# บทที่ 6

## การสร้าง Chatbots I

# การแนะนำแชทบอท

- Chatbot เป็นแอปพลิเคชันซอฟต์แวร์สนทนาที่ออกแบบมาเพื่อจำลอง
  บทสนทนาของมนุษย์ที่เกิดขึ้นบนแอปส่งข้อความ
- ซอฟต์แวร์สนทนาไม่ใช่แนวคิดใหม่ บรรทัดคำสั่งแรก
  แอปพลิเคชั่นนี้ถูกสร้างขึ้นในปี พ.ศ. 2503
- บอทสนทนาได้รับความนิยมเพิ่มมากขึ้นอย่างทวีคูณใน
  การตลาด
- เทคโนโลยี Chatbot ใช้ NLP และ AI เพื่อทำความเข้าใจสิ่งที่มนุษย์ต้องการและปรับเปลี่ยนการตอบสนองเพื่อ
  ช่วยให้ผู้ใช้ปลายทางบรรลุผลลัพธ์ที่ต้องการ (เช่น ผู้ช่วยเสมือน)

# Chatbots ทำงานอย่างไร?

- Chatbots ขับเคลื่อนด้วยการตอบสนองที่ตั้งโปรแกรมไว้ล่วงหน้า ปัญญาประดิษฐ์ หรือทั้งสองอย่าง โดยอิงตามกลไกที่ใช้ Chatbots จะประมวลผลคำถามของผู้ใช้เพื่อส่งคำตอบที่ตรงกัน

- Chatbots มี 2 ประเภทหลัก:
  - แชทบอทตามกฎ
  - แชทบอท AI

# แชทบอทตามกฎ

- ให้คำตอบโดยอิงตามกฎ if/then ที่กำหนดไว้ชุดหนึ่ง
  ดำเนินการโดยนักออกแบบแชทบอท
- ให้คำตอบที่ตรงกันเฉพาะเมื่อผู้ใช้ใช้คำสำคัญหรือคำสั่งที่ได้รับการตั้งโปรแกรมให้ตอบ
- อินเทอร์เฟซการสนทนาตามกฎเกณฑ์ไม่สามารถเรียนรู้จากประสบการณ์ในอดีตได้
  พวกเขาตอบสนองตามสิ่งที่พวกเขารู้ในขณะนั้น
- บอทตามกฎเกณฑ์คือระบบที่มีค่าใช้จ่ายถูกที่สุดในการสร้าง
- ตัวอย่าง: “ฉันจะรีเซ็ตรหัสผ่านได้อย่างไร”
  - ขั้นแรกให้มองหาคำหลักในประโยค
  - จากนั้นจับคู่คำสำคัญ ('รีเซ็ต' และ 'รหัสผ่าน') กับคำตอบที่มีอยู่ใน
    ฐานข้อมูลเพื่อให้คำตอบ

# แชทบอท AI

• สามารถสื่อสารกับผู้ใช้ได้อย่างอิสระ

• ต้องได้รับการฝึกฝนอย่างดีและมีการตอบสนองที่กำหนดไว้ล่วงหน้าจึงจะเริ่มต้นได้ • เรียนรู้จากบทสนทนาในอดีต ไม่จำเป็นต้อง

อัปเดตด้วยตนเองในภายหลัง

• การสนทนาที่ดีกว่าแบบตามกฎเกณฑ์เนื่องจากใช้ประโยชน์จาก • การเรียนรู้ของเครื่อง (ML) ช่วยให้บอทสามารถระบุรูปแบบ

ของการป้อนข้อมูลของผู้ใช้ได้
การตัดสินใจและเรียนรู้จากบทสนทนาที่ผ่านมา

• การประมวลผลภาษาธรรมชาติ (NLP) ช่วยให้บอตเข้าใจว่ามนุษย์
สื่อสารเข้าใจบริบทของการสนทนาแม้ว่าผู้สนทนาจะสะกดผิดก็ตาม

• การวิเคราะห์ความรู้สึกช่วยให้แชทบอทเข้าใจอารมณ์ของผู้ใช้

# เนื้อหา

- การนำการสนทนาแบบ Smalltalk มาใช้
- เรียนรู้วิธีใช้นิพจน์ทั่วไปและการเรียนรู้ของเครื่องเพื่อดึงความหมายจากข้อความรูปแบบอิสระ
- สร้างแชทบอทที่สามารถ
  - สอบถามฐานข้อมูล
    - วางแผนการเดินทาง
    - และช่วยคุณสั่งกาแฟ

# เอคโค่บอท 1

ผู้ใช้: สวัสดี!

