

# ปฏิบัติการ 1 วิชา 01204111 คอมพิวเตอร์และการโปรแกรม

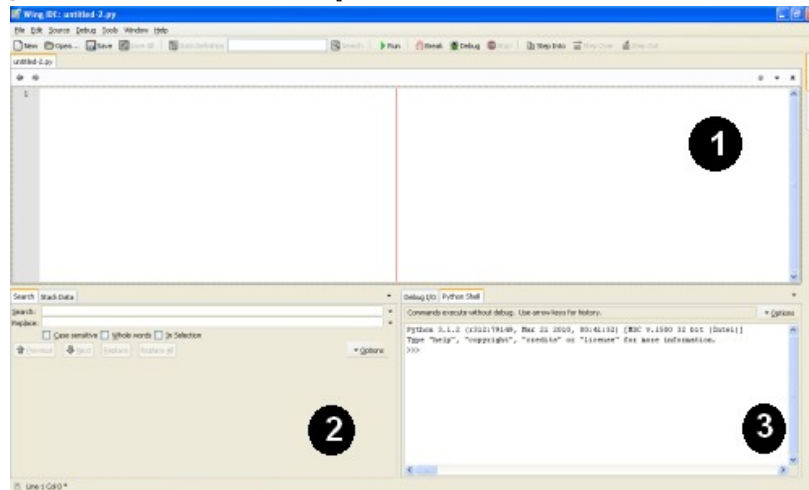
ชื่อ \_\_\_\_\_ เลขประจำตัว \_\_\_\_\_ หมู่ \_\_\_\_\_

เป้าหมาย

- หัดใช้ WingIDE และรู้จักความผิดพลาดและปัญหาในโปรแกรมภาษาไพธอนที่เกิดจากการพิมพ์ผิด
- หัดใช้ Python แบบโต้ตอบ ผ่านทาง Python Shell
- ทดลอง Turtle Graphics
- หัดพิมพ์และทดลองโปรแกรมภาษา Python

## ส่วนที่ 1 รู้จักกับ Wing IDE 101

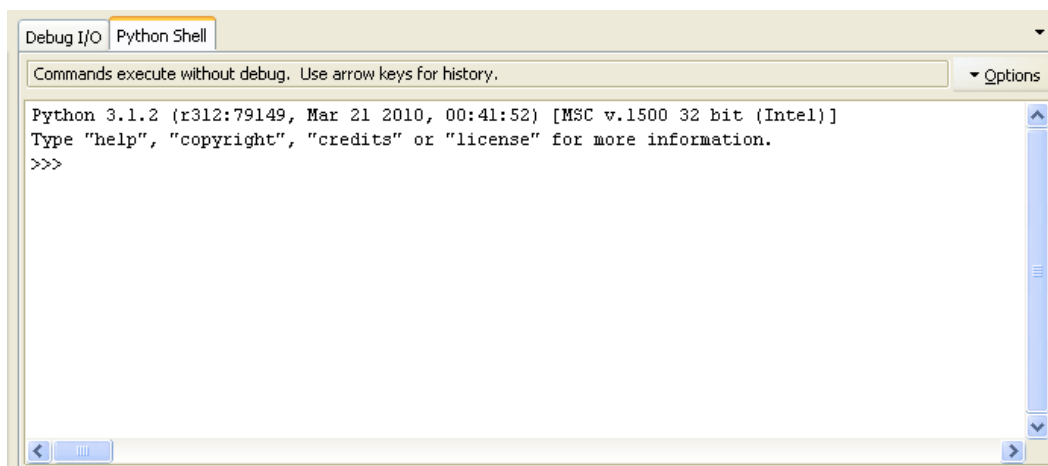
เมื่อเราเข้าสู่โปรแกรม Wing IDE 101 เราจะพบหน้าจอตั้งรูปด้านล่าง



หน้าจอตั้งกล่าวโดยปกติจะถูกแบ่งออกเป็น 3 ส่วน ดังนี้

- ส่วนที่ 1 คือส่วนสำหรับพิมพ์และแก้ไขโปรแกรม ถ้าส่วนนี้ว่างอยู่ สามารถกดปุ่ม **New** ที่บนทูลบาร์เพื่อเปิดไฟล์โปรแกรมว่างมาเพื่อแก้ไขได้
- ส่วนที่ 2 และ 3 คือส่วนพื้นที่เครื่องมือ โดยปกติในส่วนที่ 3 จะมีแท็บ **Python Shell** อยู่ ซึ่งเราจะใช้เพื่อพิมพ์คำสั่งโต้ตอบกับ Python

ส่วน Python Shell แสดงตั้งรูปด้านล่าง สังเกตว่าเราเลือกแท็บ Python Shell อยู่ไม่ใช่แท็บ Debug I/O (แสดงเป็นแถบสีส้ม)



