บทที่ 6

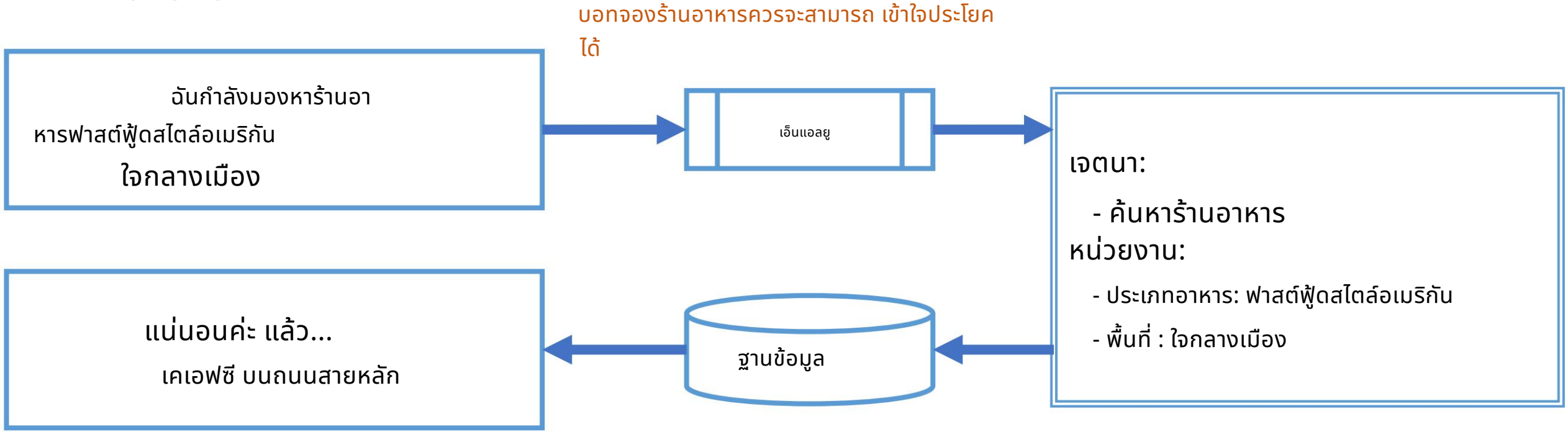
การสร้างแชทบอท II

การทำความเข้าใจเจตนาและตัวตน

หัวข้อนี้เป็นเรื่องเกี่ยวกับความเข้าใจภาษาธรรมชาติ (NLU)

NLU เป็นสาขาย่อยของ NLP ซึ่งมักเกี่ยวข้องกับการแปลงข้อความรูปแบบอิสระเป็นข้อมูลที่มีโครงสร้างภายในโดเมน
เฉพาะ

ตัวอย่าง



ในการดำเนินการนี้ เราจำเป็นต้องระบุ เจตนา ของข้อความ และแยกชุดของ เอนทิตี ที่เกี่ยวข้องออกมา

เจตนา

- เจตนาคือคำอธิบายกว้าง ๆ เกี่ยวกับสิ่งที่บุคคลกำลังพยายามจะพูด
    - ตัวอย่างเช่น "สวัสดี" "หวัดดี" คือวิธีที่ผู้คนอาจ "ทักทาย" บอทของคุณ

- มีหลายวิธีที่ต่างกันในการแสดงเจตนาที่อธิบายด้วย `restaurant_search`

    - ฉันหิว • แนะนำ
    ร้านพิซซ่าที่อร่อยให้ฉันหน่อย • ฉันอยาก
    พาเพื่อนไปกินซูชิ

เจตนา

- ไม่มีวิธีที่ถูกต้องในการกำหนดเจตนาให้กับประโยค คำตอบที่ "ถูกต้อง" ขึ้นอยู่กับแอปพลิเคชันของคุณ ตัวอย่างเช่น • หากคุณขยายความสามารถของบอทเพื่อให้สามารถจองโต๊ะให้

  คุณได้ ประโยค "ฉันอยากพาเพื่อนไปกินซูชิ" อาจอธิบายได้ดีกว่าว่าเป็นเจตนา `request_booking` มากกว่า `restaurant_search`