บอท: ฉันได้ยินคุณ คุณพูดว่า: 'สวัสดี!'

USER: คุณสบายดีไหม?

บอต: ฉันได้ยินคุณนะ คุณพูดว่า: 'คุณสบายดีไหม?'

# เอคโค่บอท II

ผู้ใช้: สวัสดี!

บอท: ฉันได้ยินคุณ คุณพูดว่า: 'สวัสดี!'

```
def ตอบสนอง(ข้อความ):
        กลับ "ฉันได้ยินคุณ! คุณพูดว่า: {}".format(ข้อความ)
def send_message(ข้อความ):
        # เรียก respond() เพื่อรับการตอบสนอง
        พิมพ์(ตอบกลับ(ข้อความ))
ส่งข้อความ("สวัสดี!")
```

- ฟังก์ชัน 'ตอบสนอง': ใช้เพื่อรับข้อความเป็นอาร์กิวเมนต์และ

  ส่งคืนการตอบสนองที่เหมาะสม • ฟังก์ชัน `send_message' ที่

จะพิมพ์สิ่งที่ผู้ใช้เพิ่งพูดไป รับการตอบสนองด้วยการเรียกใช้ฟังก์ชันการตอบสนอง แล้วจึงพิมพ์การตอบสนองของบอต

# เอคโค่บอท III

## เวลา การนำเข้า

## เวลานอน(5)

- มันไม่รู้สึกเป็นธรรมชาติเนื่องจากมีการตอบสนองกลับมาทันที ดังนั้นเราจึงสามารถสร้างความล่าช้าได้โดย การนำเข้าโมดูล `เวลา`
- ในตัวอย่างนี้ เราได้สร้างความล่าช้าครึ่งวินาที

# มาฝึกกันเถอะ!

# เอคโค่บอท 1

- สวัสดีโลก!
- คุณจะเริ่มต้นเรียนรู้วิธีสร้างแชทบอทใน Python โดยการเขียนฟังก์ชันสองอย่างเพื่อสร้างบอทที่เรียบง่ายที่สุดเท่าที่จะเป็นไปได้: EchoBot
  - EchoBot ตอบสนองโดยตอบกลับด้วยข้อความเดียวกับที่ได้รับ
- ในแบบฝึกหัดนี้ คุณจะกำหนดฟังก์ชันที่ตอบสนองต่อผู้ใช้ข้อความ.
- ในการฝึกครั้งต่อไป คุณจะทำ EchoBot ให้เสร็จสมบูรณ์โดยเขียนฟังก์ชันลงไปส่งข้อความไปยังบอท

# เอคโค่บอท 1

- เขียนฟังก์ชันที่เรียกว่า respond() โดยมีข้อความพารามิเตอร์ตัวเดียว
  ซึ่งจะส่งคืนการตอบสนองของบอท เมื่อต้องการทำเช่นนี้ ให้เชื่อมต่อสตริง "ฉันได้ยินคุณ! คุณพูดว่า:
  และข้อความ
- จัดเก็บสตริงที่เชื่อมต่อกันใน bot_message และส่งคืนผลลัพธ์นี้

# เอคโค่บอท 1

ฉันได้ยินคุณนะ คุณพูดว่า: สวัสดี!

```
bot_template = "บอท: {0}"
user_template = "ผู้ใช้: {0}"

# กำหนดฟังก์ชั่นที่ตอบสนองต่อข้อความของผู้ใช้: respond def ____(____):

    # เชื่อมโยงข้อความของผู้ใช้เข้ากับจุดสิ้นสุดของการตอบกลับของบอทมาตรฐาน bot_message = "____" +
                                                    -
    # ส่งกลับผลลัพธ์
    กลับ          -

# ทดสอบฟังก์ชัน
print(respond("hello!"))
```

