



พื้นฐานการเขียนโปรแกรม
คอมพิวเตอร์ด้วยภาษาไพธอน

Table of Contents

หน้า 1
Python คืออะไร?

หน้า 2
แนะนำโครงสร้างของภาษา

หน้า 9
Variable and Data Type

หน้า 14
Input and Output

หน้า 16
Operators

หน้า 22
Conditional Statement

หน้า 30
Iteration Statement

หน้า 35
แบบฝึกหัด

หน้า 36
แหล่งเรียนรู้เพิ่มเติม

Python คืออะไร?

Python นั้นเป็นหนึ่งในภาษาที่ใช้สำหรับเขียนโปรแกรมเพื่อสั่งงานให้คอมพิวเตอร์ทำตามที่มีมนุษย์ต้องการ โดย Python เป็นภาษาที่จัดอยู่ในภาษาระดับสูง สามารถเขียนและอ่านเข้าใจได้ง่าย ดังนั้นจึงเหมาะกับผู้ที่เพิ่มเริ่มเรียนรู้การเขียนโปรแกรม

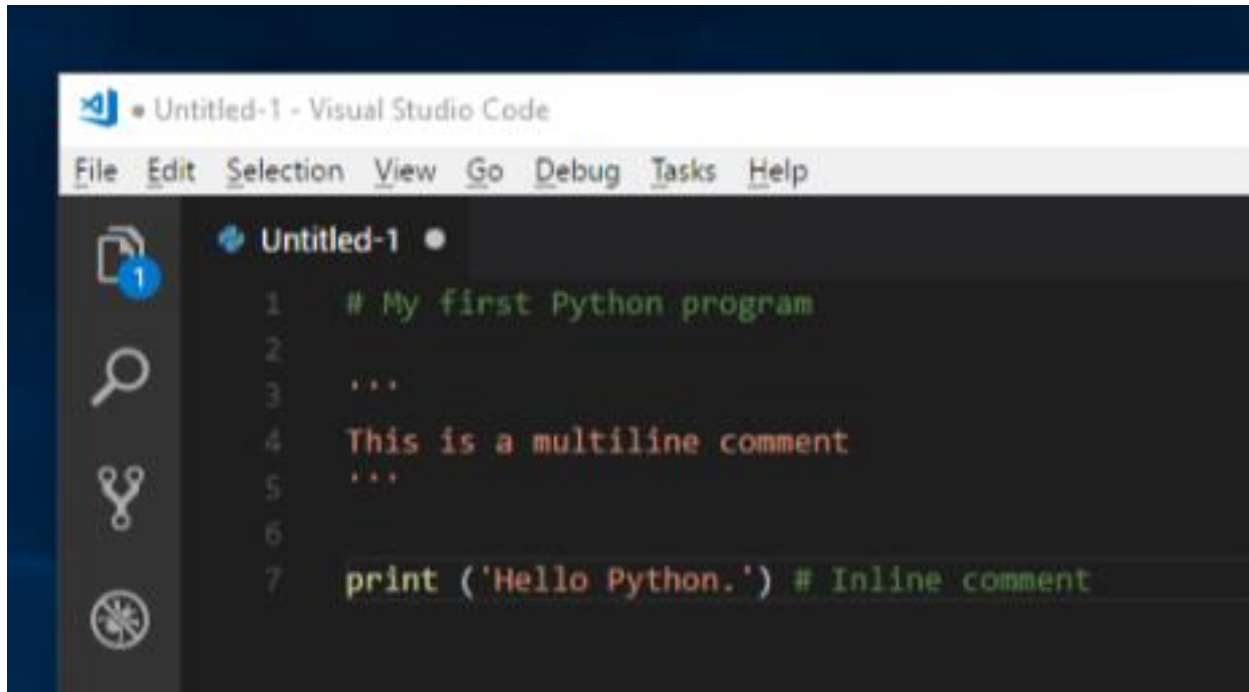


Welcome Python Developers

[See here how Microsoft is making our platform better for Python developers](#)



