Programming Fundamentals I Lab.

1. เริ่มต้นเขียนโปรแกรมด้วยภาษา Python

ชื่อ_____ รหัสนิสิต_____

ในปฏิบัติการนี้ คุณจะได้ทำความคุ้นเคยกับสภาพแวดล้อมในการเขียนโปรแกรมด้วยภาษา Python ชนิดข้อมูลและการดำเนินการพื้นฐาน ตัวแปร การรับค่าและการแสดงข้อมูล หากเครื่องคอมพิวเตอร์ของคุณยังไม่ได้ลงโปรแกรม Python คุณสามารถดาวน์โหลดได้ฟรีจากเว็บ `https://www.python.org` โดยเลือกลง Python 3

1.1 เริ่มต้นเขียนโปรแกรมด้วยภาษา Python

เปิดโปรแกรม IDLE ในเครื่องคอมพิวเตอร์ จะมีหน้าต่าง Python Shell ปรากฏขึ้นมา ให้คุณคลิกที่ `File -> New File` จะมีหน้าต่างใหม่ปรากฏขึ้น พิมพ์ข้อความต่อไปนี้ในหน้าต่างใหม่นั้น

```
print('Hello World!')
```

(คำเตือน: ในการพิมพ์ให้ระวังตัวอักษรเล็กใหญ่ และการเว้นวรรคให้ละเอียด เนื่องจากทุกอย่างมีผลต่อการทำงาน)

จากนั้นให้คลิก `File -> Save` ตั้งชื่อไฟล์ว่า `lab1.py` โดยเลือกพื้นที่เก็บเป็น Desktop และคลิก `Save` แถบ title ของหน้าต่างที่เขียนโค้ดจะเปลี่ยนเป็นชื่อไฟล์ จากนั้นให้คลิกที่ `Run -> Run Module` บันทึกข้อความที่แสดงในหน้าต่าง Python Shell ลงในกรอบด้านล่างนี้

```
...
>>> ================================ RESTART ================================
>>>


```

คราวนี้ให้คุณเพิ่มข้อความในไฟล์ `lab1.py` เป็นดังนี้

```
print('Hello World!')
print('I love "Python".')
```

เมื่อเซฟไฟล์และสั่งรันใหม่จะได้ผลเป็นเช่นไร?

หากลองเปลี่ยนบรรทัดที่สองเป็น `print('I love 'Python'.')` (เปลี่ยนเครื่องหมาย double quote (") เป็น single quote (')) เมื่อเซฟแล้วรันจะเกิดผลเช่นไร?

ทดลองเปลี่ยนบรรทัดที่สองเป็นข้อความต่าง ๆ ต่อไปนี้ และบันทึกผลในช่องด้านขวา (บันทึกเฉพาะผลในบรรทัดที่เกี่ยวข้อง, หากมีข้อผิดพลาดให้บันทึกว่ามีข้อผิดพลาด)

`print('I love \'Python\'.')`	
`print("I love \'Python\'.")`	
`print("I love 'Python'.")`	
`print("I love "Python".")`	
`print("I love \"Python\".")`	
`print("I love \\\"Python\\\".")`	

ถ้าอยากให้ในหน้าต่าง Python Shell แสดงข้อความดังนี้ จงเขียนโค้ดที่แสดงผลได้ถูกต้องตามต้องการ

```
"Logic will get you from 'A' to 'B'.
Imagination will take you everywhere."
-- Albert Einstein
```

1.2 ชนิดข้อมูลและตัวดำเนินการ

ลบคำสั่งในไฟล์ `lab1.py` ทั้งหมด ลองทดสอบคำสั่งต่อไปนี้และบันทึกผลในช่องด้านขวา

`print('Ha' + 'wai')`	
`print('Ha ' + 'Ha ' + 'Ha ')`	
`print('Ha' + '300')`	
`print('19' + '300')`	

โดยสรุป เครื่องหมาย + ระหว่างข้อความมีหน้าที่อย่างไร?

ถึงตรงนี้เราจะให้คุณรู้จักการใช้งาน Python Shell สำหรับคำนวณผลลัพธ์ของการดำเนินการระหว่างนิพจน์ต่าง ๆ ให้เปลี่ยนมาที่หน้าต่าง Python Shell ซึ่งมีเครื่องหมาย `>>>` ค้างอยู่ ทดลองพิมพ์คำสั่งดังนี้

```
>>>'Hello ' + 'Python!!!'
```