# เอคโค่บอท II

- เมื่อคุณเขียนฟังก์ชัน respond() แล้ว คุณจะกำหนดฟังก์ชันที่เรียกว่า send_message() โดยมีพารามิเตอร์ข้อความตัวเดียวซึ่งจะบันทึกข้อความและการตอบกลับของบอท

# เอคโค่บอท II

- ใช้เมธอด .format() ของสตริง user_template เพื่อรวมข้อความของผู้ใช้ลงในเทมเพลตผู้ใช้ และพิมพ์ผลลัพธ์
- เรียกใช้ฟังก์ชัน respond() ด้วยข้อความที่ส่งมาและบันทึกผลลัพธ์เป็นการตอบสนอง
- บันทึกการตอบสนองของบอทโดยใช้สตริง bot_template .format() วิธี.
- ส่งข้อความ "สวัสดี" ไปยังบอท

# เอคโค่บอท II

```
USER : สวัสดี

BOT : ฉันได้ยินคุณนะ คุณพูดว่า: สวัสดี
```

```
# สร้างเทมเพลต bot_template
= "BOT : {0}" user_template = "USER : {0}"


# กำหนดฟังก์ชันที่ส่งข้อความไปยังบอท: send_message def ____(____): # พิมพ์ user_template รวมถึง
user_message
        print(____.format(____))

        # รับคำตอบจากบอทต่อข้อความ ตอบกลับ = ____(____)

        # พิมพ์เทมเพลตบอทรวมทั้งการตอบกลับของบอท print(____.format(____))


# ส่งข้อความไปยังบอท send_message("____")
```

# การสร้างบุคลิกภาพ

# ทำไมต้องมีบุคลิกภาพ?

- แชทบอทส่วนใหญ่จะฝังอยู่ในแอปส่งข้อความที่ผู้คนรู้สึกสะดวกใจในการใช้พูดคุยกับเพื่อนๆ

- เพื่อพูดคุยเล็กๆ น้อยๆ กับผู้ใช้ก่อนที่จะลอง "ฟังก์ชัน" ใดๆ ที่พวกเขามาขอ

- ทำให้แชทบอทและผู้ช่วยเสียงเข้าถึงได้ง่ายขึ้นและใช้งานได้สนุกยิ่งขึ้น

- ผู้ใช้จะคาดหวังสิ่งนี้!

พูดคุยเล็กๆ น้อยๆ

```
คำตอบ = { "คุณชื่อ
        อะไร": "ฉันชื่อ EchoBot" "วันนี้อากาศเป็นยังไงบ้าง": "แดดออก!"          -


} def respond(ข้อความ): หาก มี
        ข้อความ อยู่ใน คำตอบ: คืนค่า คำตอบ[ข้อความ]
                respond("คุณชื่ออะไร ?")
```

ชื่อของฉันคือ EchoBot

- โปรดทราบว่าหากไม่มีข้อความที่ตรงกัน คีย์เวิร์ด `return` จะไม่สามารถเข้าถึงได้ ดังนั้น ฟังก์ชันจึงจะส่งคืนค่า None

# รวมถึงตัวแปร

```
คำตอบ = { "วันนี้อากาศ
        เป็นยังไงบ้าง": " วันนี้ {} "

} weather_today = "cloudy" def
respond(message): ถ้า มีข้อความ
        ตอบ กลับ :
                ส่งคืน การตอบกลับ[ข้อความ].format(weather_today) ตอบกลับ(" สภาพอากาศวัน
นี้ เป็นอย่างไร ")
```

วันนี้มีเมฆมาก

# การเลือกคำตอบ

```
คำตอบ = {"คุณชื่ออะไร?": [
        "ฉันชื่อเอคโคบ็อต" "พวกเขาเรียกฉันว่าเอคโคบ็อต" "ฉันชื่อบอต เอคโคบ็อต"
    ],
    -
}
นำเข้า แบบสุ่ม
def ตอบสนอง(ข้อความ):
    ถ้า ข้อความ ใน การตอบสนอง:
        กลับ random.choice(responses[ข้อความ])
ตอบสนอง("คุณชื่ออะไร ?")
```

