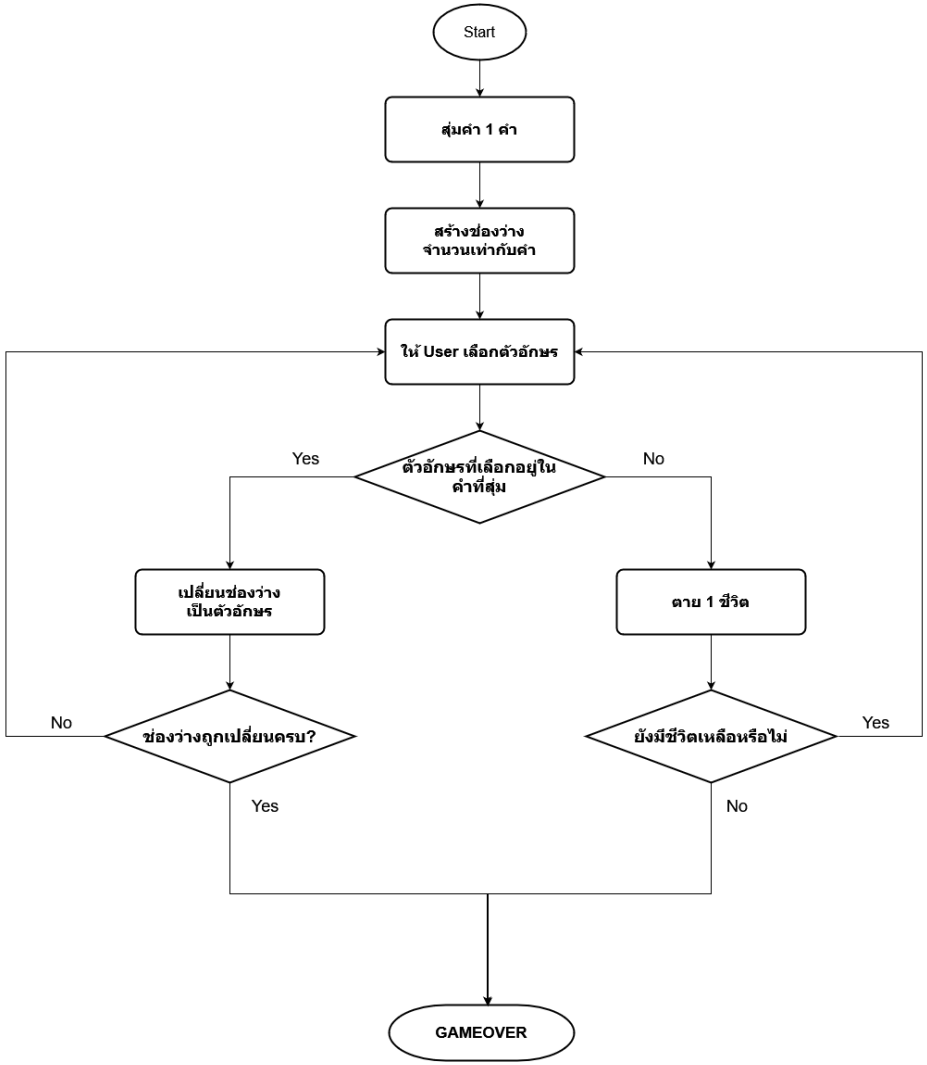
Problem Solving

- https://hangmanwordgame.com/?fca=1&success=0#/



Problem Solving

- หากเราค่อยๆ แตกงานออกไป จะพบว่าจะมีงานดังนี้
  - สุ่มคำมา 1 คำ
  - สร้างช่องว่างเท่ากับคำที่สุ่มมาได้
  - วน Loop
    - ผู้ใช้เลือกตัวอักษร
    - ถ้าตัวอักษรอยู่ในคำ
      - เปลี่ยนช่องว่างเป็นตัวอักษร
      - ถ้าเปลี่ยนครบ -> ชนะ
    - ถ้าไม่อยู่ในคำ
      - ชีวิต -1 ถ้าหมดชีวิต -> แพ้



Problem Solving

- ในการเขียนโปรแกรมให้แยกโปรแกรมเป็นส่วนๆ แล้วค่อยเขียนโปรแกรม
- TODO #1 : กำหนด word_list ต่อไปนี้ ให้สุ่มเลือกมา 1 ตัว แล้วกำหนดค่าลงในตัวแปรชื่อ chosen_word

```
word_list = ["aardvark", "baboon", "camel"]
```

- TODO #2 : สร้าง List ว่าง และตั้งชื่อว่า display และเพิ่ม '_' ใน list display เท่ากับจำนวนตัวอักษรใน chosen_word

เช่น หาก chosen_word เป็น "apple", display จะเป็น ["_", "_", "_", "_", "_"]

Problem Solving

- นำ TODO #1 และ TODO #2 มาเขียนโปรแกรม จะได้ดังนี้

main.py

```
1  import random
2
3  word_list = ["aardvark", "baboon", "camel"]
4  chosen_word = random.choice(word_list)
5
6  display = []
7  word_len = len(chosen_word)
8  for i in range(word_len):
9      display.append('_')
10
11 #Testing code
12 print(f'The solution is {chosen_word}.')
13 print(f'The display is {display}.')
```

Console Shell

```
The solution is camel.
The display is ['_', '_', '_', '_', '_'].
```

Problem Solving

- ส่วนต่อไปจะเป็นการวน Loop ซึ่งเราจะใช้หลักการ “คิดโจทย์แบบง่าย” คือ แทนที่จะวน Loop รับข้อมูล เราแค่รับ 1 ครั้งแล้วตรวจสอบว่ามีตัวอักษรนั้นหรือไม่ก่อน เมื่อทำงานได้แล้วจึงจะเพิ่มให้เป็น Loop ภายหลัง
- TODO #3 : ถามผู้ใช้ให้เดาตัวอักษร 1 ตัว แล้วเก็บในตัวแปรชื่อ guess และแปลงเป็น lowercase
  - การแปลงเป็นตัวเล็กเพื่อไม่ให้มีปัญหาเรื่องตัวเล็กหรือตัวใหญ่
- TODO #4 : ตรวจสอบว่าตัวอักษรที่ผู้ใช้เดา อยู่ในอักขระตัวตัวหนึ่งใน chosen_word หรือไม่

Problem Solving

- นำ TODO #3 และ TODO #4 มาเขียนโปรแกรมเพิ่ม จะได้ดังนี้

main.py

```
1  import random
2
3  word_list = ["aardvark", "baboon", "camel"]
4  chosen_word = random.choice(word_list)
5
6  display = []
7  word_len = len(chosen_word)
8  for i in range(word_len):
9      display.append('_')
10
11 guess = input("Guess a letter: ").lower()
12 for letter in chosen_word:
13     if letter == guess:
14         print("Right")
15     else:
16         print("Wrong")
17
18 print(chosen_word)
```

