บทที่ 16 Python กับการประยุกต์ใช้ในงานด้านต่าง ๆ

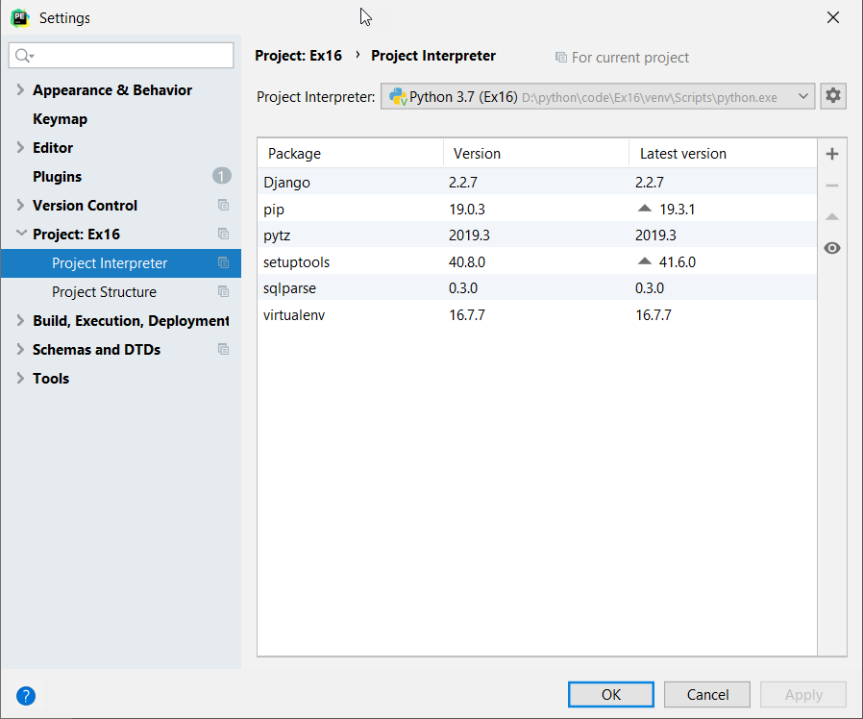
ด้วยคุณสมบัติของภาษา Python ที่รองรับการพัฒนางานที่หลากหลาย เนื่องจากมีโมดูลมากมายที่เราสามารถนำใช้ในพัฒนาตามที่ต้องการ เช่น งานด้านเว็บแอปพลิเคชัน งานด้านเครือข่าย หรืองานด้านวิทยาศาสตร์ข้อมูล เป็นต้น ดังนั้นในบทนี้ผู้อ่านจะได้เรียนรู้ถึงการประยุกต์ใช้ Python กับงานด้านต่าง ๆ ได้แก่

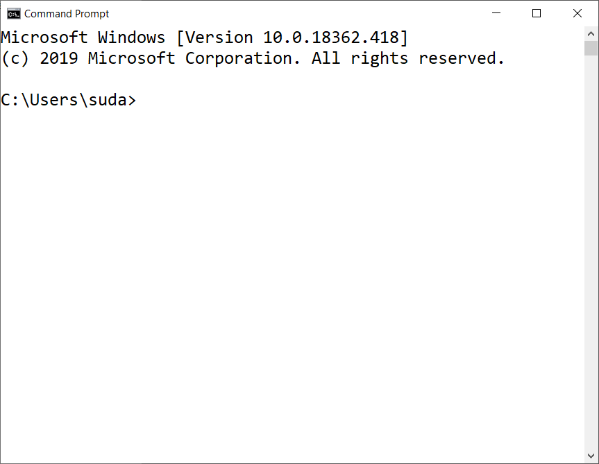
1. การเขียนเว็บแอปพลิเคชันด้วยโมดูล Django
2. การส่งข้อความผ่านเครือข่ายด้วยโมดูล socket
3. การส่งจดหมายอิเล็กทรอนิกส์ Gmail ด้วยโมดูล smtplib
4. การจัดการข้อมูลทางธุรกิจด้วยโมดูล pandas

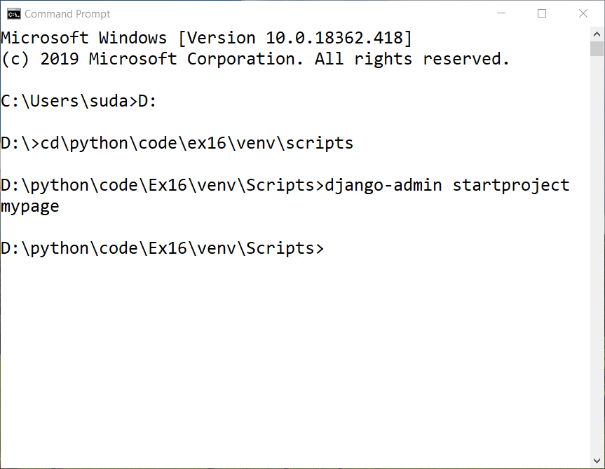
# การเขียนเว็บแอปพลิเคชันด้วยโมดูล Django

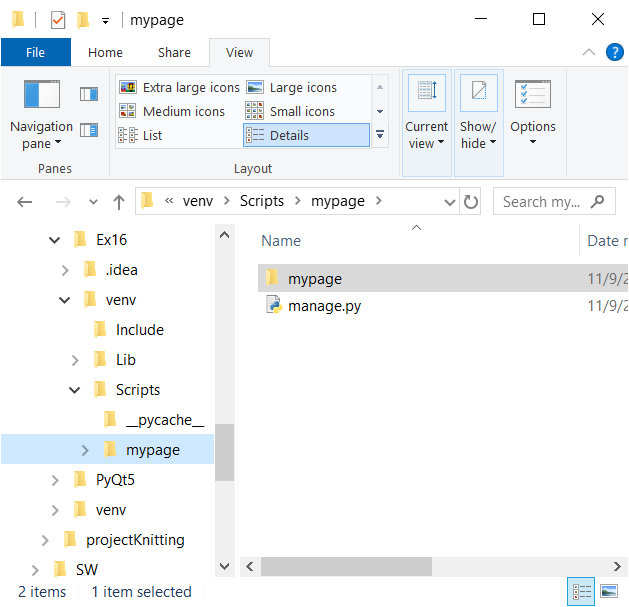
Django เป็น Framework ประเภท Open Source ที่สนับสนุนการพัฒนาโปรแกรมสำหรับงานที่เป็นเว็บแอปพลิเคชันด้วยภาษา Python โดยในภาษา Python ได้จัดเตรียมโมดูล Django ซึ่งผู้อ่านสามารถ ดำเนินการด้วยขั้นตอนเดียวกับการติดตั้งโมดูล pyodbc ซึ่งผู้เขียนได้อธิบายไว้ในบทที่ 13 แล้ว หลังจากติดตั้งได้สำเร็จแล้ว ผู้อ่านสามารถตรวจสอบได้โดยจะแสดงอยู่ในหน้าต่างของการตั้งค่า ดังนี้