# การถามคำถาม

```
คำตอบ = ["บอกฉันเพิ่มเติมหน่อยสิ!", "ทำไมคุณถึงคิดแบบนั้น?"]
นำเข้า แบบสุ่ม
def ตอบสนอง(ข้อความ):
        ส่งคืน random.choice(การตอบกลับ)
ตอบกลับ("ฉัน คิดว่าคุณเก่งมากจริงๆ")
```


วิธีที่ดีในการให้ผู้ใช้สนใจคือการถามคำถามหรือเชิญชวนให้พวกเขาอธิบายรายละเอียดเพิ่มเติม

• แทนที่จะใช้ข้อความเริ่มต้น เช่น "ขอโทษ ฉันไม่เข้าใจคุณ" คุณสามารถใช้ประโยคที่เชิญชวนให้สนทนาต่อ ได้ • คำถามเป็นวิธีที่ดีในการบรรลุผลดังกล่าว "ทำไมคุณถึงคิดแบบนั้น" "คุณรู้สึกแบบนี้มานานแค่ ไหนแล้ว" และ "บอกฉันเพิ่มเติมหน่อยสิ!" เป็นคำตอบที่เหมาะสมสำหรับข้อความประเภทต่างๆ

# มาฝึกกันเถอะ!

# พูดคุยจ้อกแจ้

• ใช้พจนานุกรมโดยมีคำถามเป็นคีย์และมีคำตอบที่ถูกต้องเป็นค่า

• ซึ่งหมายความว่าบอทจะตอบสนองอย่างถูกต้องก็ต่อเมื่อข้อความตรงกันทุกประการ ซึ่งเป็นข้อจำกัดที่สำคัญ • กำหนดฟังก์ชัน respond() ที่รับ

อาร์กิวเมนต์ข้อความ ตรวจสอบว่าข้อความมีการตอบสนองที่กำหนดไว้ล่วงหน้าหรือไม่ และส่งคืนการตอบสนองในพจนานุกรมการตอบสนองหากมีการจับคู่ หรือส่งคืนข้อความ "เริ่มต้น" ในกรณีอื่น

# พูดคุยจ้อกแจ้

```
# กำหนดตัวแปร name =
“บอท” weather
= “มีเมฆมาก”
# กำหนดพจนานุกรมด้วยคำตอบที่กำหนดไว้ล่วงหน้า responses = { "คุณชื่ออะไร": "ฉันชื่อ
{0}".format(name),
    "สภาพอากาศวันนี้เป็นอย่างไร": "สภาพอากาศคือ {0}".format(weather), "default": "ข้อความ
    เริ่มต้น"

-
# ส่งคืนการตอบสนองที่ตรงกันหากมี ค่าเริ่มต้นสำหรับกรณีอื่นคือ def ____(____):

    # ตรวจสอบว่าข้อความอยู่ในคำตอบ หรือไม่หาก อยู่ใน ____:
        -
            # ส่งคืนข้อความที่ตรงกัน bot_message = ____[____] มิฉะนั้น:


            # ส่งคืนข้อความ "ค่าเริ่มต้น" bot_message =
            ____["____"] return bot_message
```

บอต: สวัสดี!

USER : คุณชื่ออะไร?

บอท : ฉันชื่อบอท

• เพิ่มฟังก์ชัน send_message() • เพิ่มตัวแปร bot_template • ทดสอบโดยเรียก send_message(“...”)

# เพิ่มความหลากหลาย

• การได้ยินคำตอบเดิมๆ ซ้ำแล้วซ้ำเล่าอาจน่าเบื่อเล็กน้อย • ในแบบฝึกหัดนี้ คุณจะเพิ่มความหลากหลาย •

หากคุณถามบอทของคุณว่ารู้สึกอย่างไร โอกาสที่มันจะตอบกลับ

มาคือ
“โอ้ ฉันยอดเยี่ยมมาก!” หรือ “วันนี้ฉันเศร้ามาก” ควรจะเท่ากัน

• คุณจะใช้โมดูลสุ่ม - โดยเฉพาะ random.choice(ls) - ซึ่ง
เลือกองค์ประกอบแบบสุ่มจากรายการ ls

• มีการกำหนดพจนานุกรมที่เรียกว่าการตอบกลับ ซึ่งจะจับคู่ข้อความแต่ละข้อความกับรายการการตอบ
กลับที่เป็นไปได้ไว้ให้คุณแล้ว