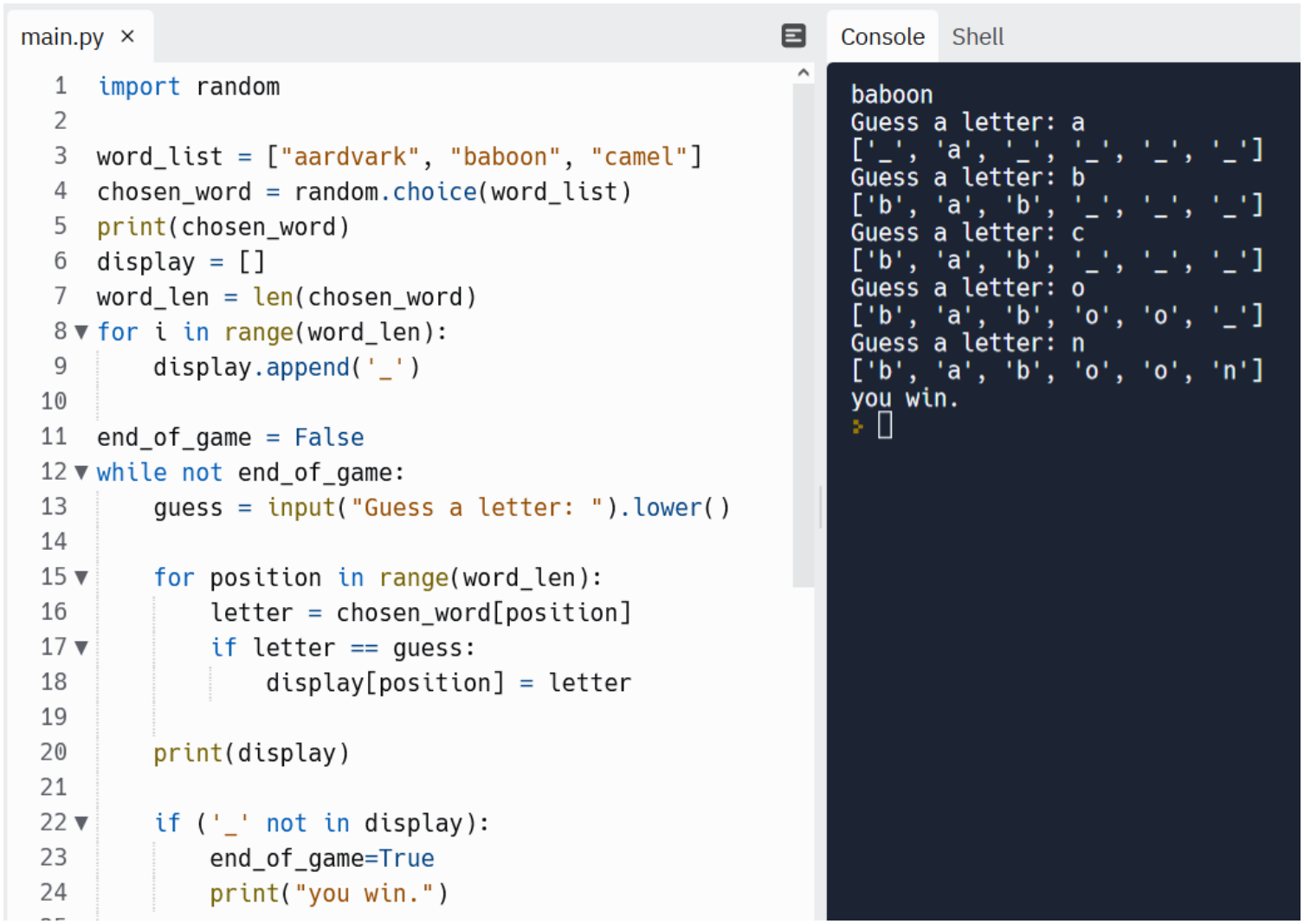
Console Shell

```
Guess a letter: a
Wrong
Right
Wrong
Wrong
Wrong
camel
```

Problem Solving

- เมื่อเรารู้แล้วว่ามีตัวอักษรอะไรที่ป้อนใน chosen_word ก็จะเปลี่ยน display ให้สอดคล้อง เช่น หากป้อนตัว a สำหรับคำว่า apple ก็จะแสดง [“a", "_", "_", "_", "_"]
- TODO #5 : เปลี่ยนค่าใน 'display’ ให้สอดคล้องกับตัวอักษรที่ทายถูก ใน chosen_word และแสดงผล
- TODO #6 : เพิ่ม while loop เพื่อให้ผู้ใช้เดาตัวอักษรไปเรื่อยๆ โดย loop หยุดเมื่อเดาครบทุกตัวและไม่มี "_"

Problem Solving



Problem Solving

- เมื่อเทียบกับ Flowchart จะเห็นว่าเราเขียนโปรแกรมไปเกือบทั้งหมดแล้ว เหลือเพียงส่วนที่ตรวจสอบการทายผิด
- เพื่อให้สนุกขึ้น เราจะสร้างภาพตัวอักษรรูป Hangman เพิ่มเข้าไป

```
stages = ['''
  +---+
  |   |
  O   |
 /|\  |
 / \  |
      |
=========
''', '''
  +---+
  |   |
  O   |
 /|\  |
 /    |
      |
=========
''', '''
  +---+
  |   |
  O   |
 /|   |
      |
      |
=========
''', '''
  +---+
  |   |
  O   |
  |   |
      |
      |
=========
''', '''
  +---+
  |   |
      |
      |
      |
      |
=========
''']
```

Problem Solving

- TODO #7 : สร้างตัวแปร 'lives' เพื่อเก็บจำนวนชีวิต โดยกำหนดให้มีค่าเท่ากับ 6 คือ ทายผิดได้ 6 ครั้ง
- TODO #8: ถ้าอักษรที่เดาไม่ใช่อักษรใน chosen_word ให้ลดค่า 'lives' ลง 1 ถ้า lives เหลือ 0 จบเกมและแสดง "You lose."
- TODO #9: พิมพ์ ASCII art จาก 'stages' ที่สอดคล้องกับ 'lives' ที่เหลืออยู่

Problem Solving

main.py

```
'''

end_of_game = False
word_list = ["aardvark", "baboon", "camel"]
chosen_word = random.choice(word_list)
word_length = len(chosen_word)

lives = 6

print(f'Pssst, the solution is {chosen_word}.')

display = []
for i in range(word_length):
    display.append('_')

while not end_of_game:
    guess = input("Guess a letter: ").lower()

    for position in range(word_length):
        letter = chosen_word[position]

        if letter == guess:
            display[position] = letter

    if guess not in chosen_word:
        lives -= 1
        if lives == 0:
            end_of_game = True
            print("You lose.")

    print(f"{' '.join(display)}")

    if ('_' not in display):
        end_of_game=True
        print("you win.")

    print(stages[lives])
```

Console Shell

```
_ a _ _ _ _

  +---+
  |   |
      |
      |
      |
      |
=========

Guess a letter: b
b a b _ _ _

  +---+
  |   |
      |
      |
      |
      |
=========

Guess a letter: c
b a b _ _ _

  +---+
  |   |
  O   |
      |
      |
      |
=========

Guess a letter:
```