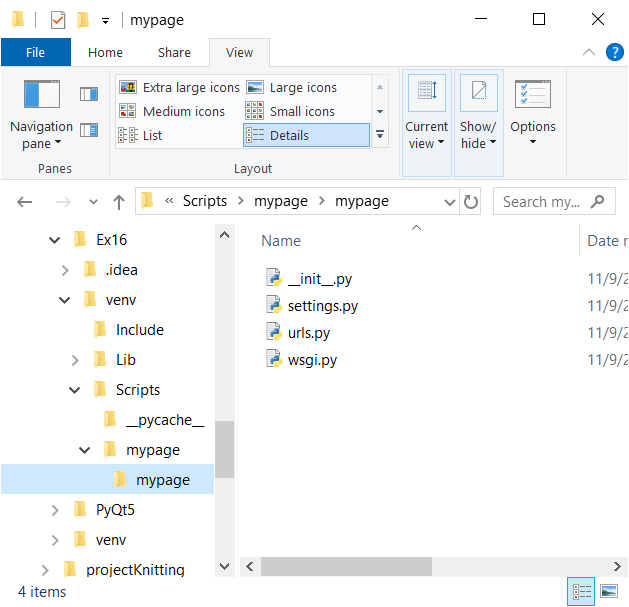
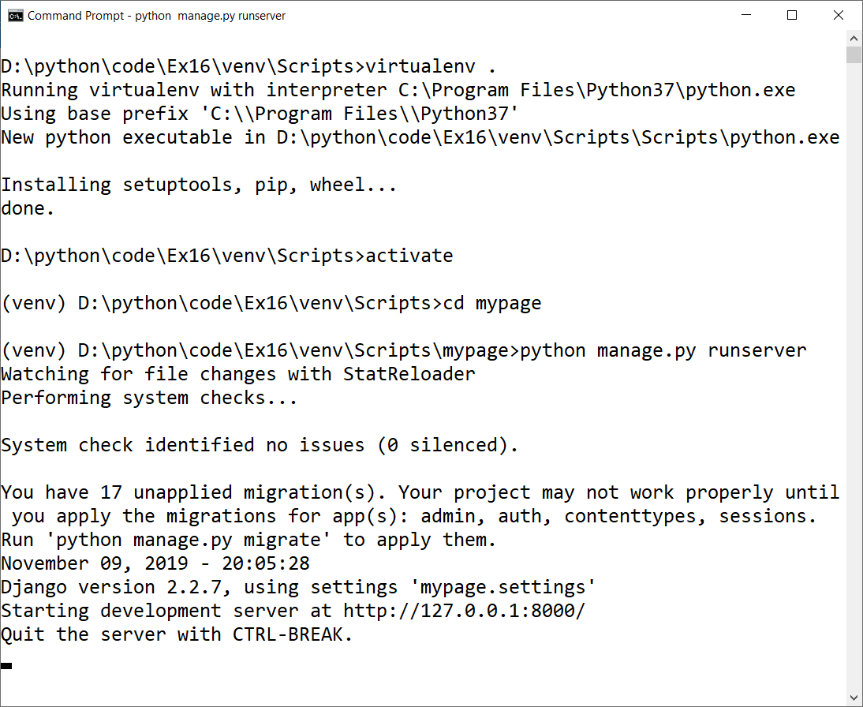
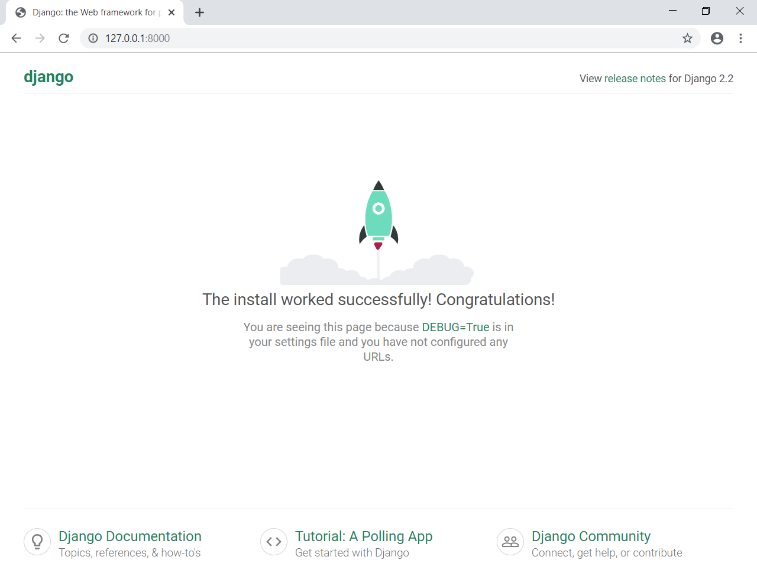
virtualenv