# เพิ่มความหลากหลาย

- นำเข้าโมดูลสุ่ม

# เพิ่มความหลากหลาย

```
# นำเข้าโมดูลสุ่ม _____???_____ ชื่อ = "บอท" สภาพ
อากาศ = "มีเมฆมาก"


# กำหนดพจนานุกรมที่มีรายการคำตอบสำหรับแต่ละข้อความ response = { "คุณชื่ออะไร": [ "ฉันชื่อ {0}".format(name), "พวกเขา
เรียกฉันว่า
    {0}".format(name), "ฉันคือ
        {0}".format(name)


    ],
    "วันนี้อากาศเป็นยังไงบ้าง": [ "อากาศเป็น
        {0}".format(weather), "วันนี้เป็น {0} ".format(weather) ],


    "ค่าเริ่มต้น": ["ข้อความเริ่มต้น"]
-
```

# เพิ่มความหลากหลาย

- หากข้อความอยู่ในคำตอบ ให้ใช้ random.choice() ในฟังก์ชัน respond() เพื่อเลือกคำตอบที่ตรงกันแบบสุ่ม
- ถ้าข้อความไม่ได้อยู่ในการตอบกลับ ให้เลือกการตอบกลับเริ่มต้นแบบสุ่ม

# เพิ่มความหลากหลาย

```
# ใช้ random.choice() เพื่อเลือกการตอบสนองที่ตรงกัน def respond(message): # ตรวจสอบว่าข้อความอยู่ในคำตอบ หรือ
ไม่ หาก มีข้อความ อยู่ ใน คำตอบ: # ส่งคืนการตอบ
        สนองที่ตรงกันแบบสุ่ม bot_message = ____.____(____[____]) มิฉะนั้น:


            # ส่งคืนการตอบสนอง "ค่าเริ่มต้น" แบบสุ่ม bot_message = ____.____(____["____"])
            return bot_message
```

# เพิ่มความหลากหลาย

• การเพิ่มความหลากหลายจะทำให้การพูดคุยกับบอทของคุณสนุกยิ่งขึ้น • ตอนนี้ 'รันโค้ด' และใช้ send_message() (ซึ่งใช้ฟังก์ชัน respond()) เพื่อถามบอทว่า "คุณชื่ออะไร"

```
บอต: สวัสดี!
USER : คุณชื่ออะไร?
บอท : ฉันชื่อบอท
USER : คุณชื่ออะไร?
บอท : ฉันชื่อบอท
USER : คุณชื่ออะไร?
บอท : ฉันคือบอท
```

• ผู้ใช้สามารถพิมพ์และสนทนาต่อได้จนกว่าจะพิมพ์ "ลาก่อน"

# การถามคำถาม

- การถามคำถามเป็นวิธีที่ยอดเยี่ยมในการสร้างบทสนทนาที่น่าสนใจโดยการตอบคำถามด้วยคำถามและตอบคำถามด้วยคำตอบ

- สร้างพจนานุกรมคำตอบโดยมี "คำถาม" และ "ข้อความ" เป็นคีย์ และรายการคำตอบที่เหมาะสมเป็นค่า

# การถามคำถาม

- กำหนดฟังก์ชัน respond() ซึ่งรับข้อความเป็นอาร์กิวเมนต์ และใช้เมธอด .endswith() ของสตริงเพื่อตรวจสอบว่าข้อความลงท้ายด้วยเครื่องหมายคำถามหรือไม่

- หากข้อความลงท้ายด้วยเครื่องหมายคำถาม ให้เลือก "คำถาม" แบบสุ่มจากพจนานุกรมคำตอบ มิฉะนั้น ให้เลือก "ข้อความ" แบบสุ่มจากคำตอบ

```
# กำหนดพจนานุกรมที่มีรายการการตอบกลับสำหรับแต่ละข้อความ

ตอบกลับ = {
        'คำแถลง': [
                'บอกฉันอีกหน่อยสิ!' 'คุณคิด
                แบบนั้นทำไม' 'คุณรู้สึกแบบนี้มานานแค่ไหนแล้ว'
                'ฉันว่ามันน่าสนใจมาก' ' คุณช่วยยืนยันได้ไหม' ' โอ้ ว้าว!' ':)'


        ],
        'คำถาม': [
                "ฉันไม่รู้ :(" 'คุณบอกฉันสิ!'


        -
-
```