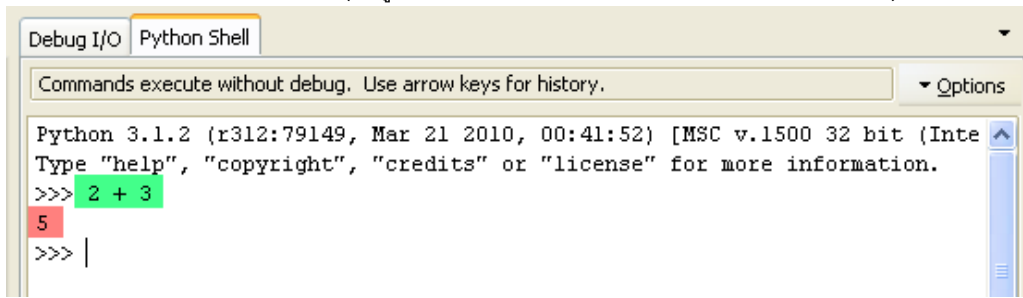
เราจะทดลอง Python โดยพิมพ์คำสั่งลงในส่วนดังกล่าวของ Wing IDE 101 เพื่อสร้างความคุ้นเคยกับแนวคิดพื้นฐานต่าง ๆ สิ่งที่เราทดลองนี้จะได้ไปเรียนอย่างละเอียดอีกครั้งในการบรรยายครั้งที่ 2

### 1.1 หัดพิมพ์โปรแกรมและทดลองแบบโต้ตอบ

ทดลองพิมพ์คำสั่งต่อไปนี้ (พิมพ์ต่อเครื่องหมาย >>> )

2 + 3

เมื่อพิมพ์แล้วให้กด Enter จะได้ผลลัพธ์ดังต่อไปนี้ (ในรูปเน้นส่วนที่พิมพ์เป็นสีเขียว ส่วนผลลัพธ์เป็นสีแดง)



ในส่วนต่อ ๆ ไป เราอาจจะแสดงผลจากการโต้ตอบกับ Python Shell ในลักษณะคล้ายกับรูปข้างต้นนี้ กล่าวคือ บรรทัดที่เราป้อนเข้าไปจะขึ้นต้นด้วยเครื่องหมาย >>> พร้อมทั้งเน้นด้วยตัวหนา ส่วนบรรทัดที่ Python ตอบมาจะแสดงเป็นบรรทัดที่ไม่มีเครื่องหมายดังกล่าวขึ้นต้นแต่จะแสดงด้วยตัว อักษรเอียงแทน เช่นจากตัวอย่างข้างต้น แทนที่จะใช้รูปเราอาจจะเขียนได้ดังนี้

```
>>> 2 + 3          # นี่คือบรรทัดที่เราพิมพ์
5          # นี่คือผลลัพธ์
```

ในการใช้งาน Python Shell ระบบจะรับสิ่งที่เราพิมพ์เข้าไปแล้วนำไปประมวลผล ถ้าเป็นนิพจน์ (เช่น 2 + 3) ระบบจะคำนวณค่าแล้วแสดงผลลัพธ์ออกมา

นิพจน์นั้นอาจจะเป็นแค่ค่าคงที่ธรรมดา เช่น

```
>>> 100
100
```

หรืออาจมีความซับซ้อนก็ได้ เช่น

```
>>> 1+2+3+4+5+6+7*100-30
691
```

หรือ

```
>>> 1/2 + 1/4 + 1/8 + 1/16
0.9375
```

เป็นต้น

ในส่วนต่อ ๆ ไป ให้นิทดลองทดลองกับ Python Shell โดยพิมพ์คำสั่งที่ระบุให้ตามลำดับ

### 1.1.1 วงเล็บ

เราสามารถใช้ Python เป็นเหมือนเครื่องคิดเลขได้ ทดลองพิมพ์นิพจน์สองนิพจน์นี้

```
2 + 3 * 5
```

และ

```
(2 + 3) * 5
```

ผลลัพธ์ของทั้งสองคำสั่งต่างกันเพราะอะไร?

พิจารณาคำสั่งด้านล่าง และให้คาดการณ์ผลลัพธ์ก่อนที่จะทดลองพิมพ์ใน Python Shell

```
2 + (3 * 5)
```

## 1.1.2 ตัวแปร

สำหรับนิพจน์ที่ได้คำนวณค่าแล้ว เราสามารถนำค่าที่ได้นั้นมาใช้ได้อีก โดยไม่ต้องคำนวณซ้ำ ๆ ผ่านทาง ตัวแปร

เราสามารถกำหนดค่าให้กับ ตัวแปร ได้โดยใช้เครื่องหมาย = เช่น (ทดลองพิมพ์ตามไปด้วย)

```
g = 9.81
```

การกำหนดค่าให้กับตัวแปรเป็นการสั่งให้ตัวแปร อ้างถึง ค่านั้น เขียนอธิบายเป็นรูปได้ดังนี้

$g \longrightarrow 9.81$

หมายเหตุ: ระหว่างตัวแปรและเครื่องหมายเท่ากับไม่จำเป็นต้องเว้นช่องว่างก็ได้ อย่างไรก็ตามเราแนะนำให้เว้นเพื่อให้อ่านได้ง่าย