Problem Solving

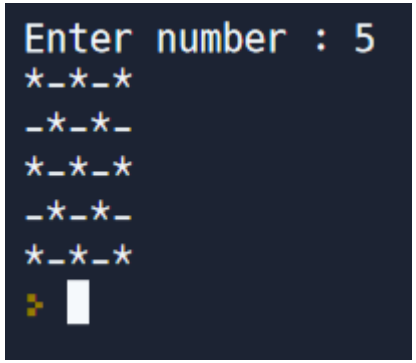
- ถึงจุดนี้เกมก็ถือว่าสมบูรณ์ แต่จะตกแต่งเล็กน้อย เพื่อให้เกมสมบูรณ์มากขึ้น
- TODO-10: - เปลี่ยนจาก word list ในไฟล์เป็น 'word_list' จากไฟล์ hangman_words.py
- TODO-11: - ย้ายรูป hangman_art ไปไว้ในไฟล์
- TODO-12: - Import logo จากไฟล์ hangman_art.py และพิมพ์ในตอนเริ่มเกม
- TODO-13: - ถ้าผู้ใช้ป้อนตัวอักษรที่เคยเดาไปแล้ว แสดงข้อความ
- TODO-14: - ถ้าตัวอักษรไม่ได้อยู่ใน chosen_word พิมพ์อักษร ให้บอก user ว่าผิด

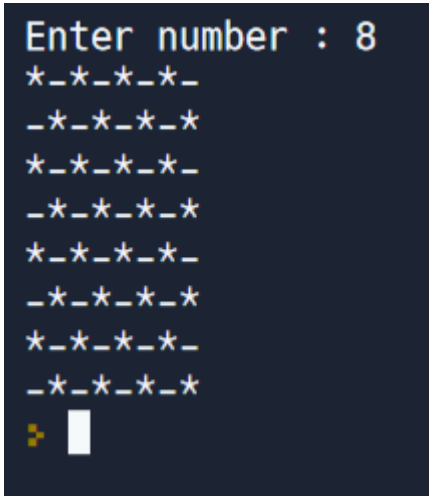
Activity

- ให้ นศ. ระดมสมอง และ เขียน Flowchart ของเกม O-X จากนั้นให้แบ่งการทำงานออกเป็นขั้นตอน

Problem Solving

- โจทย์ตัวอย่างที่ผ่านมา เป็นโจทย์ประเภทที่มีความซับซ้อนมากขึ้น แต่เป็นความซับซ้อนในแง่ของการทำงานหลายขั้นตอน
- แต่จะมีโจทย์อีกประเภทหนึ่ง ที่เป็นการคิดที่ซับซ้อน ซึ่งต้องอาศัยการฝึกฝนในอีก 1 รูปแบบ และบางครั้งอาจต้องใช้ความรู้ทางคณิตศาสตร์ช่วย
- ถ้าให้เขียนโปรแกรมที่รับ 1 Input เป็นตัวเลข โดยให้แสดงผลดังนี้ จะเขียนอย่างไร





Problem Solving

- ใช้วิธีย่อยงานเป็นลำดับ

    1. เริ่มจากพิมพ์ * จำนวนเท่ากับ n
    2. จากนั้นทำตามข้อ 1 เท่ากับ n บรรทัด
    3. ปรับการแสดงผลให้เป็นไปตามที่กำหนด

```
*****
```

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

```
*_*_*
_*_*_
*_*_*
_*_*_
*_*_*
```

Problem Solving

- ในขั้นตอนแรกสามารถเขียนเป็นโค้ดเทียมได้ดังนี้

  รับข้อมูล (n)

  แสดงผล * จำนวน n ครั้ง

- ในข้อที่ 2 สามารถเขียนเป็นโค้ดเทียมได้ดังนี้

  รับข้อมูล (n)

  ทำงานซ้ำ n บรรทัด

  แต่ละบรรทัดแสดงผล * จำนวน n ครั้ง

Problem Solving

- จากข้อ 2 ให้ลองนำไปเขียนเป็นโปรแกรมก่อน เพราะไม่ยากนัก

Problem Solving

- คราวนี้มาดูเรื่องการปรับการแสดงผล เนื่องจากสิ่งที่แสดงจะคล้ายๆ ตาหมากรุก ซึ่งจะต้องมีการเลือกระหว่าง * หรือ – ซึ่งจะต้องมองหาความสัมพันธ์
- สิ่งที่ควรพิจารณา คือ เรามีตัวแปร i และ j หากเราพิมพ์ค่า i กับ j ออกมา เราจะมองเห็นความสัมพันธ์คือ
  - ถ้า i+j เป็นเลขคู่ จะเป็นเครื่องหมาย *
  - ถ้า i+j เป็นเลขคี่ จะเป็นเครื่องหมาย –
- เมื่อเราเห็นความสัมพันธ์เราก็สามารถเขียนโปรแกรมได้โดยไม่ยาก

Problem Solving

- โปรแกรมและผลลัพธ์

main.py

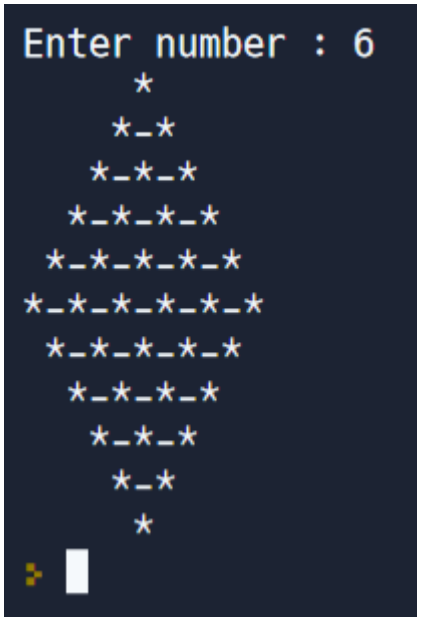
```
1  num = int(input("Enter number : "))
2
3  for i in range(1,num+1):
4      for j in range(1,num+1):
5          if ((i+j)%2==0):
6              print('*',end='')
7          else:
8              print('-',end='')
9      print('')
10
```

Console Shell

```
Enter number : 8
*-*-*-*-
-*-*-*-*
*-*-*-*-
-*-*-*-*
*-*-*-*-
-*-*-*-*
*-*-*-*-
-*-*-*-*
```

Problem Solving

- **Exercise** ลองดูผลลัพธ์ของโปรแกรมดังนี้ ให้นักศึกษาหาแนวทางเขียนโปรแกรมเพื่อรับตัวเลขจำนวน 1 ตัวแล้วแสดงรูปข้าวหลามตัดที่ข้างในเป็นตาหมากรุกตามตัวอย่าง



Problem Solving

- เรายังคงใช้วิธีเดิม คือ การคิดโจทย์ย้อนกลับ

(1)

(2)

(3)

(4)

(5)

- จากนั้นเราก็จะค่อยๆ ปรับแก้ทีละขั้นตอน ซึ่งจะทำให้การเขียนโปรแกรมสามารถเขียนได้ง่ายขึ้น