# การถามคำถาม

```
นำเข้า การตอบ สนอง
แบบสุ่ม (ข้อความ):
        # ตรวจสอบเครื่องหมายคำถาม ถ้า ____:

                # ส่งคืนคำถามสุ่ม return ____(____["____"])

        # คืนค่าคำสั่งสุ่ม return ____(____["____"])

# ส่งข้อความที่ลงท้ายด้วยเครื่องหมายคำถาม send_message(" วันนี้อากาศเป็นยังไงบ้าง?") send_message(" วันนี้อากาศเป็นยังไงบ้าง?")

# ส่งข้อความที่ไม่ลงท้ายด้วยเครื่องหมายคำถาม send_message("I love building chatbots")
send_message("I love building chatbots")
```

- ผู้ใช้สามารถพิมพ์และสนทนาต่อได้จนกว่าจะพิมพ์ "ลาก่อน"

- สร้างเทมเพลต (bot_template, user_template)

- กำหนดพจนานุกรมที่มีรายการการตอบกลับสำหรับแต่ละข้อความ
- กำหนดฟังก์ชั่นที่ส่งข้อความไปยังบอท: send_message

```
USER : วันนี้สภาพอากาศเป็นอย่างไรบ้าง?
BOT : ฉันไม่รู้ :( USER : วันนี้
สภาพอากาศเป็นอย่างไรบ้าง?
บอท : บอกฉันสิ!
ผู้ใช้ : ฉันชอบสร้างแชทบอท BOT : :)

USER : ฉันชอบสร้างแชทบอท BOT : คุณ
รู้สึกแบบนี้มานานแค่ไหนแล้ว?
```

# การประมวลผลข้อความด้วยนิพจน์ทั่วไป

# นิพจน์ปกติ

• ใช้สำหรับจับคู่ข้อความกับรูปแบบที่รู้จัก • ใช้ในการแยกวลีสำคัญ • ใช้ในการแปลงประโยคตามหลักไวยากรณ์

# การจับคู่รูปแบบ

ผู้ใช้: “คุณยังจำตอนที่คุณกินสตรอเบอร์รี่ในสวนได้ไหม?”

บอท: “ฉันจะลืมได้ยังไงตอนที่กินสตรอเบอร์รี่ในสวน?”

- ความมหัศจรรย์ของระบบนั้นอาศัยการให้ "ความประทับใจ" ว่าบอตเข้าใจคุณ โดยตรรกะพื้นฐานนั้นเรียบง่าย

- เรื่องของตัวอย่างนี้คือ
  - เรากำลังถามถึง “ความทรงจำ” • ความ
    ทรงจำเกี่ยวกับการกินสตรอเบอร์รี่ในสวน
- หากเราแยกรายละเอียดวิธีการสร้างคำตอบออก เราจะเห็นว่ามันง่ายมาก

# การจับคู่รูปแบบ

```
นำเข้า re pattern =

"คุณจำได้ไหม .*" ข้อความ = "คุณจำได้ไหมว่าคุณกินสตรอเบอร์รี่ในสวน"

match = re.search(pattern, message) ถ้า ตรงกัน: print("string matches!")
```

ไม้ขีดไฟสาย!

# การแยกวลีสำคัญ

```
นำเข้า อีกครั้ง
รูปแบบ = "ถ้า (.*)"
ข้อความ = "จะเกิดอะไรขึ้นถ้าบอทยึดครองโลก"
การจับคู่ = การค้นหาใหม่(รูปแบบ, ข้อความ)

การจับคู่กลุ่ม(0)    'ถ้าบอทยึดครองโลก'

การจับคู่กลุ่ม(1)    'บอทยึดครองโลก'
```

• กลุ่มเป็นเพียงสตริงย่อยที่เราสามารถดึงข้อมูลได้หลังจากจับคู่สตริงกับรูปแบบแล้ว • เราใช้เมธอด `group` ของวัตถุการจับคู่เพื่อดึงข้อมูลส่วนของสตริงที่จับคู่แล้ว • ดัชนี 0 คือสตริงทั้งหมด • ดัชนี 1 คือกลุ่มที่เราได้กำหนดโดยรวมวงเล็บ (...) ไว้ในรูปแบบ