หน่วยงาน

• ส่วนที่สองของ NLU คือการแยก 'เอนทิตี' ออกจากข้อความ

“จองโต๊ะสำหรับ วันที่ 10 มิถุนายน ที่ร้านซูชิในนิวยอร์กซิตี้”

• ในตัวอย่างการค้นหาร้านอาหาร หมายความว่าต้องระบุ '10 มิถุนายน ' เป็นวันที่ 'ซูชิ'
เป็นประเภทอาหาร และ 'นิวยอร์ก
ซิตี้' เป็นสถานที่ • ปัญหาที่ได้รับการศึกษา
อย่างดีใน NLP คือ "NER" ซึ่งเกือบจะเหมือน
กันทุกประการ
ปัญหา ความแตกต่างที่ • NER มักมุ่งหวัง
ที่จะค้นหาสิ่งที่เป็น 'สากล' เช่น ชื่อบุคคล องค์กร วันที่ ฯลฯ

• บอตมักต้องการคำจำกัดความที่ "แคบกว่า" สำหรับเอนทิตีที่เฉพาะเจาะจงกับโดเมนของพวกเขา

การใช้นิพจน์ทั่วไป

- ใช้ regex เพื่อค้นหาคำหลักในข้อความ
- สร้างนิพจน์ที่ตรงกับคำสำคัญใดคำหนึ่งโดยใช้
  - ตัวดำเนินการท่อ '|' และ • เพิ่มนิพจน์

  ขอบเขตคำ "\b" เพื่อไม่ให้มีอักขระตัวอักษรและตัวเลขอยู่ทั้งสองด้านของคำหลักของเรา

re.search(r"\b(hello|hey|hi)\b", "เฮ้!") ไม่ใช่ None — จริง

re.search(r"\b(hello|hey|hi)\b", "อันไหน?") ไม่ใช่ None — เท็จ

การใช้ regex สำหรับการจดจำเอนทิตี้

```
pattern = re.compile('[AZ]{1}[az]*')
ข้อความ = """แมรี่เป็นเพื่อนของฉัน เธอเรียนที่อ็อกซ์ฟอร์ดและตอนนี้ทำงานที่ Google"""

pattern.findall(ข้อความ)
```

```
['แมรี่', 'ออกซ์ฟอร์ด', 'กูเกิล']
```

- สร้างวัตถุรูปแบบโดยใช้เมธอด `re.compile`
- รูปแบบนี้ตรงกับตัวอักษรพิมพ์ใหญ่ 1 ตัวและ 0 หรือมากกว่า ตัวอักษรพิมพ์เล็ก

มาฝึกกันเถอะ!

การจำแนกประเภทความตั้งใจด้วย regex

- เริ่มต้นด้วยการใช้เทคนิคง่ายๆ ในการจดจำเจตนา - มองหาการมีอยู่ของคำหลัก

- สร้างพจนานุกรม 'คำหลัก' โดยมี • เจตนา "ทักทาย" "ลาก่อน"
  และ "ขอบคุณ" เป็นคีย์ และมี • รายการคำหลักเป็นค่าที่สอดคล้องกัน • ตัวอย่างเช่น คำหลัก
  ["ทักทาย"] ถูกตั้งค่าเป็น "["สวัสดี""]

- สร้างพจนานุกรมที่สอง คำตอบที่ระบุว่าบอตควรตอบสนองต่อเจตนาแต่ละประการอย่างไร นอกจากนี้ยังมีคำตอบเริ่มต้น
  โดยใช้คีย์ "default"