เราสามารถนำตัวแปรมาใช้งานได้ดังตัวอย่างต่อไปนี้

```
>>> g
9.81

>>> 10 * g
98.10000000000001
```

หมายเหตุ: เครื่องหมาย \* คือเครื่องหมายแทนการคูณ สังเกตว่าความละเอียดของการคำนวณบนคอมพิวเตอร์มีจำกัด แทนที่เราจะได้ค่า 98.1 พอดี กลับมีเศษเล็กน้อยรวมมาด้วย

เราเรียกตัวแปรว่าตัวแปร เพราะเราสามารถกำหนดค่าใหม่ให้กับมันได้ ดังเช่นการทดลองต่อไปนี้

```
>>> g
9.81
>>> g + 5
14.81
>>> g = 100
>>> g + 5
105
```

พิจารณาการทำงานต่อไปนี้ (ทดลองพิมพ์ไปด้วย)

```
>>> a = 100
>>> b = a + 10
>>> b

110
>>> a = 200
>>> a + 10
210
>>> b

110
```

เราสังเกตอะไรได้จากการทดลองนี้? ตัวแปร b มีการเปลี่ยนแปลงอย่างไรเมื่อเทียบกับตัวแปร a

ทดลองพิมพ์คำสั่งทั้ง 3 นี้

```
>>> x
>>> x = 10
>>> x
```

เปรียบเทียบคำตอบที่ได้จากคำสั่งทั้ง 3 อธิบายว่าเหตุใดจึงเป็นเช่นนั้น

ทดลองคำสั่งต่อไปนี้ (โดยพิมพ์ไปตามลำดับ)

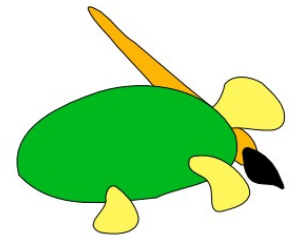
```
x * 5
y = 20
y * x
y = x
y * x
y = y + 1
y * x
```

อธิบายได้หรือไม่ว่าคำสั่ง  $y = y + 1$  ทำอะไร

## ส่วนที่ 2 เต่ายนต์

ในส่วนนี้เราจะทดลองการสั่งงานคอมพิวเตอร์ให้สร้างสรค้งงานศิลปะแบบง่าย ๆ เพื่อฝึกการคิดแบบเป็นขั้นตอน เราจะได้ทดลองวาดภาพโดยการควบคุม เต่ายนต์ ให้นึกถึงว่าเต้านี้เป็นเต่าที่ถือปากกา และเราจะสามารถสั่งเต่าให้เดินไปมาพร้อม ๆ กับลากปากกาดังกล่าว สร้างสรค้งรูปออกมาได้

สิ่งสำคัญที่สุดของการทำแบบฝึกหัดนี้ก็คือ **อย่ากลัวที่จะผิดพลาด!!!**

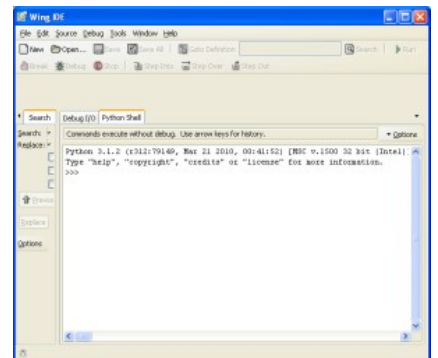


### 2.1 เปิดหน้าต่างเต่า

เราจะทำงานที่ส่วน Python Shell เป็นหลัก อย่างไรก็ตามส่วนนี้มีขนาดเล็กเมื่อเทียบกับหน้าจอทั้งหมดของ Wing IDE 101 ดังนั้นให้ขยายขนาดของส่วนดังกล่าว โดยกดลากที่บริเวณขอบของส่วนดังกล่าว (อาจจะมองไม่เห็นขอบ แต่ให้บริเวณลูกศรเปลี่ยนเป็นสัญลักษณ์ลูกศรสองหัวสำหรับลาก) เมื่อขยายแล้วให้ได้หน้าจอลักษณะดังรูปด้านล่าง

เมื่อปรับดังกล่าวได้แล้วให้ย่อหน้าต่าง Wing IDE ให้มีขนาดประมาณ 1/2 ของหน้าจอ (ถ้าหน้าต่างขยายเต็มที่ย่ำลิมกดปุ่มมุมบนขวาให้หน้าต่างเล็กขยายเต็มจอ ก่อน) ดูขนาดโดยประมาณได้จากรูปหลังจากการเปิดหน้าต่างเต่าแล้ว

เราจะเริ่มใช้งานระบบ Turtle graphics โดยการประกาศเรียกโมดูล turtle ก่อน โดยพิมพ์คำสั่งด้านล่างลงใน Python shell



```
from turtle import *
```