Problem Solving

- จากขั้นตอนที่ 5 เราเริ่มด้วยโปรแกรมก่อนหน้านี้ ซึ่งสร้างรูปสี่เหลี่ยม

Problem Solving

- จากนั้นทำให้สี่เหลี่ยมกว้างขึ้น 2 เท่า ทำได้โดยการคูณ 2 เข้าไปใน range ในส่วนของ j คือ loop ใน
- อย่างไรก็ตาม หากเราวาดรูปจะพบว่าการจะสร้างเป็นรูปข้าวหลามตัดได้ จำนวนของจุดที่กว้างที่สุดจะต้องเป็นเลขคี่ (มิฉะนั้นส่วนปลายจะไม่ใช่ 1) ดังนั้นจึงต้อง -1

main.py ×

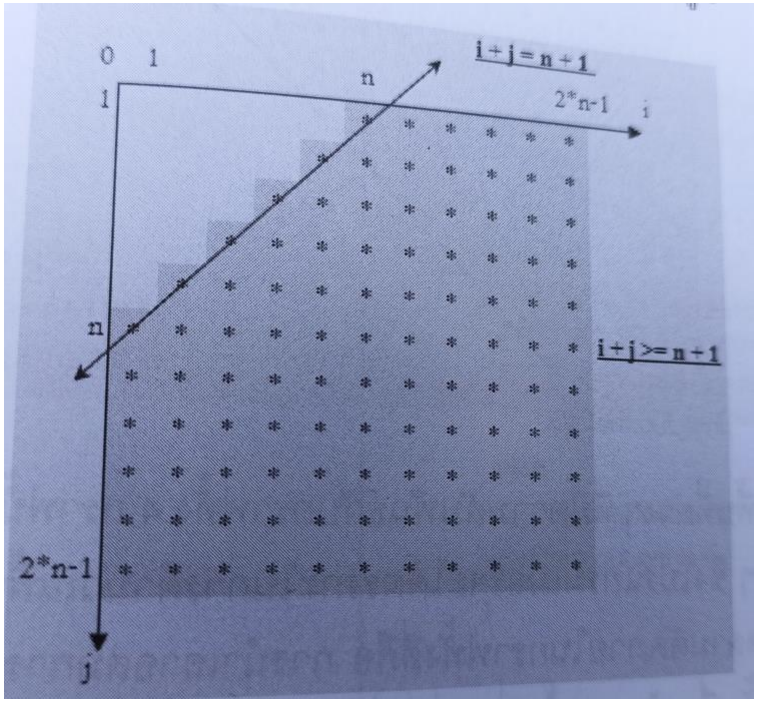
```
1  num = int(input("Enter number : "))
2
3  for i in range(1,num+1):
4      for j in range(1,(2*num)-1):
5              print('*',end='')
6      print('')
7
```

Console Shell

```
Enter number : 5
********
********
********
********
********
```

Problem Solving

- ขั้นตอนต่อไปคือ ทำให้เป็นรูป 3 เหลี่ยม 2 รูปหันหลังชนกัน โดยการเปลี่ยนเครื่องหมาย * บางส่วนให้เป็นช่องว่าง โดยหากอยู่ในพื้นที่ข้าวหลามตัดให้แสดง * แต่หากอยู่นอกพื้นที่ให้แสดงเป็นช่องว่าง
- ลองพิจารณาส่วนของ มุมบนซ้ายก่อน เมื่อเราพิจารณารูป เทียบกับตำแหน่ง coordinate จะพบว่า พื้นที่ส่วนที่จะพิมพ์ช่องว่าง คือ พื้นที่ซึ่ง i+j < n-1



Problem Solving

- จะเห็นว่าสามารถวาดรูปส่วนบนซ้ายได้แล้ว

main.py

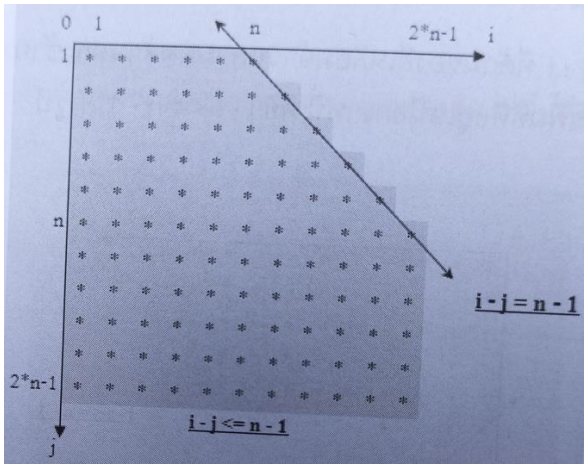
```
1  num = int(input("Enter number : "))
2
3  for i in range(1,num+1):
4      for j in range(1,(2*num)-1):
5          if i+j > num-1:
6              print('*',end='')
7          else:
8              print(' ',end='')
9      print('')
10
```

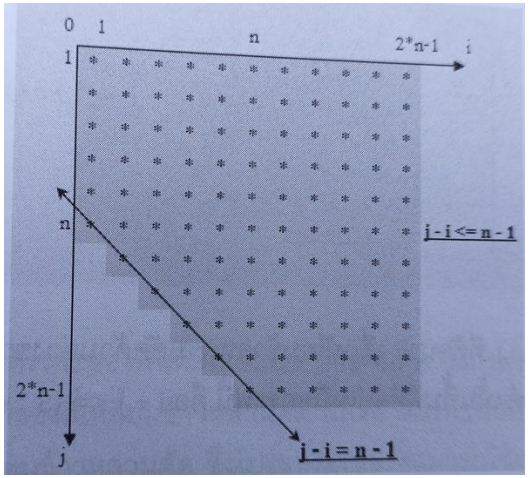
Console Shell

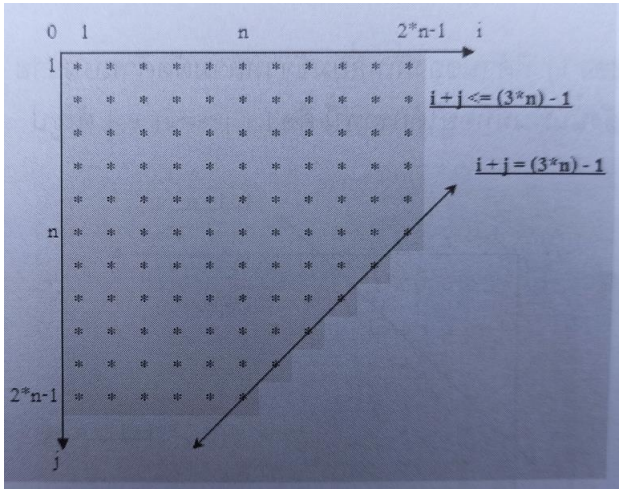
```
Enter number : 5
    *****
   ******
  *******
 ********
 ********
```

Problem Solving

- คราวนี้จะพิจารณาเงื่อนไขส่วนอื่นๆ อีก 3 ด้าน โดยวิธีวาดรูป เราจะเห็นความสัมพันธ์ดังนี้
  - ด้านบนขวาจะเป็นช่องว่างเมื่อ j-I > n-1
  - ด้านล่างซ้ายจะเป็นช่องว่างเมื่อ i-j > n-1
  - ด้านล่างขวาจะเป็นช่องว่างเมื่อ i+j > 3n-1