- สร้างฟังก์ชัน send_message() พร้อมกับเทมเพลตบอทและผู้ใช้

การจำแนกประเภทความตั้งใจด้วย regex

- สร้างพจนานุกรม 'คำหลัก'
- ทำซ้ำในพจนานุกรมคำหลักโดยใช้เจตนาและคีย์เป็นของคุณ
  ตัวแปรแบบวนซ้ำ
- ใช้ '|'.join(keys) เพื่อสร้างนิพจน์ทั่วไปเพื่อจับคู่กับคีย์เวิร์ดอย่างน้อยหนึ่งคำและส่งไปยัง re.compile()
  เพื่อคอมไพล์นิพจน์ทั่วไปเป็นอ็อบเจ็กต์รูปแบบ จัดเก็บผลลัพธ์เป็นค่าของพจนานุกรมรูปแบบ

```
# กำหนด 'คำหลัก' ในพจนานุกรม
คำหลัก = {'greet': ['hello', 'hi', 'hey'], 'goodbye': ['bye', 'farewell'], 'thankyou':
 ['thank', 'thx']}
# กำหนดพจนานุกรมของรูปแบบ
รูปแบบ = {}
# ทำซ้ำในพจนานุกรมคำสำคัญ
สำหรับใน ____:, _____
        # สร้างนิพจน์ทั่วไปและคอมไพล์เป็นวัตถุรูปแบบ
        patterns[intent] = #         -
พิมพ์รูปแบบ
พิมพ์(รูปแบบ)
```

```
{'greet': re.compile('hello|hi|hey'), 'goodbye': re.compile('bye|farewell'), 'thankyou':
re.compile('thank|thx')}
```

การจำแนกประเภทความตั้งใจด้วย regex

- กำหนดฟังก์ชันเพื่อค้นหาจุดประสงค์ของข้อความ
- ทำซ้ำตามเจตนาและรูปแบบในพจนานุกรมรูปแบบโดยใช้เมธอด .items()

- ใช้รูปแบบการ .search() เพื่อค้นหาคำหลักในข้อความ
- หากมีการตรงกัน ให้ส่งคืนเจตนาที่สอดคล้อง
- เรียกใช้ฟังก์ชัน match_intent() ของคุณภายใน response() โดยมีข้อความเป็น การโต้แย้ง.

```
# กำหนดฟังก์ชันเพื่อค้นหาเจตนาของข้อความ def match_intent(message): matched_intent
= None สำหรับ เจตนา รูปแบบ ใน ____:
```

```
คำตอบ = {'greet': 'สวัสดีคุณ! :)', 'goodbye': 'ลาก่อนนะ',
'thankyou': 'ยินดีมาก', 'default': 'ข้อความเริ่มต้น'}
```

```
# ตรวจสอบว่ารูปแบบเกิดขึ้นในข้อความหรือ ไม่ ถ้า ____: matched_intent = return
```

```
matched_intent -
```

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

```
# กำหนดฟังก์ชันตอบสนอง def
respond(ข้อความ):
    # เรียกใช้ฟังก์ชัน match_intent intent =
                -
    # ย้อนกลับไปที่ คีย์ การตอบสนองเริ่มต้น = "ค่าเริ่มต้น" ถ้า มีเจตนา
    ใน การตอบสนอง: คีย์ =
    เจตนา ส่งคืน responses[key]
```

```
# กำหนดฟังก์ชันที่ส่งข้อความไปยังบอท: send_message def send_message(message):
```

```
    # พิมพ์ user_template รวมถึง user_message print(user_template.format(message))

    # รับคำตอบจากบอทต่อข้อความ response = respond(message)

    # พิมพ์เทมเพลตบอทรวมทั้งการตอบสนองของบอท print(bot_template.format(response))
```

```
# ส่งข้อความ
send_message("hello!")
send_message("bye byeee")
send_message("ขอบคุณ มาก!")
```

```
USER : สวัสดี!
BOT : สวัสดีคุณ! :)
USER : บ๊ายบาย BOT :
ลาก่อนนะครับ USER : ขอบคุณ
มาก!
BOT : ยินดีมากเลยครับ
```

การสกัดเอนทิตี้ด้วย regex

- ใช้อีกวิธีง่ายๆ ในการค้นหาชื่อบุคคลในประโยค เช่น "สวัสดี ฉันชื่อเดวิด คอปเปอร์ฟิลด์"

- ค้นหาคำหลัก "ชื่อ" หรือ "ชื่อ" และค้นหาคำที่เป็นตัวพิมพ์ใหญ่
  ใช้ regex และถือว่านั่นคือชื่อ
- แบบฝึกหัดนี้คือการกำหนดฟังก์ชัน find_name() เพื่อดำเนินการนี้

การสกัดเอนทิตี้ด้วย regex

• ใช้ re.compile() เพื่อสร้างรูปแบบในการตรวจสอบว่ามีคำสำคัญ "ชื่อ" หรือ "การเรียก" เกิดขึ้นหรือไม่