จากนั้นให้สั่ง สร้าง เต่า แล้วกำหนดค่าให้ตัวแปร t อ้างถึงเต่าตัวนั้น โดยพิมพ์คำสั่ง

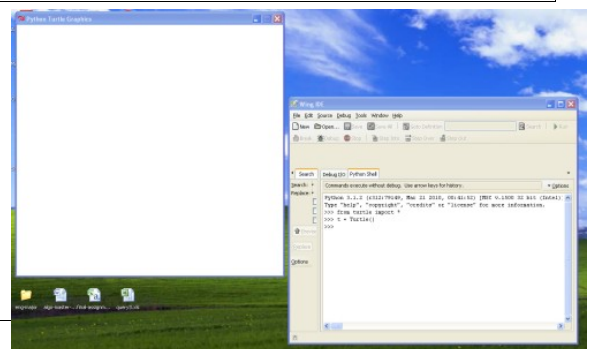
```
t = Turtle()
```

จะมีหน้าต่างชื่อ Python Turtle Graphics ปรากฏขึ้น หน้าต่างนี้อาจจะใช้เวลาเล็กน้อยก่อนจะปรากฏขึ้น และอาจจะอยู่ด้านหลังหน้าต่าง Wing IDE ให้กดเลือกเพื่อเลื่อนขึ้นมาด้านหน้า

จากนั้นปรับขนาดของหน้าต่างทั้งสองหน้าต่างให้เห็นได้ทั้งคู่พร้อม ๆ กัน เพราะว่าเราจะสั่งงานที่ Wing IDE แต่ภาพวาดของเต่าจะแสดงที่หน้าต่างใหม่นี้ ตัวอย่างแสดงดังรูปด้านขวา

เมื่อได้ดังนี้แล้ว ลองพิมพ์

```
t.forward(100)
```



จะเห็นเส้นลากไปทางขวา (ผลจากการลากปากกา) พร้อมเครื่องหมายลูกศร (แสดงถึงตำแหน่งและทิศทางการหันของเต่า) นั่นคือเราพร้อมจะทดลองกับเต่ายนต์แล้ว

หมายเหตุ: ระหว่างการทดลอง ถ้าจำเป็นต้องปิดโปรแกรม Wing IDE หรือปิดเครื่อง ให้สั่งคำสั่งสองคำสั่งนี้ Python Shell เพื่อเปิด

หน้าต่างเต่าก่อนจะทำงานได้

```
from turtle import *  
t = Turtle()
```

## 2.2 ทดลองคำสั่ง

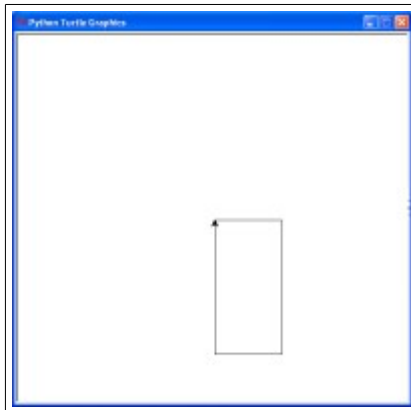
จากเส้นดังกล่าว ทดลองสั่งคำสั่งต่อไปนี้ทีละคำสั่ง และสังเกตพฤติกรรมของเต่ายนต์

```
t.right(90)  
t.forward(200)
```

การสั่ง forward และ right ทำให้เต่ายนต์ทำอะไร?

คำสั่ง **forward** และ คำสั่ง **right** เป็นคำสั่งที่สั่งให้เต่ายนต์ เวลาเราพูดถึงคำสั่งเราจะละส่วน t. เอาไว้ แต่เมื่อจะสั่งเราต้องอ้างถึงเต่ายนต์ที่เราจะสั่งด้วย

ทดลองสั่งให้เต่ายนต์เคลื่อนที่จนกระทั่งได้รูปสี่เหลี่ยมผืนผ้าขนาด 100 x 200 ดังตัวอย่างด้านล่าง