แล้วกดแป้น `enter` บันทึกผลที่เกิดขึ้นใน Python Shell

ทดลองพิมพ์นิพจน์ต่อไปนี้ใน Python Shell และบันทึกผลที่ได้ในช่องด้านขวา

`'26' + '1590'`	
`26 + 1590`	
`30 * 5`	
`32 / 9`	

จากตรงนี้ จะเห็นว่า `'26' + '1590'` และ `26 + 1590` นั้นได้ผลไม่เหมือนกัน เนื่องจากคอมพิวเตอร์มองเห็น `'26'` และ `'1590'` เป็น *ข้อความ* เช่นเดียวกับข้อความ `'Hello'` แต่ตัวเลข `26` และ `1590` (ที่ไม่อยู่ภายใน quote) นั้น คอมพิวเตอร์จะเห็นเป็นจำนวน จึงสามารถนำค่ามาคำนวณในเชิงตัวเลขได้

เราเรียกข้อมูลต่าง ๆ ที่มีลักษณะเป็นข้อความภายในเครื่องหมาย quote ว่ามีชนิดข้อมูลเป็น *สตริง* (`str`) ในขณะที่ข้อมูลประเภทตัวเลขแบ่งเป็นสองชนิด ได้แก่ *จำนวนเต็ม* (`int`) และ *จำนวนจริง* (`float`)

เราสามารถตรวจสอบชนิดข้อมูลของนิพจน์ใด ๆ ได้โดยใช้คำสั่ง `type()` ให้คุณทดลองพิมพ์คำสั่งต่อไปนี้ในหน้าต่าง Python Shell และกรอกผลที่ได้ในช่องด้านขวา

`type('Hello')`	
`type('1590')`	
`type(30 * 5)`	
`type(32 / 9)`	

ถึงตรงนี้ เราจะให้คุณทำการทดลองเพื่อศึกษาหน้าที่ของตัวดำเนินการจำนวนหนึ่งในภาษา Python โดยให้คุณทดลองแทนค่าของ (`int`), (`float`), และ (`str`) ด้วยข้อมูลตามชนิดที่ถูกต้อง พิจารณาผลลัพธ์ที่ได้ และวิเคราะห์หน้าที่ของตัวดำเนินการนั้น ๆ บันทึกลงในช่องด้านขวา ตามตัวอย่าง

(`str`) + (`str`)	นำข้อความทั้งสองมาต่อกันเป็นข้อความเดียว
(`int/float`) + (`int/float`)	
(`int/float`) - (`int/float`)	
(`int/float`) * (`int/float`)	
(`int/float`) / (`int/float`)	
(`int/float`) ** (`int/float`)	
(`int/float`) // (`int`)	
(`int/float`) % (`int`)	
(`str`) * (`int`)	

คราวนี้ เราจะมาทดลองการใช้ตัวดำเนินการหลายตัวร่วมกัน โดยบันทึกผลจากคำสั่งต่อไปนี้ในช่องด้านขวา

`print(16 + 2 ** 3)`	
`print((16 + 2) ** 3)`	
`print(16 + (2 ** 3))`	

จะเห็นว่าตัวดำเนินการแต่ละตัวมีความสำคัญไม่เท่ากัน โดยหากนิพจน์ปราศจากเครื่องหมายวงเล็บ ตัวดำเนินการที่มีความสำคัญมากจะถูกทำก่อน จงทำการทดลองเพื่อวิเคราะห์ลำดับความสำคัญของตัวดำเนินการเกี่ยวกับตัวเลข ได้แก่ `+`, `-`, `*`, `/`, `**`, `//`, และ `%` บันทึกลำดับความสำคัญที่ได้ในช่องด้านล่าง

1.3 ตัวแปร

เราสามารถสั่งให้คอมพิวเตอร์เก็บข้อมูลไว้ในหน่วยความจำชั่วคราว เพื่อนำไปใช้ต่อภายหลัง โดยการกำหนด *ตัวแปร* ให้ข้อมูลแต่ละตัวที่ต้องการเก็บ คำสั่งในการเก็บข้อมูลใส่ตัวแปรในภาษา Python นั้น ใช้การเรียกชื่อตัวแปร ตามด้วยเครื่องหมายเท่ากับ และตามด้วยค่าหรือนิพจน์ที่ต้องการนำผลลัพธ์ไปเก็บใส่ตัวแปร หลังจากที่มีการกำหนดค่าให้ตัวแปรใดแล้ว เราสามารถนำชื่อตัวแปรนั้นไปใช้งานได้เหมือนกับข้อมูลทั่วไป

ทดลองพิมพ์โปรแกรมต่อไปนี้ใส่ไฟล์ `lab1.py` รันและบันทึกผลที่ได้

```
print('Welcome!!!!')
x = 5
y = 4
print(x)
x = 3
print(x * y)
```

จากการทดลองเมื่อสักครู่นี้ ค่าของตัวแปร `x` ในคำสั่ง `print(x)` และ `print(x * y)` เป็นค่าเดียวกันหรือไม่ เพราะเหตุใด?