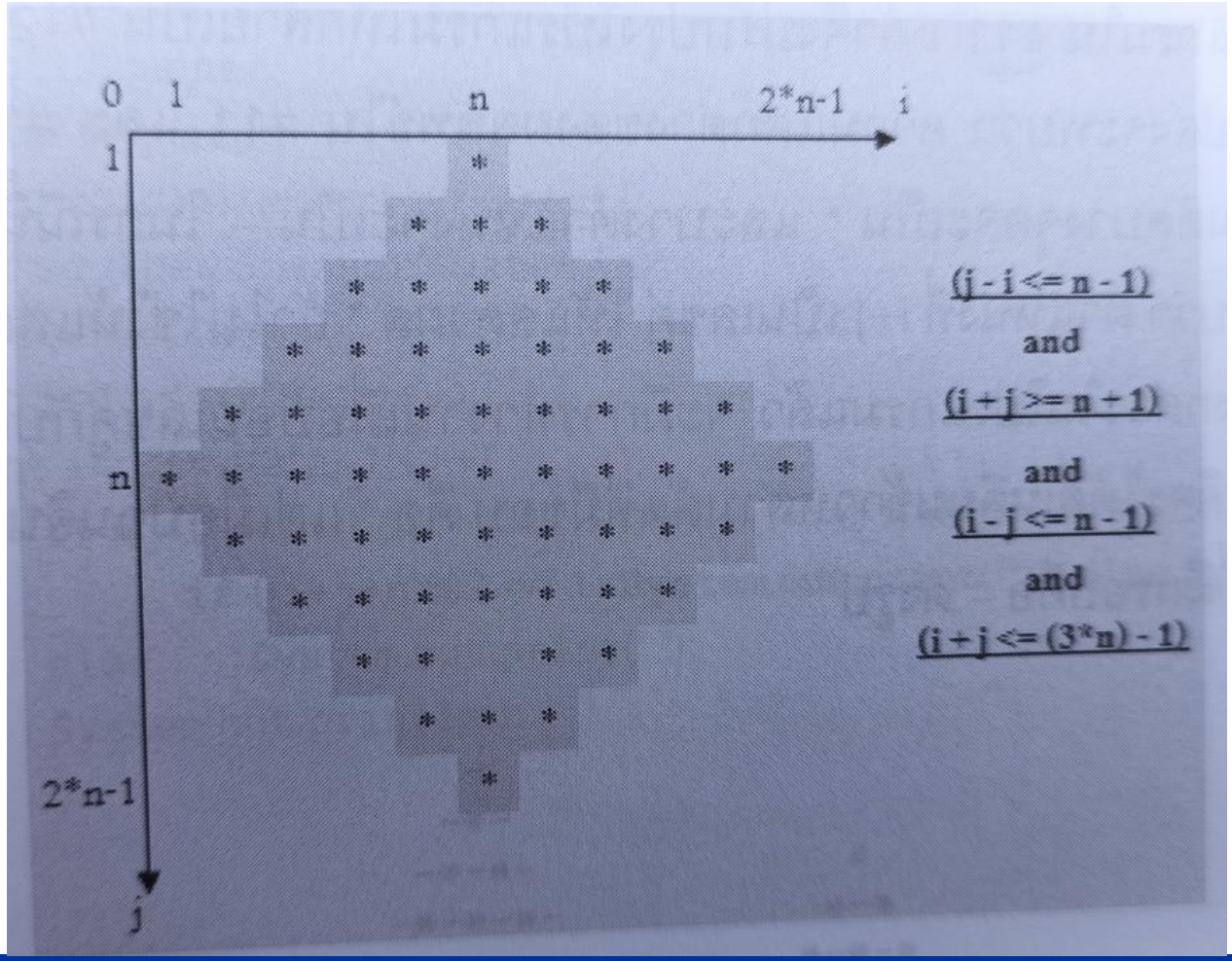




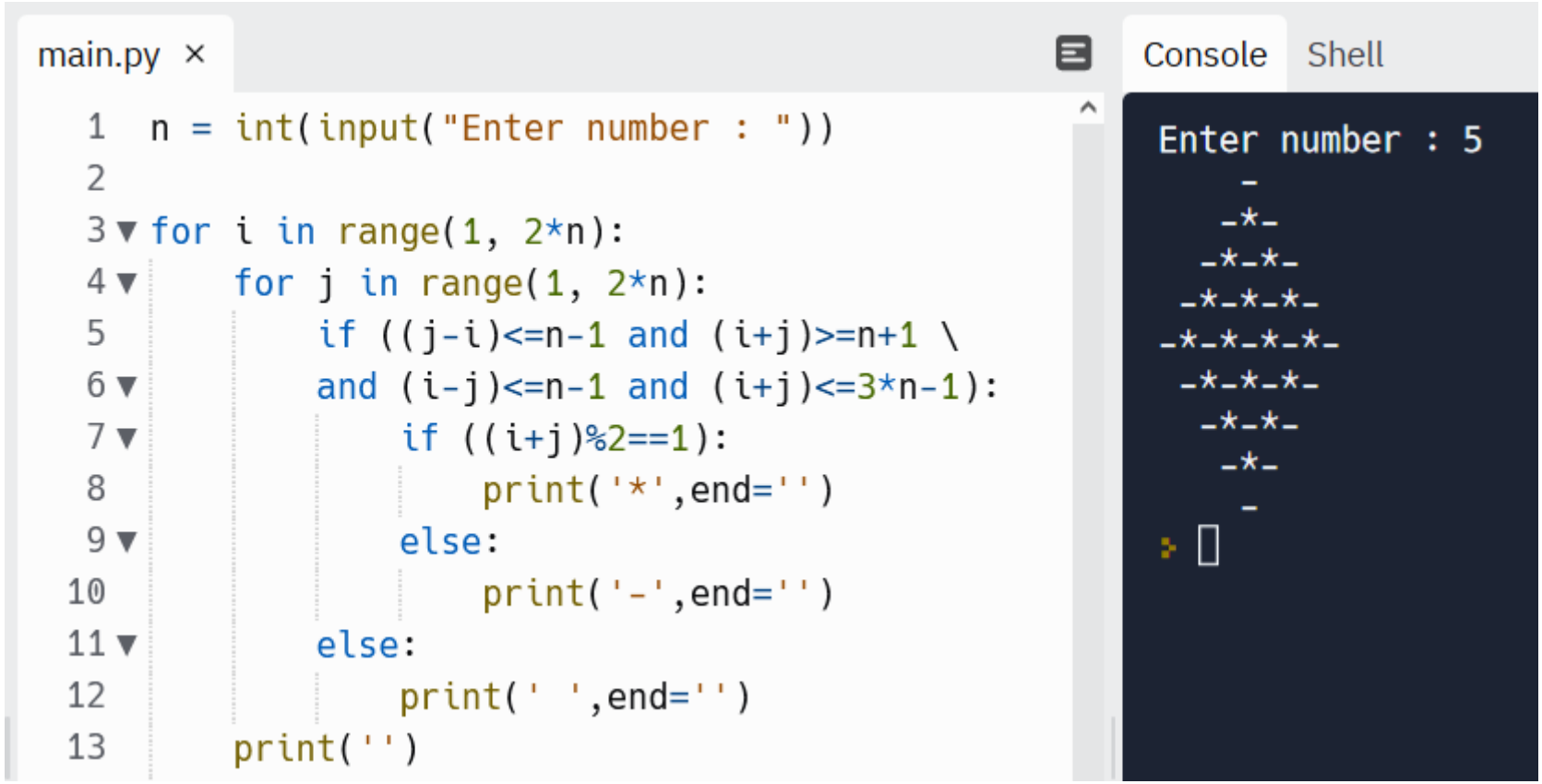


Problem Solving

- เมื่อรวมทุกเงื่อนไข ที่จะพิมพ์รูป * จะมีดังนี้

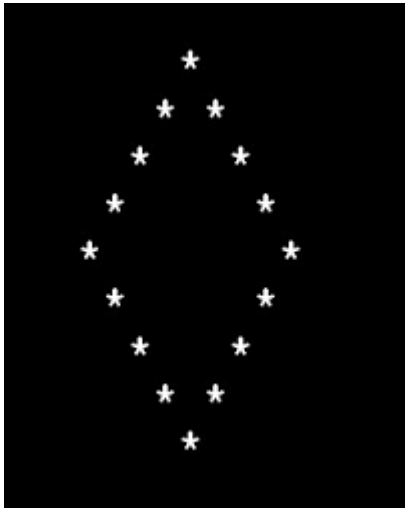


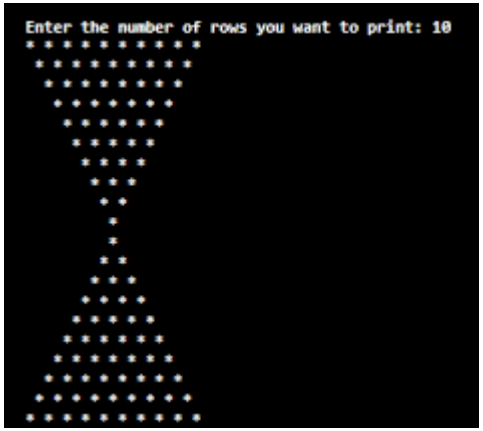
Problem Solving



Problem Solving

- ลองพิจารณาเขียนโปรแกรมเพื่อแสดงรูปต่างๆ





```
0000000
0111110
0122210
0123210
0122210
0111110
0000000

n=3
```

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

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

```
   0
  010
 01210
0123210
 01210
  010
   0

n=3
```

ตัวแปรในหน่วยความจำ

- ตัวแปรที่เก็บในหน่วยความจำจะมีตำแหน่งที่อยู่ ซึ่งสามารถแสดงได้ โดยใช้ id()
- Python จะพยายามลดการใช้พื้นที่ โดยข้อมูลที่มีค่าเดียวกัน จะพยายามเก็บไว้ตำแหน่งเดียวกัน (แต่ไม่เสมอไป)
- ข้อมูลที่อยู่ใน List จริงๆ แล้ว ก็คือตำแหน่งของแต่ละข้อมูล ตามรูป

main.py

```
1  a = 10
2  b = 10
3  c = 20
4  print(id(a))
5  print(id(b))
6  print(id(c))
7  print('')
8  str1 = "abc"
9  str2 = "abc"
10 str3 = "def"
11 print(id(str1))
12 print(id(str2))
13 print(id(str3))
14 print('')
15 lst = [a, b, c]
16 print(id(lst))
17 print(id(lst[0]))
18 print(id(lst[1]))
19 print(id(lst[2]))
```

Console Shell

```
140528103135456
140528103135456
140528103135776

140527406437552
140527406437552
140527405104240

140527404814848
140528103135456
140528103135456
140528103135776