บันทึกคำสั่งที่สั่งลงด้านล่างนี้

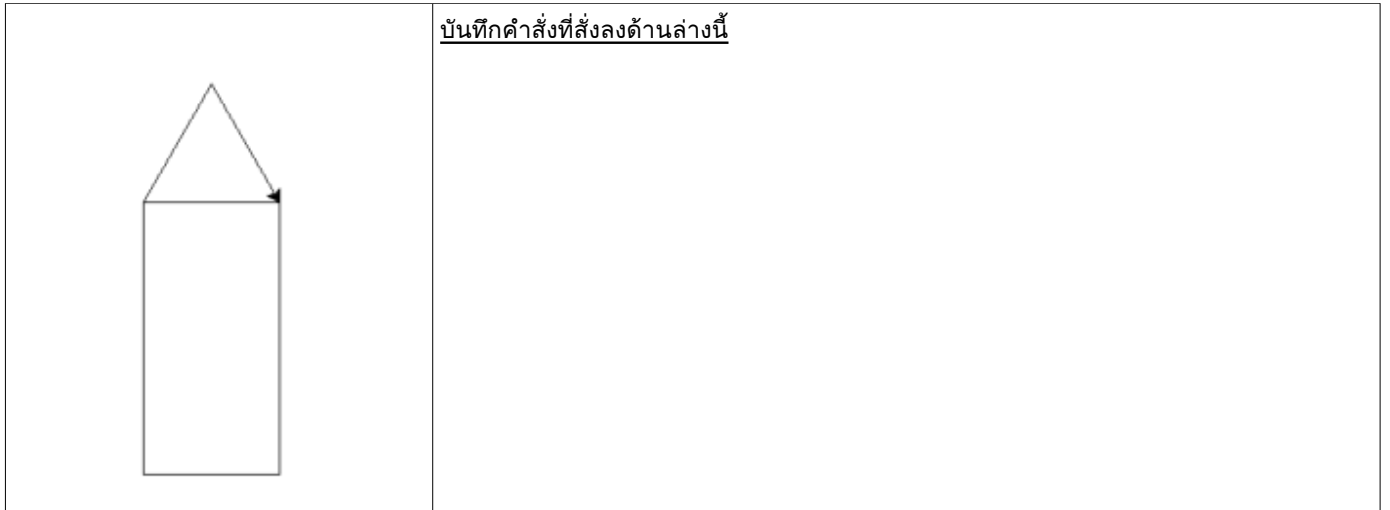
**ถ้าสั่งผิดพลาด:** อย่าเพิ่งกลัวว่าเต่ายนต์จะวาดผิด ทำให้ได้เส้นเกินมา สำหรับแบบฝึกหัดนี้ให้ถือว่า เส้นเกินดีกว่าเส้นขาด

ถ้าเราสั่ง forward ใกล้เคียง สามารถสั่ง backward(ระยะ ทาง) ให้เต่ายนต์เดินถอยหลังย้อนกลับได้ ในทำนองเดียวกัน เราสามารถสั่ง left(มุม) เพื่อให้เต่าหันซ้ายได้ หรือถ้าต้องการยกเลิกทุกอย่าง สามารถสั่ง clear() ให้ลบทุกอย่างทิ้งได้ (แต่เต่ายนต์ยังอยู่ที่เดิม) ถ้าต้องการให้เต่ายนต์กลับมาที่จุดเริ่มต้นสั่ง home()

อย่าลืมนะว่าจะสั่งคำสั่งเหล่านี้ต้องสั่งผ่านทางเต่ายนต์ เช่น สั่ง t.backward(100) เป็นต้น

ทดลองสั่งให้เต่ายนต์เคลื่อนที่ต่อจนกระทั่งได้รูปบ้านดังตัวอย่างด้านล่าง (หมายเหตุ:กว่าจะทำรูปดังกล่าวเป็นตัวอย่างได้ ผู้ทำต้องลบไปหลายรอบเหมือนกัน แต่อย่าลืมนะ! จะมีเส้นเกินมาบ้างก็ไม่เป็นไร)

คำใบ้ สามเหลี่ยมด้านเท่ามีมุมภายใน 60 องศา (แต่ไม่จำเป็นต้องสร้างสามเหลี่ยมด้านเท่าก็ได้)



บันทึกคำสั่งที่สั่งลงด้านล่างนี้

## 2.2.1 สรุปคำสั่ง

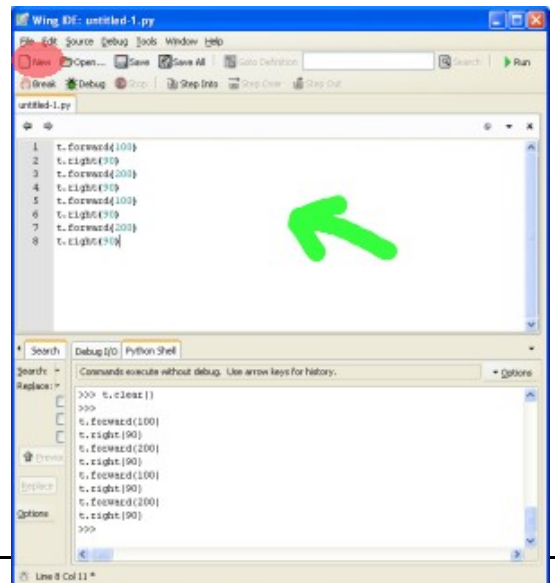
- **forward(ระยะทาง)** และ **backward(ระยะทาง)** ---- ให้เต่ายนต์เคลื่อนที่ไปด้านหน้า / ด้านหลัง
- **left(มุม)** และ **right(มุม)** ---- ให้เต่ายนต์หันซ้าย / หันขวา
- **home()** ---- ให้เต่ายนต์กลับบ้าน (กลับไปจุดเริ่มต้น และหันหัวไปทางขวานกับแกน x)
- **clear()** ---- ลบภาพทั้งหมด