• สร้างรูปแบบในการค้นหาคำที่มีตัวพิมพ์ใหญ่ • ใช้เมธอด .findall() ใน name_pattern เพื่อดึงคำที่ตรงกันทั้งหมดในข้อความ

• เรียกใช้ฟังก์ชัน find_name() ของคุณภายใน response()

```
# กำหนด find_name()
def find_name(ข้อความ):
    ชื่อ = ไม่มี
    # สร้างรูปแบบการตรวจสอบว่ามีคำสำคัญเกิดขึ้นหรือไม่
    name_keyword = #        -
    สร้างรูปแบบสำหรับการค้นหาคำที่มีตัวพิมพ์ใหญ่
    name_pattern = if       -
    name_keyword.search(ข้อความ):
        # รับคำที่ตรงกันในสตริง
        name_words =    -
        ถ้า len(name_words) > 0:
            # คืนชื่อหากมีคำสำคัญอยู่
            ชื่อ =          ' '.เข้าร่วม(คำชื่อ)
    ชื่อ กลับ
```

```
# กำหนดคำตอบ() def คำ
ตอบ(ข้อความ):
    # ค้นหาชื่อ
    ชื่อ =                -
    ถ้า ชื่อ เป็น None: ส่งกลับ "สวัสดี!"
        มิฉะนั้น: ส่งกลับ "สวัสดี {0}!".format(ชื่อ)
```

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

# กำหนดฟังก์ชันที่ส่งข้อความไปยังบอท: send_message def send_message(message):

    # พิมพ์ user_template รวมถึง user_message print(user_template.format(message))

    # รับคำตอบจากบอทต่อข้อความ response = respond(message)

    # พิมพ์เทมเพลตบอทรวมทั้งการตอบสนองของบอท print(bot_template.format(response))
```

```
# ส่งข้อความ
send_message("ฉัน ชื่อเดวิด คอปเปอร์ฟิลด์") send_message("เรียก ฉันว่าอิ
ชมาเอล") send_message("คนอื่น เรียกฉันว่าแคสแซน
ดรา") send_message("ฉัน เดินไปโรงเรียน")
```

```
USER : ชื่อผมคือ David Copperfield BOT : สวัสดี
David Copperfield!
USER : เรียกฉันว่า อิชมา
เอล BOT : สวัสดี อิชมาเอล!
USER : ผู้คนเรียกฉันว่า Cassandra BOT :
สวัสดี Cassandra!
USER : ฉันเดินไปโรงเรียน
BOT : สวัสดี!
```

การสร้างผู้ช่วยเสมือนจริง

การสร้างผู้ช่วยเสมือนสามารถทำได้ตั้งแต่ง่ายไปจนซับซ้อนอย่างเหลือเชื่อ ขึ้นอยู่กับความซับซ้อนของฟังก์ชันที่คุณต้องการ

ผู้ช่วยเสมือนอัจฉริยะ (IVA) หรือผู้ช่วยส่วนตัวอัจฉริยะ (IPA)

- ตัวแทนซอฟต์แวร์ที่สามารถดำเนินการงานหรือบริการให้กับบุคคลตามคำสั่งหรือคำถาม

- บางครั้งใช้คำว่า “Chatbot” เพื่ออ้างถึงผู้ช่วยเสมือนที่เข้าถึงโดย
  การสนทนาออนไลน์ที่มีวัตถุประสงค์เพื่อความบันเทิงเท่านั้น

- ผู้ช่วยเสมือนบางคนสามารถตีความคำพูดของมนุษย์และตอบสนองผ่านเสียงสังเคราะห์ได้

- ผู้ใช้สามารถ
  - ถามคำถามผู้ช่วย • ควบคุมอุปกรณ์ระบบ
    อัตโนมัติในบ้านและการเล่นสื่อผ่านเสียง และจัดการงานพื้นฐานอื่นๆ เช่น อีเมล รายการสิ่งที่ต้องทำ
  -

ผู้ช่วยเสมือน (VA)

• NLP ช่วยให้แชทบอทเข้าใจภาษาได้ในขณะที่มนุษย์พูด

• VA ไม่เพียงแค่อ่านคำพูด แต่สามารถ • เข้าใจเจตนาและ

- เข้าใจบริบทของคำถาม/การสนทนา

วิธีนี้ช่วยให้การโต้ตอบดำเนินไปในรูปแบบการสนทนาแทนที่จะเป็นช่วงถาม-ตอบ

• Chatbots ที่ใช้ NLP สามารถสนทนากับผู้ใช้ได้เช่นเดียวกับการสนทนากับ
ตัวแทนของมนุษย์และได้รับคำตอบที่คล้ายกัน

มาฝึกกันเถอะ!

การสร้างผู้ช่วยเสมือนใน Python

- เพื่อทำความเข้าใจฟังก์ชันพื้นฐานของผู้ช่วยเสมือนที่ค่อนข้างง่ายก่อนที่จะเจาะลึกมากขึ้น

- ก่อนอื่น ให้ติดตั้งโมดูลและไลบรารีที่เกี่ยวข้อง:

```
pip ติดตั้ง pyttsx3
pip ติดตั้ง SpeechRecognition
pip ติดตั้ง PyAudio
```

- SpeechRecognition: เป็นหนึ่งในไลบรารี Python สำหรับการจดจำและประมวลผลคำพูดของมนุษย์ •

Pyttsx3: เป็นไลบรารีการแปลงข้อความเป็นคำพูดใน Python

การสร้างผู้ช่วยเสมือนใน Python

- นำเข้าโมดูลและไลบรารี

```
นำเข้า pyttsx3 นำเข้า
speech_recognition เป็น sr นำเข้า webbrowser
นำเข้า datetime
```

การสร้างผู้ช่วยเสมือนใน Python

- ฟังก์ชัน “ผู้ช่วย” (pytsx3)
  - เพื่อกำหนดว่าผู้ช่วยของคุณคือ “ใคร” หรือ “อะไร” • เพื่อกำหนด เสียงของผู้ช่วยเสมือนง่าย ๆ นี้
- สลับระหว่างเสียงชายและหญิงโดยสลับ 0 และ 1 ใน voices[].id
- ฟังก์ชัน “runAndWait” จะควบคุมคิวและทำให้สามารถได้ยินเสียงพูดในระบบ

```
def assistant(audio): engine = pyttsx3.init()
        # getter: เพื่อรับค่าคุณสมบัติ engine ปัจจุบัน voices
        = engine.getProperty('voices') # วิธี setter # [0] สำหรับเสียง
        ชาย # [1] สำหรับเสียงหญิง engine.setProperty('voice',
        voices[1].id)




        # วิธีการควบคุม เครื่องมือพูดของผู้ช่วยพูด(เสียง)


        # บล็อก/ประมวลผลคำสั่งที่อยู่ในคิว engine.runAndWait()
```

การสร้างผู้ช่วยเสมือนใน Python

- ฟังก์ชันการทักทาย
  - เขียนวลีใดๆ ที่คุณต้องการให้ผู้ช่วยเสมือนใช้

```
def greeting(): # นี่เป็นคำทักทาย
    ง่ายๆ และแจ้งให้ผู้ใช้ทราบว่า # ผู้ช่วยได้เริ่มใช้ งาน assistant("สวัสดี ฉันคือผู้
    ช่วยเสมือนของคุณ



ฉันจะช่วยคุณได้อย่างไร")
```

การสร้างผู้ช่วยเสมือนใน Python

- ส่วนประกอบหลัก

```
กำหนด core_code():
        # อันดับแรกเราจะเรียก greeting # เพื่อทำเครื่องหมาย
        greeting เริ่มต้น()


รหัสแกน()
```

ฟังก์ชั่นอินพุตเสียง: การยอมรับคำสั่งเสียง

• กำหนดวิธีที่ผู้ช่วยประมวลผลคำสั่งด้วยวาจา • ตั้งค่าไมโครโฟนเป็นแหล่ง "การจดจำเสียง"

```
def audioinput(): # ฟังก์ชันนี้
    ใช้สำหรับรับอินพุตเสียงจากผู้ใช้ aud = sr.Recognizer() โดยมี sr.Microphone() เป็น แหล่งที่มา:


        พิมพ์('การฟังและการประมวลผล')
        # การหยุดชั่วคราวที่นี่เป็นทางเลือก
        aud.pause_threshold = 0.7 audio =
        aud.listen(source)
        # การใช้ try (สำหรับคำสั่งที่ถูกต้อง) และข้อยกเว้นเมื่อผู้ช่วย # ไม่ "จับ" คำสั่ง try: print("understanding") # en-eu ใช้สำหรับสำเนียงภาษาอังกฤษเท่านั้น ในที่นี้เราสามารถ
        ใช้ 'en-GB' หรือ 'en-au' # สำหรับสำเนียงอังกฤษและออสเตรเลีย

        poet =
            aud.recognize_google(audio, language='en-
              us') print("you said: ", phrase) ยกเว้น Exception เป็น exp: print(exp) print("Can you please repeat that") return "None"




        กลับ วลี
```

- เพิ่มโค้ดนี้ลงในฟังก์ชัน core_code() เพื่อทดสอบ audioinput()

```
while (True): # การเปลี่ยน
        แบบสอบถามให้เป็นตัวพิมพ์เล็ก # เพื่อให้แน่ใจว่าจะทำงานได้เกือบตลอดเวลา วลี =
        audioinput().lower()



        หาก "คุณชื่ออะไร" ใน วลี: assistant("ฉัน คือผู้ช่วยเสมือนที่ไม่มีชื่อของคุณ")
                ดำเนินการต่อ



        # ทริกเกอร์/เงื่อนไขในการออกจากโปรแกรม elif "bye" ใน วลี: assistant("กำลังออก ขอ ให้เป็น
        วันที่ดี") exit()
```

• ฟังก์ชั่นของวัน: บอกวัน

```
def theDay(): # ฟัง
    ก์ชั่นนี้ใช้สำหรับวัน day =
    datetime.datetime.today().weekday() + 1 # การกำหนดหมายเลขจะทำให้ดูสะอาด
    ขึ้นเล็กน้อย Day_dict = { 1: 'วันจันทร์', 2: ' วันอังคาร', 3: 'วันพุธ', 4: 'วันพฤหัสบดี',
    5: 'วันศุกร์', 6 : 'วัน
        เสาร์', 7: 'วันอาทิตย์'
```

```
    } ถ้า วัน ใน Day_dict.keys(): weekday
        = Day_dict[day] print(weekday)
        assistant("มันคือ "
    + weekday)
```

```
ในขณะที่ (จริง):
        -
        elif "วันนี้เป็นวันอะไร" ใน วลี: theDay() ต่อไป
```

• ฟังก์ชันเวลา: บอกเวลา

```
def theTime(): # ฟังก์ชั่นนี้ใช้
    สำหรับเวลา time = str(datetime.datetime.now()) # เวลาต้อง
    ถูกแบ่งส่วนเพื่อให้เข้าใจเสียงได้ดีขึ้น print(time) hour = time[11:13] min =
    time[14:16] assistant(" เวลาขณะนี้คือ" + hour + "Hours and" +
    min + "Minutes")
```

```
ในขณะที่ (จริง):
    -
    elif "ตอนนี้กี่โมงแล้ว" ใน วลี: theTime() ต่อไป
```

การสร้างผู้ช่วยเสมือนใน Python

- ใช้โมดูลเว็บเบราว์เซอร์เพื่อเปิดเว็บไซต์ใดๆ

```
ในขณะที่ (จริง):
    -
    elif "เปิด google" ใน วลี: assistant("กำลังเปิด Google
                ") webbrowser.open("www.google.com") ดำเนิน
                การต่อ
```

การสร้างผู้ช่วยเสมือนใน Python

- ใช้โมดูล Wikipedia เพื่อค้นหาหัวข้อภายใน Wikipedia

ในขณะที่ (จริง):

-

```
elif "wiki" ใน วลี: # ดึงข้อมูลจาก Wiki
        assistant("กำลังตรวจสอบ วิกิพีเดีย ") วลี = วลี.replace("wiki ", "") # จะ
        จำกัดการสรุปให้เหลือสี่บรรทัด ผลลัพธ์ = wikipedia.summary(วลี, ประโยค=4)
        assistant("ตาม วิกิพีเดีย") assistant(ผลลัพธ์) ดำเนินการต่อ
```

คำถาม

อ้างอิง:

https://campus.datacamp.com/

https://medium.datadriveninvestor.com/