```

ตัวแปรในหน่วยความจำ

- ในบรรทัดที่ 14 เราเปลี่ยนข้อมูลใน a จะเห็นว่าตำแหน่งของ a เปลี่ยนไปอยู่ที่อื่น แต่ใน list ที่เคยกำหนดค่า a ยังใช้ข้อมูลตำแหน่งเดิม จะเห็นได้ว่าการเปลี่ยนค่า a ไม่ได้ทำให้ข้อมูลใน list เปลี่ยนไปด้วย
- ในบรรทัดที่ 21 เราเปลี่ยนข้อมูลใน lst[0] จะเห็นว่า list ไม่ได้เปลี่ยนตำแหน่ง แต่ข้อมูลใน lst[0] มีการเปลี่ยนตำแหน่งไป

main.py

```
1  a = 10
2  b = 10
3  c = 20
4  print(id(a))
5  print(id(b))
6  print(id(c))
7  print('')
8  lst = [a, b, c]
9  print(id(lst))
10 print(lst[0],id(lst[0]))
11 print(lst[1],id(lst[1]))
12 print(lst[2],id(lst[2]))
13 print('')
14 a = 1
15 print(id(a))
16 print(id(lst))
17 print(lst[0],id(lst[0]))
18 print(lst[1],id(lst[1]))
19 print(lst[2],id(lst[2]))
20 print('')
21 lst[0] = 1
22 print(id(lst))
23 print(lst[0],id(lst[0]))
24 print(lst[1],id(lst[1]))
25 print(lst[2],id(lst[2]))
```

Console Shell

```
140464777607392
140464777607392
140464777607712

140464087679488
10 140464777607392
10 140464777607392
20 140464777607712

140464777607104
140464087679488
10 140464777607392
10 140464777607392
20 140464777607712

140464087679488
1 140464777607104
10 140464777607392
20 140464777607712
```

Tuple

- Tuple เป็นโครงสร้างข้อมูลที่คล้ายกับ List แต่ Tuple เป็นแบบ Immutable
- Tuple จะใช้สัญลักษณ์ () ในการกำหนด โดยภายใน Tuple จะเป็นข้อมูลอะไรก็ได้ รวมทั้ง String, Tuple หรือ List