## 2.3 การป้อนคำสั่งที่สะดวก (และปลอดภัย) ขึ้น

การป้อนคำสั่งทีละคำสั่งทำให้เมื่อเราทำขั้นตอนยาว ๆ แล้วผิดพลาด การแก้ไขทำได้ยากมาก เราจะเปลี่ยนเป็นเขียนคำสั่งหลาย ๆ คำสั่งแล้วค่อยนำไปสั่งใน Python Shell การเขียนในลักษณะนี้ทำให้ถ้าเราสั่งผิดพลาดเราสามารถแก้ไขได้ง่าย โดยไม่ต้องนั่งพิมพ์ทุกอย่างใหม่

ให้กดปุ่ม New ที่มุมซ้ายของทูลบาร์ (เน้นด้วยวงกลมสีแดง) จากนั้นย่อขนาดส่วน Python Shell ให้เตี้ยลง (หรือจะขยายหน้าจอ Wing IDE ให้สูงขึ้นก็ได้)

ในส่วนด้านบนเรียกว่าส่วน Editor (เอดิเตอร์) จะเป็นส่วนที่เราป้อนคำสั่งได้หลาย ๆ คำสั่งโดยยังไม่ถูกนำไปทำงาน (ลูกศรสีเขียวชี้อยู่) เมื่อเราเขียนจนพอใจแล้ว ให้ใช้เมาส์ลากเลือกส่วนดังกล่าว (จะเห็นเป็นสีเหลือง ๆ) จากนั้น Copy (โดยกด Ctrl-C) แล้วไปที่ส่วน Python Shell แล้วกด Paste (กด Ctrl-V) เพื่อสั่งชุดคำสั่งดังกล่าว



## 2.4 มาวาดรูปกัน!

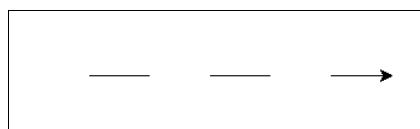
ก่อนที่เราจะสามารถวาดรูปที่ซับซ้อนได้ เราจะต้องรู้จักคำสั่งในการควบคุมเต่ายนต์อีกสองคำสั่งคือ

- **penup()** ให้เต่ายนต์ยกปากกา
- **pendown()** ให้เต่ายนต์วางปากกา

ให้ลบทุกอย่างทิ้ง โดยสั่ง

```
t.home()
t.clear()
```

จากนั้นลองใช้คำสั่งยกและวางปากกาสั่งให้เต่ายนต์วาดรูปด้านล่างนี้



คำสั่งที่ใช้คือ

## 2.5 สร้างคำสั่งใหม่

*อย่ากลัวสี่เหลี่ยมจัตุรัส! ถ้ามีคำสั่งที่สั่งได้ง่าย ๆ ก็คงดีนะสิ*

เราสามารถสอน Python ให้รู้จักคำสั่งใหม่ได้ หลักการคร่าว ๆ ก็คือ เราสามารถรวมชุดของคำสั่งที่ใช้บ่อย ๆ มานิยามเป็นคำสั่งใหม่ เพื่อให้ใช้งานได้

พิมพ์ส่วนของโปรแกรมนี้ในหน้า Editor ก่อนจะคัดลอกมาที่ Python Shell

```
def square():  
    t.forward(100)  
    t.right(90)  
    t.forward(100)  
    t.right(90)  
    t.forward(100)  
    t.right(90)  
    t.forward(100)  
    t.right(90)
```

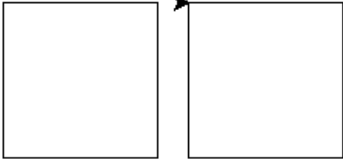
ในการพิมพ์ให้ระวังเครื่องหมาย : ที่ท้ายบรรทัดแรก และระวังย่อหน้าให้ตรงกันด้วย สังเกตว่าเมื่อเราขึ้นบรรทัดใหม่ถัดจากบรรทัดแรก Wing IDE จะพยายามจัดย่อหน้าให้โดยอัตโนมัติ

ให้คัดลอกส่วนดังกล่าวมาใส่ใน Python Shell ถ้าไม่มีข้อผิดพลาดแสดงว่าเรานิยามคำสั่ง `square()` แล้ว ลองสั่งโดยพิมพ์

```
square()
```

เตาจะวาดรูปสี่เหลี่ยมจัตุรัสขนาด 100 x 100

ลองสั่งให้เตาวาดรูปตั้งรูปด้านล่าง โดยใช้คำสั่ง `square()`

	<p><u>บันทึกคำสั่งที่ใช้ด้านล่าง</u></p>
---	--

คำสั่ง `square` ที่เราสร้างขึ้นมานั้น ทำงานเดิมซ้ำกัน 4 ครั้ง ในภาษา Python สามารถเขียนให้สั้นลงได้ เป็นดังด้านล่าง

```
def square():  
    for x in range(4):  
        t.forward(100)  
        t.right(90)
```