แนะนำโครงสร้างของภาษา

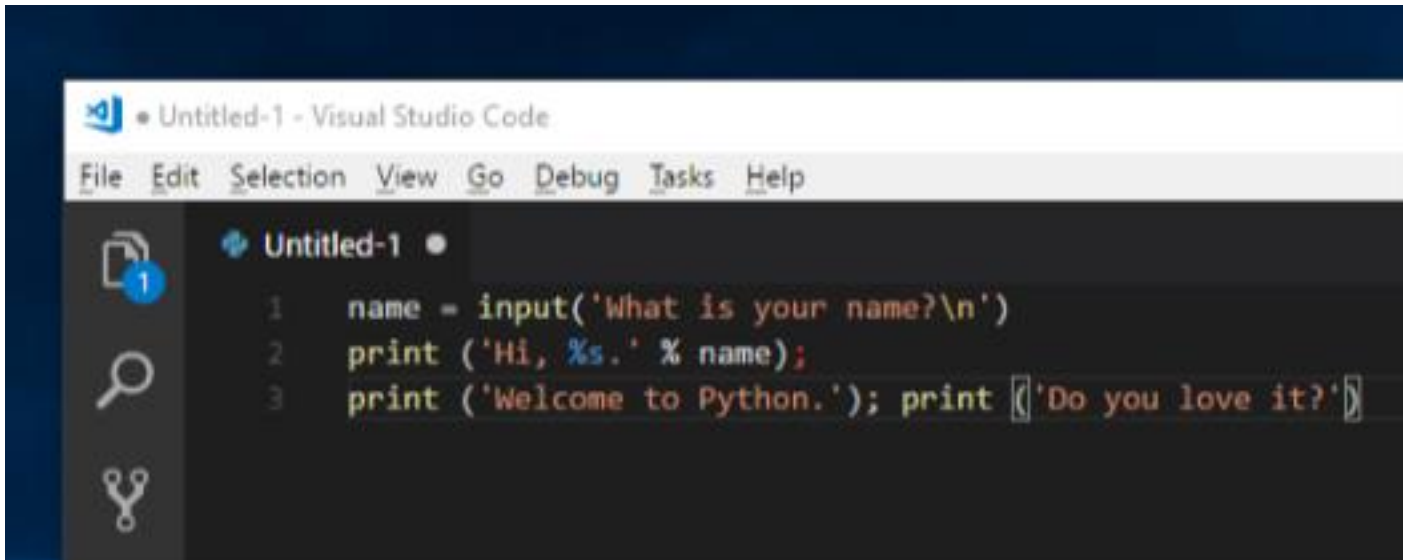


```
1  # My first Python program
2
3  ...
4  This is a multiline comment
5  ...
6
7  print ('Hello Python.') # Inline comment
```

Comment

คอมเมนต์ในภาษา Python นั้นเริ่มต้นด้วยเครื่องหมาย # คอมเมนต์สามารถเริ่มต้นที่ตำแหน่งแรกของบรรทัดและหลังจากนั้นจะประกอบไปด้วย Whitespace หรือโค้ดของโปรแกรม หรือคำอธิบาย ซึ่งโดยทั่วไปแล้วคอมเมนต์มักจะใช้สำหรับอธิบายซอสโค้ดที่เราเขียนขึ้นและมันไม่มีผลต่อการทำงานของโปรแกรม

ในตัวอย่าง เราได้คอมเมนต์สามแบบด้วยกัน แบบแรกเป็นการคอมเมนต์แบบ single line แบบที่สองเป็นการคอมเมนต์แบบ multiline line และแบบสุดท้ายเป็นการคอมเมนต์แบบ inline หรือการคอมเมนต์ภายในบรรทัดเดียวกัน



The screenshot shows the Visual Studio Code interface. The title bar reads 'Untitled-1 - Visual Studio Code'. The menu bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Debug', 'Tasks', and 'Help'. The left sidebar shows a file explorer with 'Untitled-1' selected. The main editor area displays the following Python code:

```

1  name = input('What is your name?\n')
2  print ('Hi, %s.' % name);
3  print ('Welcome to Python.');
```