main.py

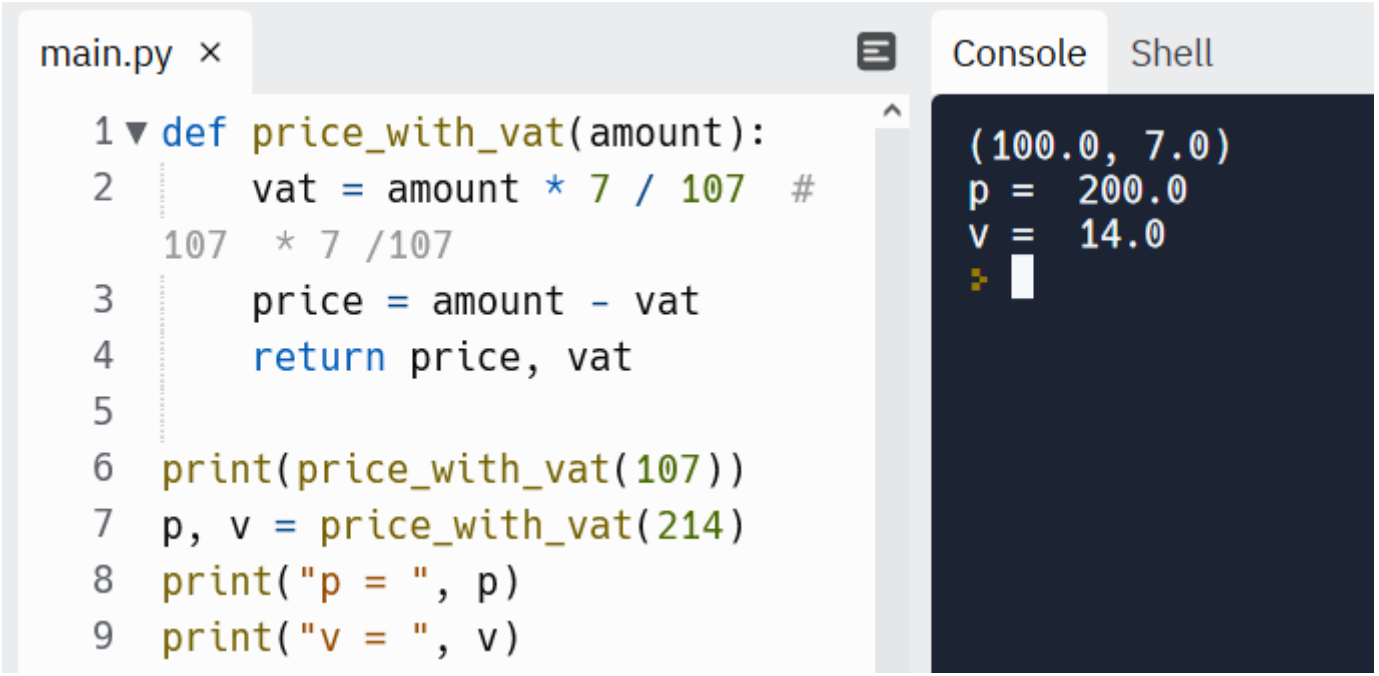
```
# Empty tuple
my_tuple = ()
print(my_tuple)

# Tuple having integers
my_tuple = (1, 2, 3)
print(my_tuple)

# tuple with mixed datatypes
my_tuple = (1, "Hello", 3.4)
print(my_tuple)

# nested tuple
my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
print(my_tuple)
```

Console Shell

```
()
(1, 2, 3)
(1, 'Hello', 3.4)
('mouse', [8, 4, 6], (1, 2, 3))
```

Tuple

- เนื่องจาก Tuple เป็น Immutable จึงมักใช้งานกับข้อมูลที่ไม่มีการเปลี่ยนแปลง เช่น ข้อมูลที่ return จากฟังก์ชัน กรณีที่มีหลายค่า ก็จะ return ออกมาเป็น Tuple



Tuple

- การสร้าง Tuple ไม่จำเป็นต้องใช้ ( ) เสมอไป จริงๆ แล้วสิ่งที่สร้าง Tuple คือเครื่องหมาย , (เรียกว่า Tuple Packing) และยังสามารถ unpack กระจายออกมาที่ตัวแปรอื่นได้ด้วยตามตัวอย่าง

main.py

```
1  my_tuple = 3, 4.6, "dog"
2  print(my_tuple)
3  print(type(my_tuple))
4
5  # tuple unpacking is also possible
6  a, b, c = my_tuple
7
8  print(a)      # 3
9  print(b)      # 4.6
10 print(c)      # dog
```

Console Shell

```
(3, 4.6, 'dog')
<class 'tuple'>
3
4.6
dog
```

Tuple

- ในคำสั่งแรก แม้จะใส่วงเล็บ แต่ก็ไม่เกิดตัวแปรชนิด Tuple แต่ถ้าใส่เครื่องหมาย , ต่อท้าย ก็จะเป็น Tuple เพราะสิ่งที่สร้าง Tuple คือ , เช่นในคำสั่งที่ 3 ไม่มีวงเล็บ แต่ก็เป็นชนิด Tuple

main.py

```
1  my_tuple = ("hello")
2  print(type(my_tuple))  # <class 'str'>
3
4  # Creating a tuple having one element
5  my_tuple = ("hello",)
6  print(type(my_tuple))  # <class 'tuple'>
7
8  # Parentheses is optional
9  my_tuple = "hello",
10 print(type(my_tuple))  # <class 'tuple'>
11
```

Console Shell

```
<class 'str'>
<class 'tuple'>
<class 'tuple'>
>
```

Tuple

- หลายๆ ครั้งที่เราใช้ Tuple โดยไม่ได้ตั้งใจ เช่น เขียนว่า

  a, b, c = 10, 20, 30

  เป็นการสร้าง Tuple ที่มีสมาชิก 10, 20, 30 จากนั้นจึง Unpack ให้กับตัวแปร a, b, c

- จริงๆ แล้วกลไก unpack สามารถใช้งานได้หลากหลาย

  a, b, c = [10, 20, 30]

  a, b, c = ‘XYZ’

  for e in 10, 20, ‘abc’:

  for e in ‘abc’:

  a, b = b, a

Tuple

- สำหรับ การเข้าถึง Tuple จะเหมือนกับ List ตามตัวอย่าง

main.py

```python
# Accessing tuple elements using indexing
my_tuple = ('p','e','r','m','i','t')

print(my_tuple[0])   # 'p'
print(my_tuple[5])   # 't'

# nested tuple
n_tuple = ("mouse", [8, 4, 6], (1, 2, 3))

# nested index
print(n_tuple[0][3])       # 's'
print(n_tuple[1][1])       # 4

# Negative indexing for accessing tuple elements
my_tuple = ('p', 'e', 'r', 'm', 'i', 't')

# Output: 't'
print(my_tuple[-1])

# Output: 'p'
print(my_tuple[-6])
```

Console Shell

```
p
t
s
4
t
p
```

Tuple

- สามารถทำ Slicing ได้

main.py

```
# Accessing tuple elements using slicing
my_tuple = ('e','n','g','i','n','e','e','r')

# elements 2nd to 4th
print(my_tuple[1:4])

# elements beginning to 2nd
print(my_tuple[:-7])

# elements 8th to end
print(my_tuple[7:])

# elements beginning to end
print(my_tuple[:])
```

Console Shell

```
('n', 'g', 'i')
('e',)
('r',)
('e', 'n', 'g', 'i', 'n', 'e', 'e', 'r')
```

Tuple

- การเปลี่ยนแปลงค่าไม่สามารถทำได้ เพราะเป็น Immutable
- **แต่...** การเปลี่ยนแปลงค่าสมาชิกที่เป็น Mutable สามารถเปลี่ยนได้ เพราะที่ไม่เปลี่ยนคือตำแหน่ง (id) ของสมาชิกไม่เปลี่ยน (สำหรับการลบ ใช้ del)

main.py

```
1  # Changing tuple values
2  my_tuple = (4, 2, 3, [6, 5])
3
4  # However, item of mutable element can be changed
5  my_tuple[3][0] = 9    # Output: (4, 2, 3, [9, 5])
6  print(my_tuple)
7
8  # Tuples can be reassigned
9  my_tuple = ('p', 'r', 'o', 'g', 'r', 'a', 'm')
10
11 print(my_tuple)
12
```