ระวังย่อหน้าให้ตรงกัน และอย่าลืมเครื่องหมาย : หลังบรรทัด `def` และ `for`

ให้เขียนส่วนของโปรแกรมห้างต้นในส่วน Editor จากนั้นคัดลอกมาที่ส่วน Python Shell และทดลองสั่ง `square()`

คำสั่ง `for` ที่ใช้ในรูปแบบนี้ข้างต้น จะทำให้คำสั่งที่เขียนในส่วนที่ย่อนหน้าเข้าไปทำซ้ำเป็นจำนวนรอบเท่ากับค่า ที่ใส่ในฟังก์ชัน `range` ในตัวอย่างข้างต้นคือ 4 เราจะเรียนรายละเอียดมากกว่านี้ในการบรรยายครั้งที่ 5





พิจารณาผลลัพธ์ที่ได้ จากแนวคิดดังกล่าว เราสามารถสร้างคำสั่งวาดวงกลมได้ดังนี้

```
def circle():
    for x in range(36):
        t.forward(20)
        t.right(10)
```

### ส่วนที่ 3 ทดลองเขียนโปรแกรมทายเลข

ในส่วนนี้เราจะทดลองพัฒนาโปรแกรมเกมทายตัวเลข โดยเราจะเขียนโปรแกรมในส่วน Editor จากนั้นจะเรียกให้โปรแกรมทำงานโดยตรง (โดยไม่ต้องตัดออกไปวางใน Python Shell เมื่อในส่วนก่อน) การทดลองในส่วนนี้มีเป้าหมายเพื่อฝึกให้นิสิตคุ้นเคยกับ Wing IDE 101 และภาษา Python เพื่อลดความผิดพลาดที่จะเกิดขึ้นในการเขียนเมื่อเรียนต่อ ๆ ไป

เกมทายตัวเลขมีการทำงานดังนี้

- เกมจะสุ่มตัวเลขจาก 1 - 100
- ให้ผู้ใช้ทาย เกมจะใบ้โดยการบอกว่ามากไป หรือน้อยไป
- เมื่อผู้ใช้ทายถูกจะจบเกม

เราจะเริ่มเขียนโปรแกรมโดยเขียนไปทีละส่วน และค่อย ๆ เพิ่มความสามารถให้กับโปรแกรมทีละขั้น

#### 3.1 เริ่มต้น: บอกว่ามากไปหรือน้อยไป

เริ่มเขียนโดยกดปุ่ม New บนทูลบาร์ จากนั้นพิมพ์โปรแกรมด้านล่างนี้

```
answer = 53
guess = int(input("Guess the number between 1-100: "))
if guess > answer:
    print("Too high")
if guess < answer:
    print("Too low")
if guess == answer:
    print("Correct answer!")
```

สั่งให้โปรแกรมทำงานโดยกดปุ่ม Run บนทูลบาร์ (เป็นรูปสามเหลี่ยมสีเขียว) โปรแกรมจะทำงานในส่วน Python Shell เมื่อโปรแกรมทำงานเสร็จเราจะกลับมาที่เครื่องหมาย >>> ที่พร้อมจะรับคำสั่งต่อไป ถ้าต้องการสั่งให้โปรแกรมทำงานอีกครั้ง ให้กดปุ่ม Run เช่นเดิม

ข้อควรระวัง: ดูเครื่องหมาย : ให้ครบ ดูวงเล็บและเครื่องหมายคำพูด (") ให้เข้าคู่

ลองพิจารณาโปรแกรมข้างต้น แล้วตอบคำถามต่อไปนี้

คำถาม	คำตอบ
ต้องทายเท่าใดโปรแกรมถึงจะตอบว่า Correct answer!	
ต้องทายเท่าใดโปรแกรมถึงจะตอบว่า Too high	
ต้องทายเท่าใดโปรแกรมถึงจะตอบว่า Too low	
คำสั่ง if ทำอะไร?	

สังเกตว่าในโปรแกรมข้างต้น เรายังไม่ทราบวิธีการสุ่มเลข เราเลยกำหนดให้คำตอบมีค่าเป็น 53 ไว้ก่อน

#### 3.2 ทายหลายครั้ง

โปรแกรมที่เขียนแล้วนั้นทำงานแค่รอบเดียว เราจะแก้โปรแกรมให้ทำงานหลายรอบ ซ้ำ ๆ กัน โดยใช้โครงสร้าง while

แก้โปรแกรมให้เป็นตามด้านล่าง ระวังเรื่องย่อหน้า, เครื่องหมาย และตัว T ใหญ่ ในคำว่า True

```
answer = 53
while True:
    guess = int(input("Guess the number between 1-100: "))
    if guess > answer:
        print("Too high")
    if guess < answer:
        print("Too low")
    if guess == answer:
        print("Correct answer!")
```

สังเกตว่าลักษณะของโปรแกรมที่เราเขียนจะมีการย่อหน้าเป็นชั้น ๆ ระดับของการย่อหน้ามีความหมายบอกถึงขอบเขตของโครงสร้างที่ครอบคลุมย่อหน้า นั้น ในตัวอย่างข้างต้นคือโครงสร้าง while