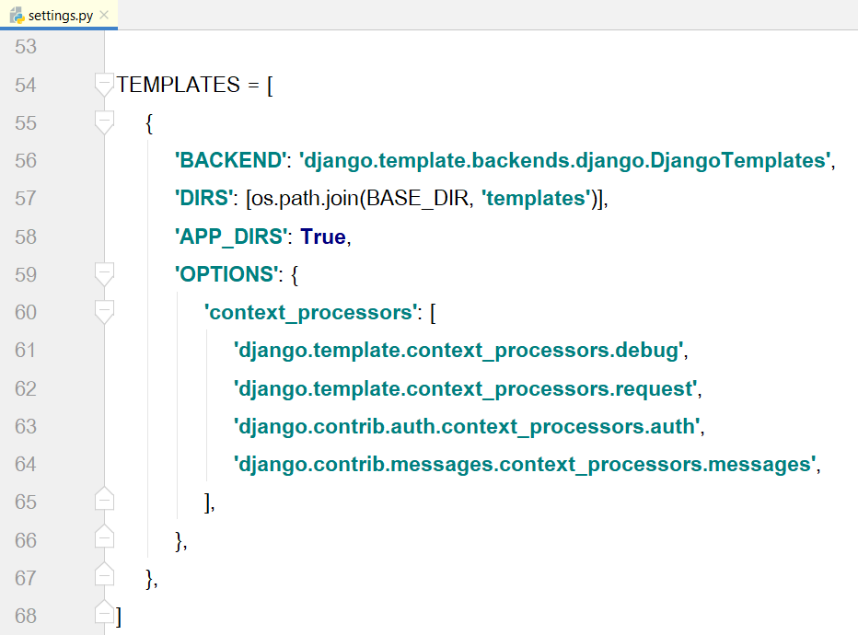
[[python\_16\_01.bmp]]

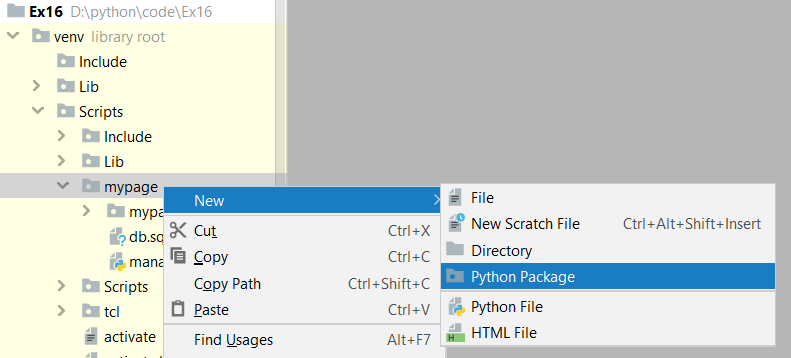
[[python\_16\_02.bmp]]

[[python\_16\_03.bmp]]

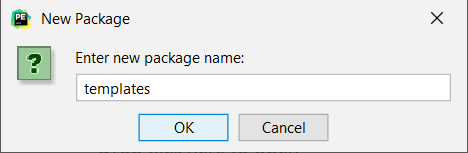
[[python\_16\_04.bmp]]

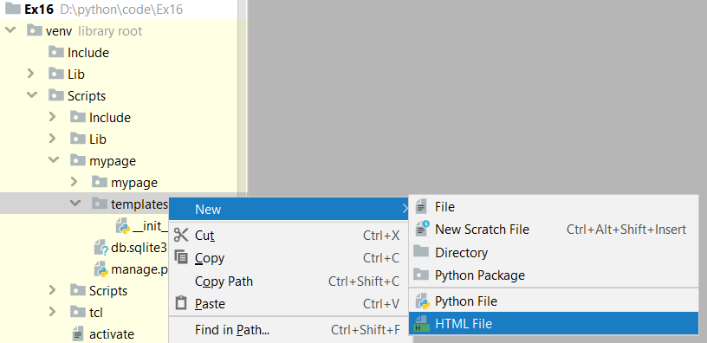
[[python\_16\_05.bmp]] [[python\_16\_06.bmp]] [[python\_16\_07.bmp]]

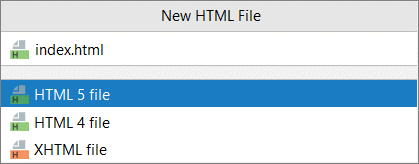
[[python\_16\_08.bmp]]

[[python\_16\_09.bmp]]

[[python\_16\_10.bmp]]

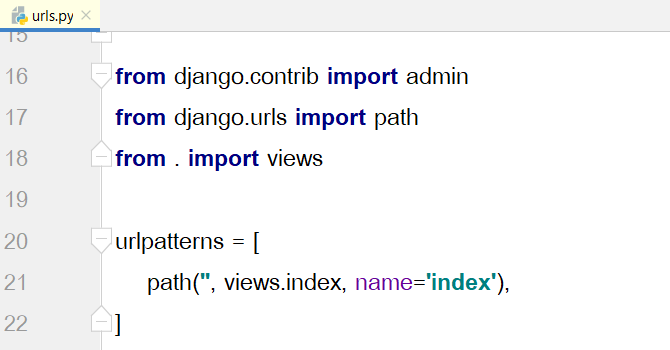
[[python\_16\_11.bmp]]

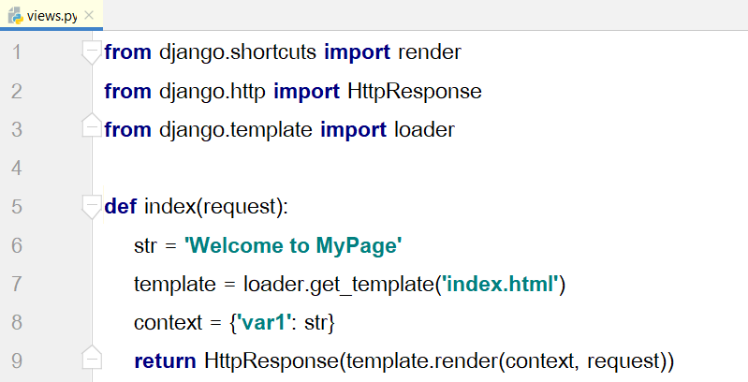
[[python\_16\_12.bmp]]