ทดลองเปลี่ยนคำสั่ง `x = 3` เป็น `x = x - 3` รันและสังเกตผลที่ได้ คุณคิดว่าคำสั่ง `x = x - 3` มีความหมายอย่างไร?

การตั้งชื่อตัวแปรในภาษา Python นั้นมีเงื่อนไขอยู่ ในส่วนนี้เราจะให้คุณทำการทดลองเพื่อสรุปกฎการตั้งชื่อตัวแปร

ให้คุณลบโค้ดทั้งหมดในไฟล์ `lab1.py` และทดลองคำสั่งสร้างตัวแปรต่อไปนี้ทีละคำสั่ง บันทึกว่าสามารถรันได้หรือไม่

`n = 1`	
`n1 = 1`	
`1n = 1`	
`number = 1`	
`a# = 1`	
`n_1 = 1`	
`_num_10_ = 1`	
`1_num_ = 1`	

ทดลองตั้งชื่อตัวแปรต่าง ๆ จนได้ข้อสรุป บันทึกกฎการตั้งชื่อตัวแปรลงในช่องด้านล่าง

1.4 การรับค่า

ทดลองแก้โปรแกรมในไฟล์ `lab1.py` เป็นโค้ดต่อไปนี้

```
name = input('Please enter your name:')
print('Hello, ' + name)
```

เมื่อรันโปรแกรมจะค้างที่ข้อความ `Please enter your name:` ให้คุณพิมพ์ `Joey` และกดปุ่ม `enter` บันทึกผลที่ได้

ลองรันโปรแกรมอีกครั้ง แต่คราวนี้ให้พิมพ์ `1234` และกดปุ่ม `enter` บันทึกผลที่ได้

คำสั่ง `input` รับข้อมูลเป็นชนิดข้อมูลใด? (คุณอาจทำการทดลองเพิ่มเติมเพื่อความมั่นใจ)

ทดลองแก้โปรแกรมในไฟล์ `lab1.py` เป็นโค้ดต่อไปนี้

```
time = input('Enter time range in weeks:')
print('The time range is ' + (time * 7) + ' days.')
```

รันโปรแกรม เมื่อโปรแกรมค้างที่ข้อความ `Enter time range in weeks:` ให้พิมพ์ 2 และกด `enter` บันทึกผลที่ได้

ให้คุณพิจารณาจากข้อความในโปรแกรม คุณคิดว่าผลลัพธ์ที่ได้ใช่ผลลัพธ์ที่ต้องการจริง ๆ หรือไม่ เพราะเหตุใด?

ให้คุณทดลองแก้โปรแกรมในไฟล์ `lab1.py` ใหม่เป็นโค้ดต่อไปนี้

```
time_str = input('Enter time range in weeks:')
time = int(time_str)
print('The time range is ' + str(time * 7) + ' days.')
```

ทดลองรันและสังเกตผล ใช่ผลลัพธ์ที่ต้องการจริง ๆ หรือไม่?

จากการทดลอง คำสั่ง `int()` และคำสั่ง `str()` มีหน้าที่อย่างไร?

1.5 โจทย์ปัญหา

1. จงเขียนโปรแกรมที่รับชื่อ, ปีเกิดของผู้ใช้, และปีที่ต้องการรู้อายุ และทำการแสดงอายุของผู้ใช้ในปีดังกล่าวโดยมีรูปแบบการแสดงผลตามตัวอย่าง

```
Enter your name: Top
Enter your birth year: 1985
Enter a year: 2029
Hi Top,
In 2029 you will be 44 years old.
```

2. จงเขียนโปรแกรมแปลงอุณหภูมิจากองศา Celsius เป็นองศา Fahrenheit โดยสมการในการแปลงเป็นดังนี้

$$F = \frac{9}{5}C + 32$$

ให้มีรูปแบบการแสดงผลตามตัวอย่างต่อไปนี้

```
Enter the temperature in Celsius: 22.6
The temperature is 72.68 degree Fahrenheit.
```

3. จงเขียนโปรแกรมที่รับราคาสินค้ากับจำนวนเงินที่ลูกค้าจ่าย และแสดงจำนวนเงินที่ต้องทอน พร้อมกับแจกแจงด้วยว่าต้องทอนธนบัตรแต่ละชนิดเป็นจำนวนเท่าใด ดูรูปแบบจากตัวอย่าง

```
Welcome to Change Calculator.
```

```
Price: 247

Amount tendered: 1000

Change: 753
500: 1
100: 2
50: 1
20: 0
10: 0
5: 0
2: 1
1: 1

Thank you.
```