เราจะได้เรียนเกี่ยวกับลักษณะการควบคุมของโครงสร้าง while ต่อไป ในที่นี่ให้เข้าใจว่า ถ้าเงื่อนไขที่ตามหลังคำว่า while เป็นจริง คำสั่งที่อยู่ภายในการควบคุมของโครงสร้าง while ก็จะถูกทำซ้ำไปเรื่อย ๆ ในที่นี่คำว่า True (จริง) แทนเงื่อนไขที่เป็นจริงเสมอ

ลองสั่งให้โปรแกรมทำงาน และทดลองป้อนค่าต่าง ๆ

เมื่อสั่งให้โปรแกรมทำงาน เราจะพบว่าคำสั่งภายในโครงสร้าง while ถูกทำงานซ้ำไปเรื่อย ๆ ไม่มีวันสิ้นสุด แม้ว่าเราจะพิมพ์คำตอบที่ถูกต้องแล้วก็ตาม

เราสามารถใส่คำสั่ง break ควบคู่กับโครงสร้าง while เพื่อให้การทำงานซ้ำจบลง เราจะแก้โปรแกรมให้เป็นดังด้านล่าง

ทดลองเรียกให้โปรแกรมทำงานและทดลองทายค่าต่าง ๆ

```
answer = 53
while True:
    guess = int(input("Guess the number between 1-100: "))
    if guess > answer:
        print("Too high")
    if guess < answer:
        print("Too low")
    if guess == answer:
        print("Correct answer!")
        break
```

ข้อควรระวัง: ระวังการเว้นย่อหน้าของคำสั่ง break ให้ถูกต้อง

ในภาษา Python การเว้นย่อหน้ามีความสำคัญมาก ทดลองแก้โปรแกรมโดยเลื่อนย่อหน้าของคำสั่ง break ให้เป็นดังโปรแกรมด้านล่าง

```
answer = 53
while True:
    guess = int(input("Guess the number between 1-100: "))
    if guess > answer:
        print("Too high")
    if guess < answer:
        print("Too low")
    if guess == answer:
        print("Correct answer!")
    break
```

จากนั้นทดลองสั่งให้โปรแกรมทำงาน และป้อนการทายค่าด้วยค่าต่าง ๆ เช่น 10, 53 หรือ 80

การทำงานของโปรแกรมเป็นอย่างไร อธิบายได้หรือไม่ว่าเหตุใดโปรแกรมหลังจากที่แก้ตำแหน่งของคำสั่ง break แล้วจึงทำงานแตกต่างไป

### 3.3 สุ่มค่าตัวเลข

ในการสุ่มค่าตัวเลข เราจะใช้ฟังก์ชัน randint จาก โมดูล random ดังนั้นที่ต้นโปรแกรมของเรา เราจะต้องประกาศขอใช้โมดูลดังกล่าวเสียก่อน โดยสั่ง

```
import random
```

หลังจากบรรทัดดังกล่าว เราจะสามารถเรียกใช้ทุก ๆ ฟังก์ชันที่อยู่ในโมดูล random ได้ โดยเขียนในลักษณะ random.ชื่อฟังก์ชัน

ฟังก์ชัน randint รับค่าสองค่า คือค่าต่ำสุดและค่าสูงสุดของตัวเลขที่ต้องการสุ่ม เราแก้ไขโปรแกรมของเราเพื่อเรียกใช้ฟังก์ชันดังกล่าวได้ดังนี้

```
import random

answer = random.randint(1,100)
while True:
    guess = int(input("Guess the number between 1-100: "))
    if guess > answer:
        print("Too high")
    if guess < answer:
        print("Too low")
    if guess == answer:
        print("Correct answer!")
        break
```

### 3.4 นับจำนวนครั้งการทายเลข

เราจะเก็บจำนวนรอบของการทายไว้ในตัวแปรชื่อ count โดยตัวแปรนี้จะมีค่าเริ่มต้นเป็น 0 และจะค่อย ๆ เพิ่มค่าขึ้นเมื่อเราทำงาน

```
import random

answer = random.randint(1,100)
count = 0
while True:
    count = count + 1
    guess = int(input("Guess the number between 1-100: "))
    if guess > answer:
        print("Too high")
    if guess < answer:
        print("Too low")
    if guess == answer:
        print("Correct answer!")
        break
print("You guessed",count,"times.")
```

บรรทัด count = count + 1 สามารถเขียนย่อให้เข้าใจได้ง่ายขึ้นเป็น

```
count += 1
```

(เพิ่มเติม ไม่จำเป็นต้องทำ) ถ้าเราต้องการให้โปรแกรมพิมพ์ข้อความว่า "You did very well." ถ้าผู้ใช้สามารถทายเลขได้ภายใน 5 ครั้ง จะต้องเพิ่มส่วนของโปรแกรมใดเข้าไปตอนท้ายโปรแกรมข้างต้น?