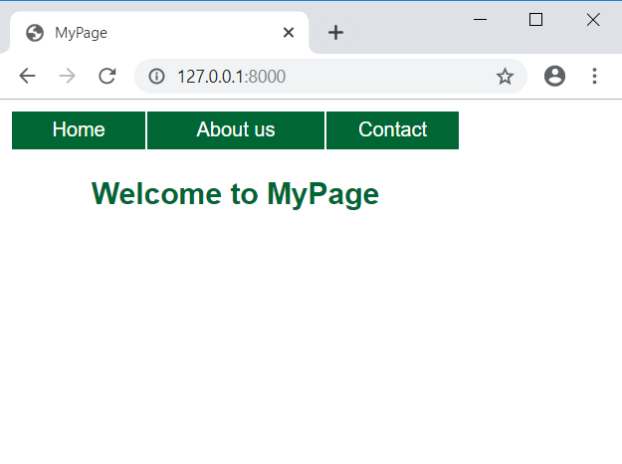
[[python\_16\_13.bmp]]

**ตัวอย่างโปรแกรม index.html** โปรแกรมแสดงหน้าเว็บ Home ซึ่งมีโค้ดดังนี้

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>MyPage</title>  <style type="text/css">  body,td,th {  font-size: 16px;  color: #006633;  font-family: Arial, Helvetica, sans-serif;  }  a:link {  color: #FFFFFF;  text-decoration: none;  }  a:visited {  text-decoration: none;  color: #FFFFFF;  }  a:hover {  text-decoration: none;  color: #FFFFFF;  }  a:active {  color: #FFFFFF;  }  </style>  </head>  <body>  <table width="75%" border="0" bordercolor="#FFFFFF">  <tbody align="center" valign="middle">  <tr height="30" bgcolor="#006633">  <td width="30%"><a href="/">Home</a></td>  <td width="40%"><a href="/about/">About us</a></td>  <td width="30%"><a href="/contact/">Contact</a></td>  </tr>  <tr height="30">  <td colspan="3"><h2>{{var1}}</h2></td>  </tr>  </tbody>  </table>  </body>  </html> |

[[python\_16\_14.bmp]]

[[python\_16\_15.bmp]]

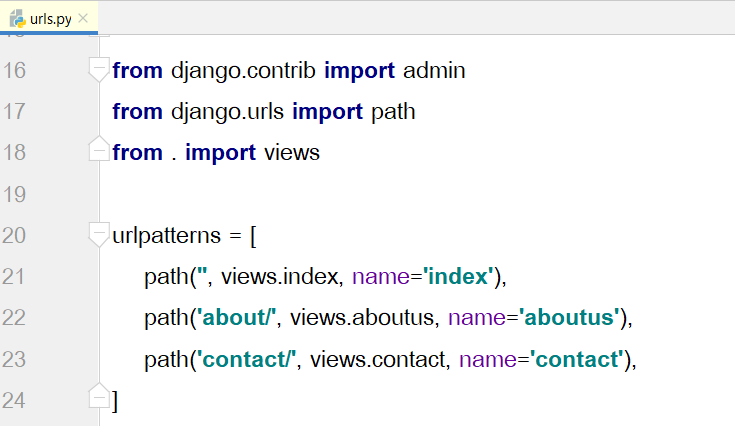
[[python\_16\_16.bmp]]

**ตัวอย่างโปรแกรม aboutus.html** โปรแกรมแสดงหน้าเว็บ About us ซึ่งมีโค้ดดังนี้

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>About</title>  <style type="text/css">  body,td,th {  font-size: 16px;  color: #FF0000;  font-family: Arial, Helvetica, sans-serif;  }  a:link {  color: #FFFFFF;  text-decoration: none;  }  a:visited {  text-decoration: none;  color: #FFFFFF;  }  a:hover {  text-decoration: none;  color: #FFFFFF;  }  a:active {  color: #FFFFFF;  }  </style>  </head>  <body>  <table width="75%" border="0" bordercolor="#FFFFFF">  <tbody align="center" valign="middle">  <tr height="30" bgcolor="#006633">  <td width="30%"><a href="/">Home</a></td>  <td width="40%"><a href="/about/">About us</a></td>  <td width="30%"><a href="/contact/">Contact</a></td>  </tr>  <tr height="30">  <td colspan="3"><h2>{{var2}}</h2></td>  </tr>  </tbody>  </table>  </body>  </html> |

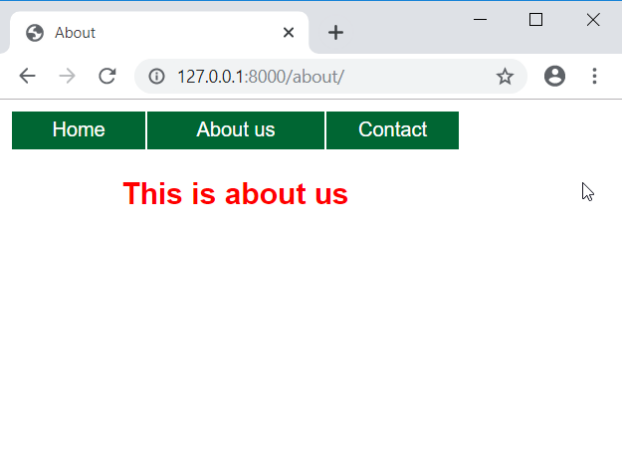
**ตัวอย่างโปรแกรม contact.html** โปรแกรมแสดงหน้าเว็บ About us ซึ่งมีโค้ดดังนี้