Console Shell

```
(4, 2, 3, [9, 5])
('p', 'r', 'o', 'g', 'r', 'a', 'm')
>
```

Tuple

- สำหรับ method ต่างๆ ของ Tuple ก็คล้ายกับ List แต่จะไม่มี method ที่จะไปเปลี่ยนค่าของ Tuple

main.py ×

```
1  my_tuple = ('a', 'p', 'p', 'l', 'e',)
2
3  print(my_tuple.count('p'))  # Output: 2
4  print(my_tuple.index('l'))  # Output: 3
```

Console Shell

```
2
3
```

- สามารถใช้ membership ได้

main.py ×

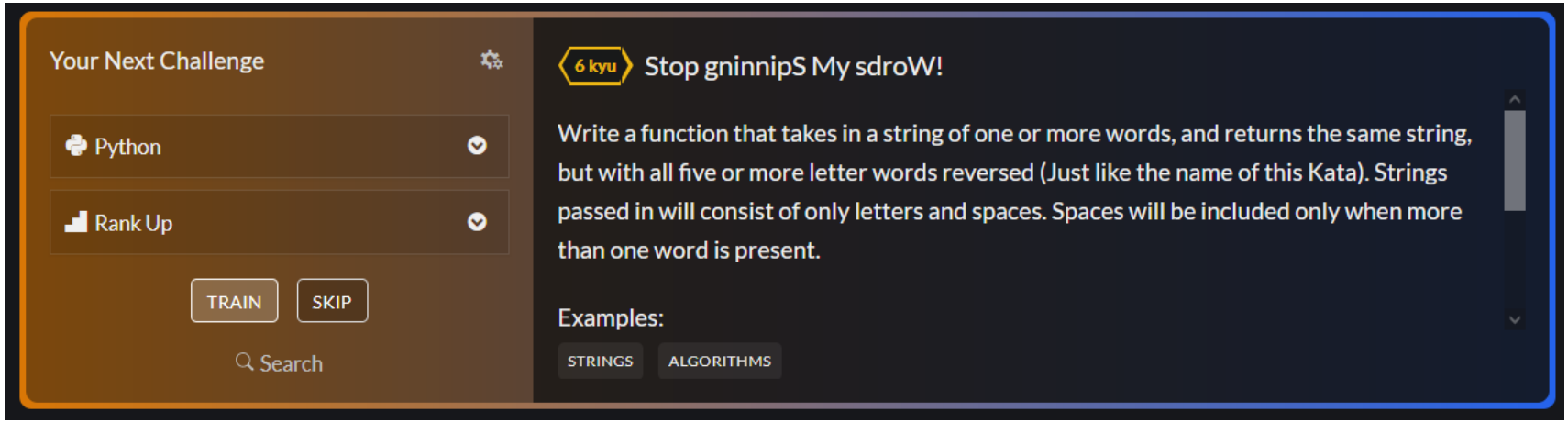
```
1  my_tuple = ('a', 'p', 'p', 'l', 'e',)
2
3  # In operation
4  print('a' in my_tuple)
5  # Not in operation
6  print('g' not in my_tuple)
```

Console Shell

```
True
True
```

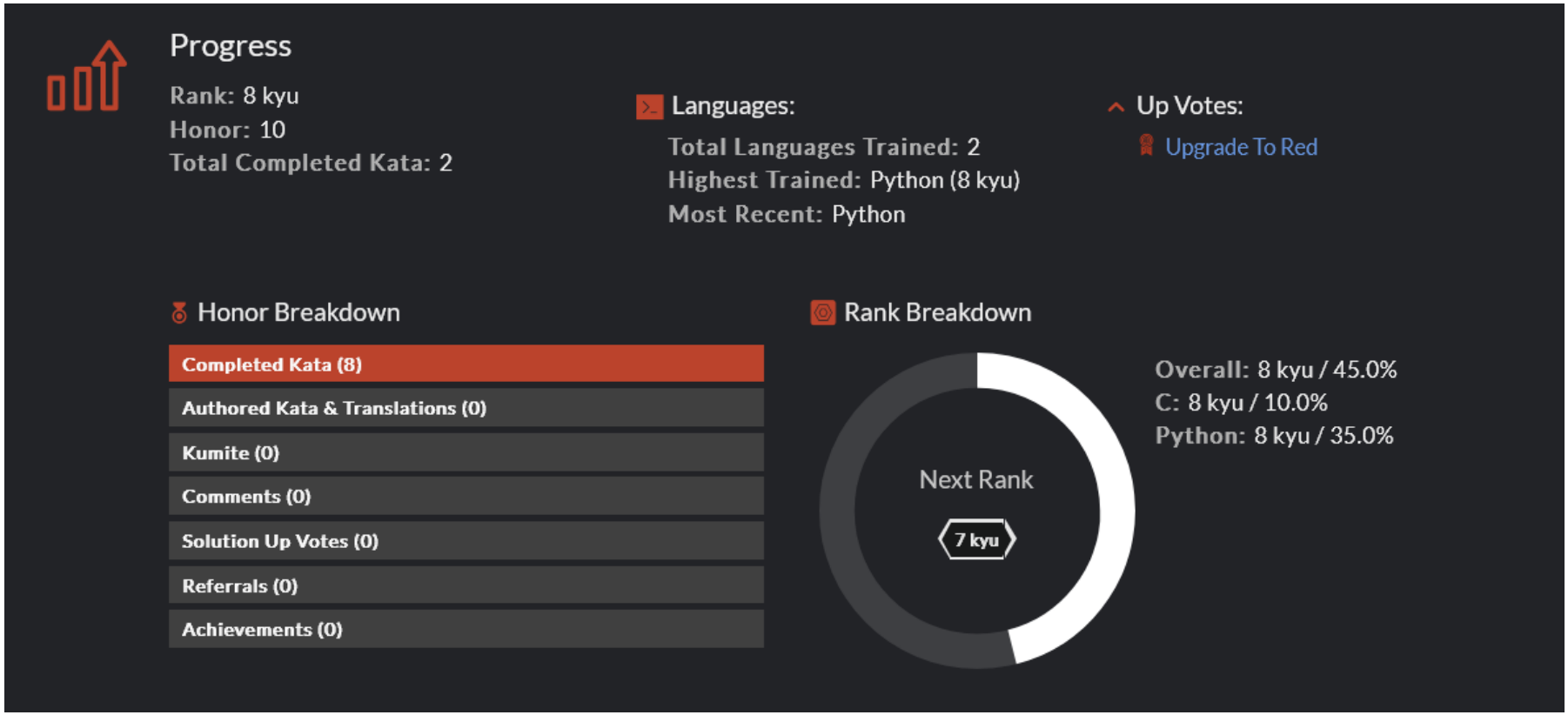
Codewars

- เป็นเว็บไซต์สำหรับฝึกการเขียนโปรแกรม
- เข้าไปที่ https://www.codewars.com/collections/basic-python
- ให้สร้าง Account
- ในการฝึกฝนจะมีให้เลือกหลายแบบ ถ้าใหม่ๆ ให้เลือก fundamental แล้วกด Train



Codewars

- หน้า Profile



For your attention