# การแปลงทางไวยากรณ์

```
นำเข้า re def

swap_pronouns(วลี): ถ้า 'ฉัน' อยู่ใน วลี: กลับ re.sub('ฉัน',

        'คุณ', วลี) ถ้า 'ของฉัน' อยู่ใน วลี: กลับ

                re.sub('ของฉัน', 'ของคุณ', วลี) มิฉะนั้น: กลับวลี

พิมพ์(swap_pronouns("ฉัน เดินไปโรงเรียน"))
```

```
คุณเดินไปโรงเรียน
```

# มาฝึกกันเถอะ!

# การแยกวลีสำคัญ

• วิธีที่โปรแกรมดูเหมือนจะเข้าใจสิ่งที่คุณบอก • ในแบบฝึกหัดนี้ คุณจะจับคู่ข้อความกับรูปแบบทั่วไปและแยกวลีโดยใช้ re.search() • สร้างพจนานุกรมที่เรียกว่ากฎ ซึ่งจะจับคู่รูปแบบต่อไปนี้:

• "คุณคิดว่า (.*)" • "คุณจำ (.*)
ได้ไหม" • "ฉันต้องการ (.*)" • "ถ้า (.*)“

```
กฎ = { 'คุณ
        คิดว่า (.*)': [
                'if { 0 }? แน่นอน', 'ไม่มีทาง'], 'คุณจำ
                (.*) ได้ไหม': [ 'คุณคิดว่า
        ฉันจะลืม { 0 } ' ทำไมคุณถึงลืม { 0 } ไม่ได้ ",
                'แล้ว { 0 } ล่ะ ', 'ใช่ .. แล้ว?'], 'ฉันต้องการ (.*) ': [ 'มันจะหมายความ
                ว่าอย่างไรหากคุณมี { 0 } ', 'ทำไมคุณถึงต้องการ { 0 } ', "อะไรที่ทำให้คุณไม่ได้รับ
                { 0 } "], 'if (.*) ': [ "คุณคิด
                จริงๆ หรือว่า { 0 } ', 'คุณ
        ต้องการให้ { 0 } ', 'คุณคิด
                อย่างไรเกี่ยวกับ { 0 } ', 'จริงๆ แล้ว--ถ้า { 0 } ' ]
```

-

# การแยกวลีสำคัญ

• ทำซ้ำในพจนานุกรมกฎโดยใช้เมธอด .items() พร้อมด้วยรูปแบบ
และการตอบกลับเป็นตัวแปรตัววนซ้ำของคุณ

• ใช้ re.search() กับรูปแบบและข้อความเพื่อสร้างอ็อบเจ็กต์การจับคู่ • หากมีการจับคู่ ให้ใช้ random.choice()

เพื่อเลือกคำตอบ • หากมี '{0}' อยู่ในคำตอบนั้น ให้ใช้เมธอด .group() ของอ็อบเจ็กต์การ

จับคู่ด้วย
ดัชนี 1 เพื่อดึงข้อมูลวลี

```
# กำหนด match_rule()
def match_rule(กฎ, ข้อความ):
    ตอบกลับ วลี = "ค่าเริ่มต้น" ไม่มี
    # ทำซ้ำในพจนานุกรมกฎ
    สำหรับใน ___:, _____
        # สร้างวัตถุที่ตรงกัน
        match =  -
        ถ้า match ไม่ใช่ None:
            # เลือกคำตอบแบบสุ่ม
            การตอบสนอง =  -
            ถ้า '{0}' ใน การตอบสนอง:
                วลี = #  -
    คืนคำตอบและวลี
    ส่งคืน รูปแบบการตอบสนอง(วลี)
# ทดสอบกฎการจับคู่
พิมพ์(กฎการจับคู่(กฎ, "คุณจำวันเกิดครั้งสุดท้ายของคุณได้ไหม"))
```

- ผู้ใช้สามารถพิมพ์และสนทนาต่อได้จนกว่าจะพิมพ์ "ลาก่อน"