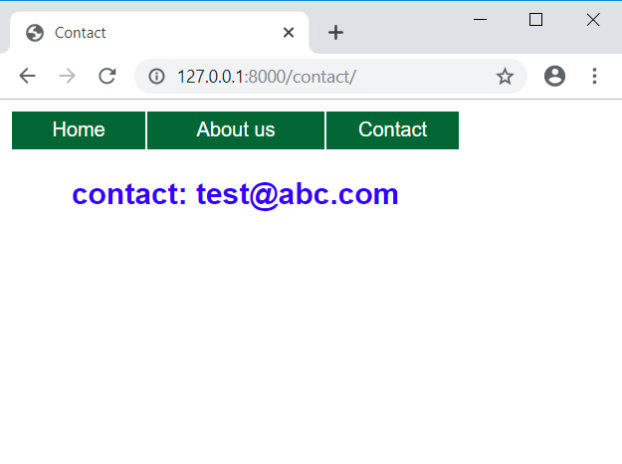
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Contact</title>  <style type="text/css">  body,td,th {  font-size: 16px;  color: #3300FF;  font-family: Arial, Helvetica, sans-serif;  }  a:link {  color: #FFFFFF;  text-decoration: none;  }  a:visited {  text-decoration: none;  color: #FFFFFF;  }  a:hover {  text-decoration: none;  color: #FFFFFF;  }  a:active {  color: #FFFFFF;  }  </style>  </head>  <body>  <table width="75%" border="0" bordercolor="#FFFFFF">  <tbody align="center" valign="middle">  <tr height="30" bgcolor="#006633">  <td width="30%"><a href="/">Home</a></td>  <td width="40%"><a href="/about/">About us</a></td>  <td width="30%"><a href="/contact/">Contact</a></td>  </tr>  <tr height="30">  <td colspan="3"><h2>{{var3}}</h2></td>  </tr>  </tbody>  </table>  </body>  </html> |

[[python\_16\_17.bmp]]

**ตัวอย่างโปรแกรม views.py** โปรแกรมแสดงฟังก์ชันสำหรับเรียกหน้าเว็บแต่ละหน้าให้ทำงาน ซึ่งมีโค้ดดังนี้

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | from django.shortcuts import render from django.http import HttpResponse from django.template import loader  def index(request):  str = 'Welcome to MyPage'  template = loader.get\_template('index.html')  context = {'var1': str}  return HttpResponse(template.render(context, request))  def aboutus(request):  str = 'This is about us'  template = loader.get\_template('aboutus.html')  context = {'var2': str}  return HttpResponse(template.render(context, request))  def contact(request):  str = 'contact: test@abc.com'  template = loader.get\_template('contact.html')  context = {'var3': str}  return HttpResponse(template.render(context, request)) |

[[python\_16\_18.bmp]]

[[python\_16\_19.bmp]]

## ขั้นตอนการสร้างโปรเจค

# การส่งข้อความผ่านเครือข่ายด้วยโมดูล socket

การเขียนโปรแกรมผ่านเครือข่ายแบบเครื่องลูกข่าย - แม่ข่าย (Client - Server) จะประกอบด้วยสถาปัตยกรรม 2 ส่วน คือ

1) คอมพิวเตอร์ที่ทำหน้าที่เป็น Server ซึ่งจะทำงานให้บริการอยู่ตลอดเวลา

2) คอมพิวเตอร์ที่ทำหน้าที่เป็น Client ที่สามารถร้องขอการเชื่อมต่อเพื่อใช้บริการจาก Server ได้

## ขั้นตอนการทำงานระหว่าง Client - Server Mode

ในการทำงานระหว่าง Client – Server Mode นั้น คอมพิวเตอร์ที่ทำหน้าที่เป็น Server จะมี หมายเลข IP Address ประจำเครื่อง และจะเปิดการเชื่อมต่อ (Connection) รอไว้ที่หมายเลข Port ใดๆ เพื่อรอการร้องขอการเชื่อมต่อจากคอมพิวเตอร์ที่ทำหน้าที่เป็น Client ซึ่ง Port ในที่นี้ก็คือช่องทางการเชื่อมต่อระหว่าง Server กับ Client โดยจะถูกกำหนดด้วยเลขจำนวนเต็มระหว่าง 0 - 65535 แต่ผู้อ่านไม่ควรนำหมายเลข Port ตั้งแต่ 0 - 1024 มาใช้ เนื่องจากหมายเลข Port ดังกล่าวถูกจองไว้เพื่อการใช้งานต่างๆ เช่น หมายเลข 80 ใช้เพื่องานบริการ Web Server, หมายเลข 21 ใช้เพื่องานบริการ FTP Server เป็นต้น ผู้เขียนจึงขอแนะนำให้ใช้หมายเลย Port ตั้งแต่ 1024 ขึ้นไป

การเขียนโปรแกรมเพื่อเชื่อมต่อการทำงานระหว่าง Server กับ Client ในภาษา Python จะใช้โมดูล Socket ซึ่งจะทำหน้าที่สร้างลิงค์เพื่อเปิด Socket หรือเรียกว่าเป็นการเปิด Port และมีการนำ Buffer มาใช้จัดเรียงข้อมูลไว้ในพื้นที่ที่ได้จัดเตรียมไว้ก่อน เพื่อแก้ปัญหาของการรับส่งข้อมูลเนื่องมาจากความเร็วในการสื่อสารระหว่าง Server กับ Client ที่แตกต่างกัน

## การเขียนโปรแกรมระหว่าง Client - Server Mode

เนื่องจากคอมพิวเตอร์ที่ทำหน้าที่เป็น Server จะทำงานให้บริการอยู่ตลอดเวลา การเขียนโปรแกรมฝั่ง Server Mode จึงเริ่มต้นจากการสร้างและผูก Socket เข้ากับหมายเลข IP Address ประจำเครื่อง เพื่อเปิดการเชื่อมต่อรอไว้ที่หมายเลข Port ที่กำหนด จนกระทั่งมีคอมพิวเตอร์ที่ทำหน้าที่เป็น Client ร้องขอการเชื่อมต่อผ่านหมายเลข IP Address และ หมายเลข Port ที่ตรงกัน การสื่อสารระหว่างกันก็จะเกิดขึ้นและสามารถส่งข้อความถึงกันได้

เพื่อความเข้าใจผู้อ่านสามารถศึกษาการเขียนโปรแกรมระหว่าง Client - Server Mode ได้จากตัวอย่างโปรแกรมต่อไปนี้

**ตัวอย่างโปรแกรมที่ 16-1** โปรแกรมการส่งข้อความฝั่ง Server Mode ซึ่งมีโค้ดดังนี้

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | import socket  s = socket.socket() host = "127.0.0.1" port = 5000 print("server IP:", host) print("... waiting connection request from client ...") s.bind((host, port)) s.listen(5) while True:  c, addr = s.accept()  print("... get connection from client ...")  while True:  sdata = input("server >>> ")  c.send(str.encode(sdata))  if str(sdata) == "BYE":  check = 0  c.close()  break  else:  rdata = c.recv(1024)  rdata = bytes.decode(rdata)  print("client >>> ", rdata)  if check == 0:  s.close()  break |

จากโปรแกรมที่ 16-1 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1 อิมพอร์ตโมดูล socket

บรรทัดที่ 3 สร้างและกำหนด socket เป็น ออบเจ็กต์ s

บรรทัดที่ 4 กำหนดหมายเลข IP Address เท่ากับ 127.0.0.1 ซึ่งเป็น หมายเลข IP Address ที่ใช้สำหรับทดสอบการทำงานของโปรแกรม และในที่นี้จะจำลองเครื่องคอมพิวเตอร์ของเราให้เป็นทั้ง Server และ Client

บรรทัดที่ 5 กำหนดหมายเลข Port เท่ากับ 5000

บรรทัดที่ 6-7 แสดงข้อความ หมายเลข IP Address และข้อความรอรับการร้องขอการเชื่อมต่อจาก client

บรรทัดที่ 8-9 ผูกออบเจ็กต์ s เข้ากับหมายเลข IP Address และหมายเลข Port และรอการเชื่อมต่อ

บรรทัดที่ 10-26 ใช้คำสั่ง while True เพื่อวนรอบรอการเชื่อมต่อจาก Client ไปเรื่อยๆ

บรรทัดที่ 11-12 เมื่อมีการร้องขอการเชื่อมต่อมาจาก Client แสดงข้อความได้รับการเชื่อมต่อจาก Client

บรรทัดที่ 13-23 ใช้คำสั่ง while True เพื่อวนรอบการรับข้อมูลชนิดข้อความและ encode ข้อความให้เป็น

ข้อมูลชนิด bytes ก่อนส่งให้ client โดยใช้ข้อความ BYE เป็นจุดสิ้นสุดของการส่งข้อความ

ตามคำสั่งบรรทัดที่ 16-19

บรรทัดที่ 21-23 รับข้อความจาก client และจัดเรียงข้อมูลใน buffer ครั้งละ 1024 bytes และทำการ

decode ข้อความชนิด bytes เป็น string แล้วแสดงผล

บรรทัดที่ 24-26 ถ้าสิ้นสุดการส่งข้อความแล้ว ให้ปิดการเชื่อมต่อจาก client

**ตัวอย่างโปรแกรมที่ 16-2** โปรแกรมการส่งข้อความฝั่ง Client Mode ซึ่งมีโค้ดดังนี้

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | import socket  s = socket.socket() host = "127.0.0.1" port = 5000 print("server IP:", host) print("... request connection to server ...") s.connect((host, port)) while True:  sdata = s.recv(1024)  sdata = bytes.decode(sdata)  print("server >>> ", sdata)  if sdata == "BYE":  s.close()  break  else:  sdata = input("client >>> ")  s.send(str.encode(sdata)) |

จากโปรแกรมที่ 16-2 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1 อิมพอร์ตโมดูล socket

บรรทัดที่ 3 สร้างและกำหนด socket เป็น ออบเจ็กต์ s

บรรทัดที่ 4 กำหนดหมายเลข IP Address เท่ากับ 127.0.0.1

บรรทัดที่ 5 กำหนดหมายเลข Port เท่ากับ 5000

บรรทัดที่ 6-7 แสดงข้อความ หมายเลข IP Address และข้อความร้องขอการเชื่อมต่อไปยัง server

บรรทัดที่ 8 เชื่อมต่อออบเจ็กต์ s เข้ากับหมายเลข IP Address และหมายเลข Port

บรรทัดที่ 9-18 ใช้คำสั่ง while True เพื่อวนรอบการรับข้อความจาก server

บรรทัดที่ 10-15 รับข้อความจาก server และจัดเรียงข้อมูลใน buffer ครั้งละ 1024 bytes และทำการ

decode ข้อความชนิด bytes เป็น string แล้วแสดงผล โดยใช้ข้อความ BYE เป็นจุดสิ้นสุด

ของการรับข้อความ ตามคำสั่งบรรทัดที่ 13-15

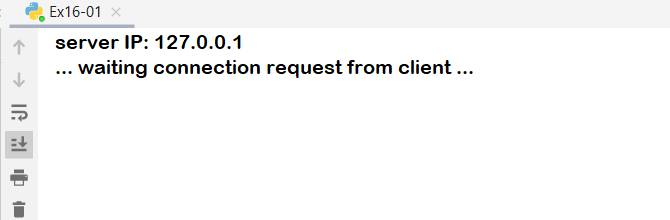
บรรทัดที่ 17-18 รับข้อมูลชนิดข้อความและ encode ข้อความให้เป็นข้อมูลชนิด bytes ก่อนส่งกลับให้

Server

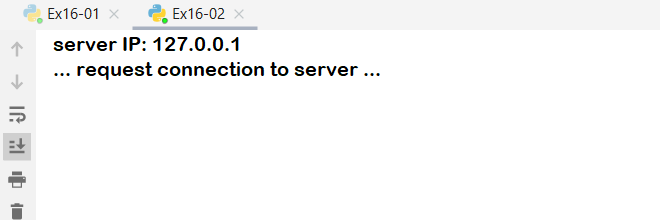
### ผลการทำงานของโปรแกรม

เนื่องจากการทำงานโปรแกรม 16-1 และโปรแกรม 16-2 เป็นการส่งข้อความระหว่างกัน เราจึงต้องรันทั้ง 2 โปรแกรม ควบคู่กันไป โดยมีผลการทำงาน ดังนี้