# คำสรรพนาม

- แปลงวลีที่แยกออกมาจากบุคคลที่หนึ่งเป็นบุคคลที่สองและรอง
  ในทางกลับกัน
- ในภาษาอังกฤษ การผันกริยาเป็นเพียงการสลับคำ เช่น "me" กับ "you", "my" กับ "your"
- ในแบบฝึกหัดนี้ คุณจะกำหนดฟังก์ชันที่เรียกว่า replace_pronouns() ซึ่งใช้ re.sub() เพื่อแมป "ฉัน" และ "ของฉัน" เป็น "คุณ" และ "ของคุณ" (และในทางกลับกัน) ในสตริง

# คำสรรพนาม

• หากข้อความมีคำว่า "me" ให้ใช้ re.sub() เพื่อแทนที่ด้วย "you" • หากข้อความมีคำ
ว่า "my" ให้แทนที่ด้วย "your" • หากข้อความมีคำว่า "your" ให้
แทนที่ด้วย "my" • หากข้อความมีคำว่า "you" ให้แทนที่ด้วย
"me"

```
# กำหนด replace_pronouns() def replace_pronouns(message):
message = message.lower() ถ้า 'me' อยู่ใน ข้อความ: # แทนที่ 'me' ด้วย
        'you' return ถ้า 'my' อยู่ใน ข้อความ: # แทนที่ 'my' ด้วย 'your'
        return
                        -
                        -
        หากเป็น 'ของคุณ' ใน ข้อความ:
                # แทนที่ 'your' ด้วย 'my' กลับถ้า มี 'you' ใน ข้อความ:
                        -
                # เปลี่ยน 'คุณ' เป็น 'ฉัน' กลับ
                        -
        ส่ง ข้อความ กลับ

พิมพ์(replace_pronouns(" วันเกิดปีที่แล้วของฉัน"))
พิมพ์(replace_pronouns("ไป กับฉันที่ฟลอริดา")) พิมพ์(replace_pronouns("ฉัน มี
ปราสาทของตัวเอง"))
```

```
วันเกิดปีที่แล้วของคุณ ไป
ฟลอริดากับคุณ ฉันมีปราสาท
เป็นของตัวเอง
```

# การนำทุกอย่างมารวมกัน

- นำทุกอย่างจากแบบฝึกหัดก่อนหน้ามารวมกัน • สร้างฟังก์ชัน

match_rule(), send_message(), replace_pronouns(), เทมเพลต และพจนานุกรม กฎ

# การนำทุกอย่างมารวมกัน

- รับการตอบสนองและวลีโดยเรียก match_rule() ด้วยกฎ พจนานุกรมและข้อความ
- ตรวจสอบว่าการตอบสนองเป็นเทมเพลตหรือไม่โดยดูว่ามีสตริงรวมอยู่ด้วยหรือไม่ '{0}'. ถ้าเป็นเช่นนั้น:
  - ใช้ฟังก์ชัน replace_pronouns() กับวลี
  - รวมวลีโดยใช้ .format() ในการตอบสนองและเขียนทับค่าของการตอบสนอง

```
# กำหนดคำตอบ() def คำ
ตอบ(ข้อความ):
        # เรียก match_rule response
        = วลี =                 -
                        -

        ถ้า '{0}' ตอบ สนอง :
                # แทนที่คำสรรพนามในวลี วลี =
                                -
                # ใส่ประโยคในคำตอบ response = return response
                                        -
```

```
# ส่งข้อความ send_message("
คุณจำวันเกิดปีที่แล้วของฉันได้ไหม") send_message(" คุณคิดว่ามนุษย์ควรจะกังวลเกี่ยวกับ
AI หรือไม่") send_message("ฉัน ต้องการเพื่อนที่เป็นหุ่นยนต์") send_message("จะเป็นอย่างไร หากคุณสามารถเป็น
อะไรก็ได้ที่คุณต้องการ")
```

USER : คุณจำวันเกิดครั้งสุดท้ายของฉันได้ไหม

บอต : คุณคิดว่าฉันจะลืมวันเกิดปีที่แล้วของคุณเหรอ

# คำถาม

อ้างอิง: https://

campus.datacamp.com/

https://www.chatbot.com/blog/chatbot-guide/