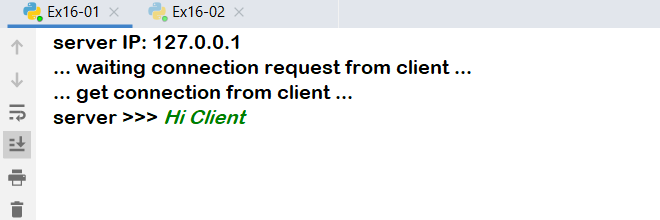
1. ผลการทำงานของโปรแกรม 16-1: server รอการเชื่อมต่อจาก client

 [[python\_16\_20.bmp]]

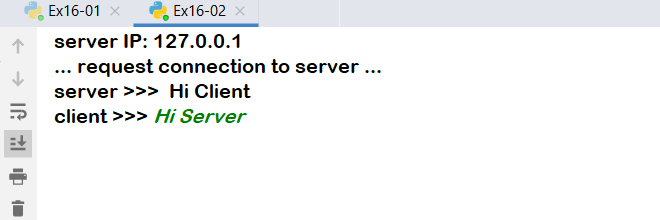
1. ผลการทำงานของโปรแกรม 16-2: client ร้องขอการเชื่อมต่อไปยัง server

 [[python\_16\_21.bmp]]

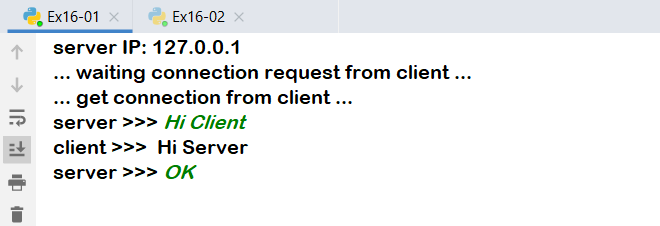
1. ผลการทำงานของโปรแกรม 16-1: server แสดงข้อความได้รับการเชื่อมต่อจาก client และพิมพ์ข้อความ Hi Client เพื่อส่ง

 [[python\_16\_22.bmp]]

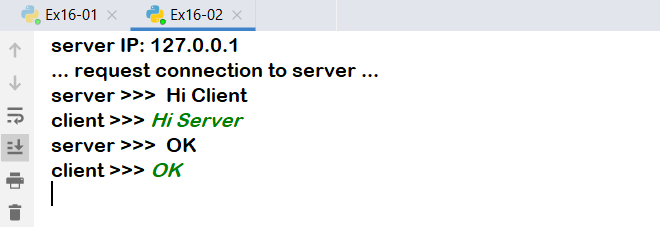
1. ผลการทำงานของโปรแกรม 16-2: client แสดงข้อความ Hi Client และพิมพ์ข้อความ Hi Server เพื่อส่ง

 [[python\_16\_23.bmp]]

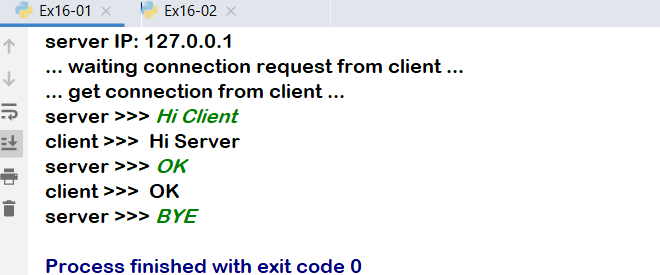
1. ผลการทำงานของโปรแกรม 16-1: server แสดงข้อความ Hi Server และพิมพ์ข้อความ OK เพื่อส่ง

 [[python\_16\_24.bmp]]

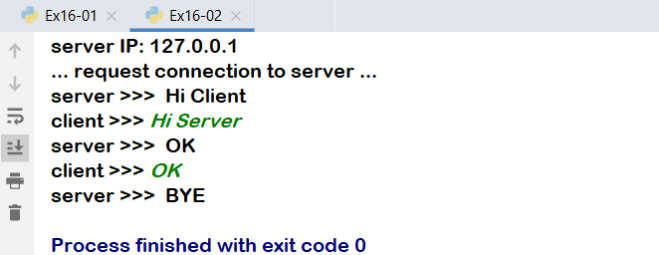
1. ผลการทำงานของโปรแกรม 16-2: client แสดงข้อความ OK และพิมพ์ข้อความ OK เพื่อส่ง

 [[python\_16\_25.bmp]]

1. ผลการทำงานของโปรแกรม 16-1: server แสดงข้อความ OK และพิมพ์ข้อความ BYE เพื่อส่ง และโปรแกรมก็จะจบการทำงาน

 [[python\_16\_26.bmp]]

1. ผลการทำงานของโปรแกรม 16-2: แสดงข้อความ BYE โปรแกรมก็จะจบการทำงาน

 [[python\_16\_27.bmp]]

# การส่งจดหมายอิเล็กทรอนิกส์ Gmail ด้วยโมดูล smtplib

เนื่องจากคอมพิวเตอร์ที่ทำหน้าที่เป็น Server จะทำงานให้บริการอยู่ตลอดเวลา การเขียนโปรแกรมฝั่ง Server Mode จึงเริ่มต้นจากการสร้างและผูก Socket เข้ากับหมายเลข IP Address ประจำเครื่อง เพื่อเปิดการเชื่อมต่อรอไว้ที่หมายเลข Port ที่กำหนด จนกระทั่งมีคอมพิวเตอร์ที่ทำหน้าที่เป็น Client ร้องขอการเชื่อมต่อผ่านหมายเลข IP Address และ หมายเลข Port ที่ตรงกัน การสื่อสารระหว่างกันก็จะเกิดขึ้นและสามารถส่งข้อความถึงกันได้

เพื่อความเข้าใจผู้อ่านสามารถศึกษาการเขียนโปรแกรมระหว่าง Client - Server Mode ได้จากตัวอย่างโปรแกรมต่อไปนี้

**ตัวอย่างโปรแกรมที่ 16-3** โปรแกรมการส่งข้อความฝั่ง Server Mode ซึ่งมีโค้ดดังนี้

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | import smtplib  recipient = input("recipient: ") subject = input("ืsubject: ") body = input("message: ")  try:  server = smtplib.SMTP('smtp.gmail.com:587')  server.ehlo()  server.starttls()  username = 'coconuties@gmail.com'  password = 'Pizza2400'  server.login(username, password)  msg = 'Subject: {}\n\n{}'.format(subject, body)  server.sendmail(username, recipient, msg)  server.quit()  print("Done") except:  print("Fail... try again") |

จากโปรแกรมที่ 16-3 อธิบายการทำงานของโปรแกรมได้ ดังนี้

บรรทัดที่ 1 อิมพอร์ตโมดูล socket

บรรทัดที่ 3 สร้างและกำหนด socket เป็น ออบเจ็กต์ s

บรรทัดที่ 4 กำหนดหมายเลข IP Address เท่ากับ 127.0.0.1 ซึ่งเป็น หมายเลข IP Address ที่ใช้สำหรับทดสอบการทำงานของโปรแกรม และในที่นี้จะจำลองเครื่องคอมพิวเตอร์ของเราให้เป็นทั้ง Server และ Client

บรรทัดที่ 5 กำหนดหมายเลข Port เท่ากับ 5000

บรรทัดที่ 6-7 แสดงข้อความ หมายเลข IP Address และข้อความรอรับการร้องขอการเชื่อมต่อจาก client

บรรทัดที่ 8-9 ผูกออบเจ็กต์ s เข้ากับหมายเลข IP Address และหมายเลข Port และรอการเชื่อมต่อ

บรรทัดที่ 10-26 ใช้คำสั่ง while True เพื่อวนรอบรอการเชื่อมต่อจาก Client ไปเรื่อยๆ

บรรทัดที่ 11-12 เมื่อมีการร้องขอการเชื่อมต่อมาจาก Client แสดงข้อความได้รับการเชื่อมต่อจาก Client

บรรทัดที่ 13-23 ใช้คำสั่ง while True เพื่อวนรอบการรับข้อมูลชนิดข้อความและ encode ข้อความให้เป็น

ข้อมูลชนิด bytes ก่อนส่งให้ client โดยใช้ข้อความ BYE เป็นจุดสิ้นสุดของการส่งข้อความ

ตามคำสั่งบรรทัดที่ 16-19

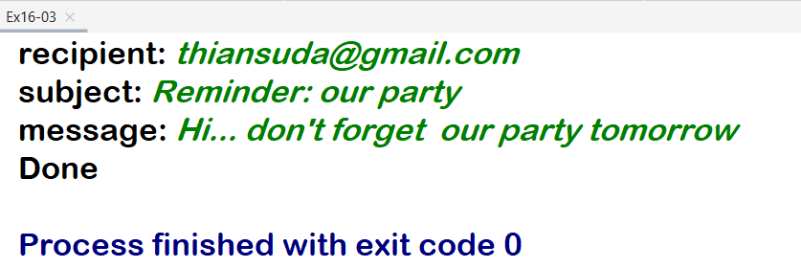
บรรทัดที่ 21-23 รับข้อความจาก client และจัดเรียงข้อมูลใน buffer ครั้งละ 1024 bytes และทำการ

decode ข้อความชนิด bytes เป็น string แล้วแสดงผล

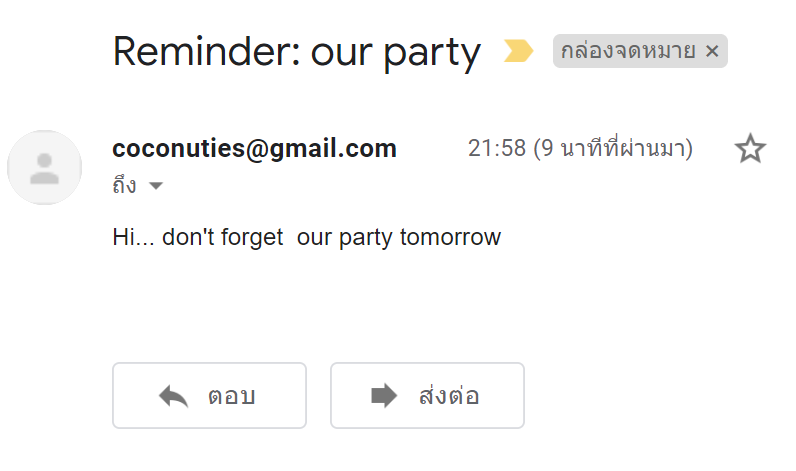
บรรทัดที่ 24-26 ถ้าสิ้นสุดการส่งข้อความแล้ว ให้ปิดการเชื่อมต่อจาก client

### ผลการทำงานของโปรแกรม

1. ผลการทำงานของโปรแกรม 16-3: server รอการเชื่อมต่อจาก client

 [[python\_16\_28.bmp]]

1. เมื่อตรวจสอบ Gmail พบว่า จะได้รับ e-mail จาก coconuties@gmail.com

 [[python\_16\_29.bmp]]

# การจัดการข้อมูลทางธุรกิจด้วยโมดูล pandas

### สรุปท้ายบท

ในบทนี้ผู้อ่านได้เรียนรู้การเขียนโปรแกรมผ่านเครือข่ายแบบเครื่องลูกข่าย - แม่ข่าย (client - server) ซึ่งประกอบด้วยส่วนของ server ที่เปิดการเชื่อมต่อ และส่วนของ client ที่สามารถร้องขอการเชื่อมต่อ server ที่ระบุในโปรแกรมฝั่ง client ผ่าน socket ที่มีการผูกเข้ากับหมายเลข IP address และ หมายเลข port ที่ระบุในโปรแกรมฝั่ง server ทำให้โปรแกรมทั้ง 2 ฝั่ง สามารถส่งข้อความถึงกันได้ ซึ่งผู้อ่านสามารถนำไปประยุกต์ใช้กับการส่งข้อความระหว่างกันด้วยหมายเลข IP Address จริงได้

# แบบฝึกหัด

จงเขียนโปรแกรมส่งข้อความระหว่างคอมพิวเตอร์ 2 เครื่อง ด้วยหมายเลข IP Address ประจำแต่ละเครื่อง ที่มีการเชื่อมต่อกันผ่านเครือข่าย LAN