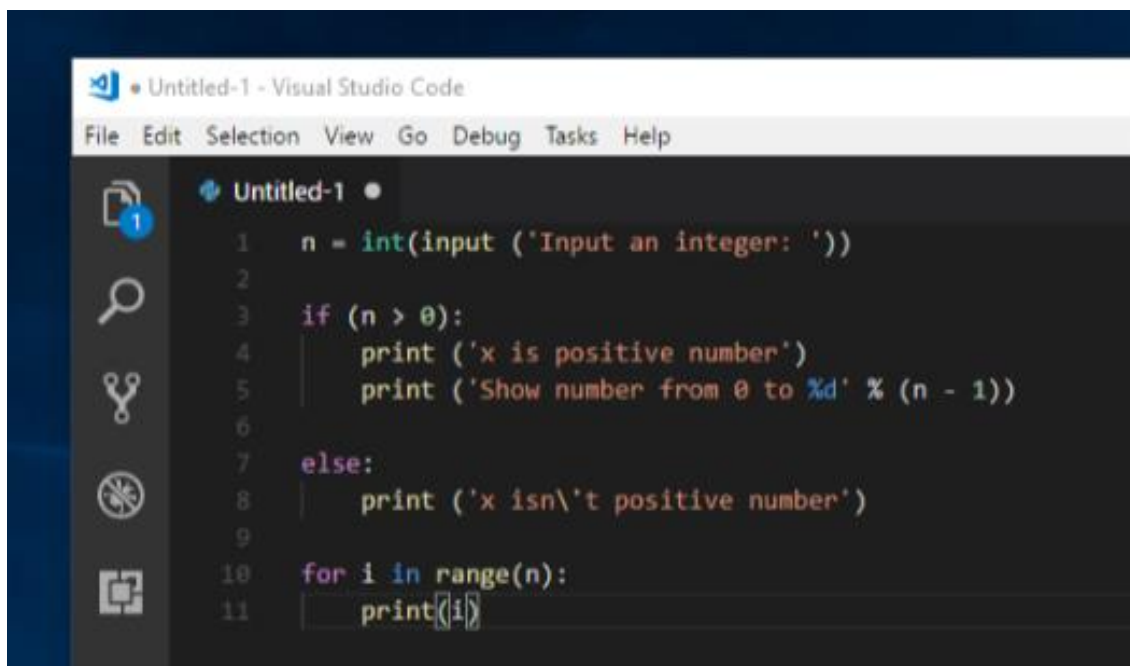
Statement

คำสั่งการทำงานของโปรแกรม แต่ละคำสั่งในภาษา Python นั้นจะแบ่งแยกด้วยการขึ้นบรรทัดใหม่ ซึ่งจะแตกต่างจากภาษา C และ Java ซึ่งใช้เครื่องหมายเซมิโคลอนสำหรับการจบคำสั่งการทำงาน แต่อย่างไรก็ตาม ในภาษา Python นั้น คุณสามารถมีหลายคำสั่งในบรรทัดเดียวกันได้โดยใช้เครื่องหมายเซมิโคลอน ;

ในตัวอย่าง เรามี 4 คำสั่งในโปรแกรม สองบรรทัดแรกเป็นคำสั่งที่ใช้บรรทัดใหม่ในการจบคำสั่ง ซึ่งเป็นแบบปกติในภาษา Python และบรรทัดสุดท้ายเรามีสองคำสั่งในบรรทัดเดียวกันที่คั่นด้วยเครื่องหมาย ; สำหรับการจบคำสั่ง

Indentation

ในภาษา Python นั้นใช้ Whitespace และ Tab สำหรับกำหนดบล็อกของโปรแกรม เช่น คำสั่ง If Else For หรือการประกาศฟังก์ชัน ซึ่งคำสั่งเหล่านี้เป็นคำสั่งแบบบล็อก โดยจำนวนช่องว่างที่ใช้ขึ้นต้องเท่ากัน มาดูตัวอย่างของบล็อกคำสั่งในภาษา Python

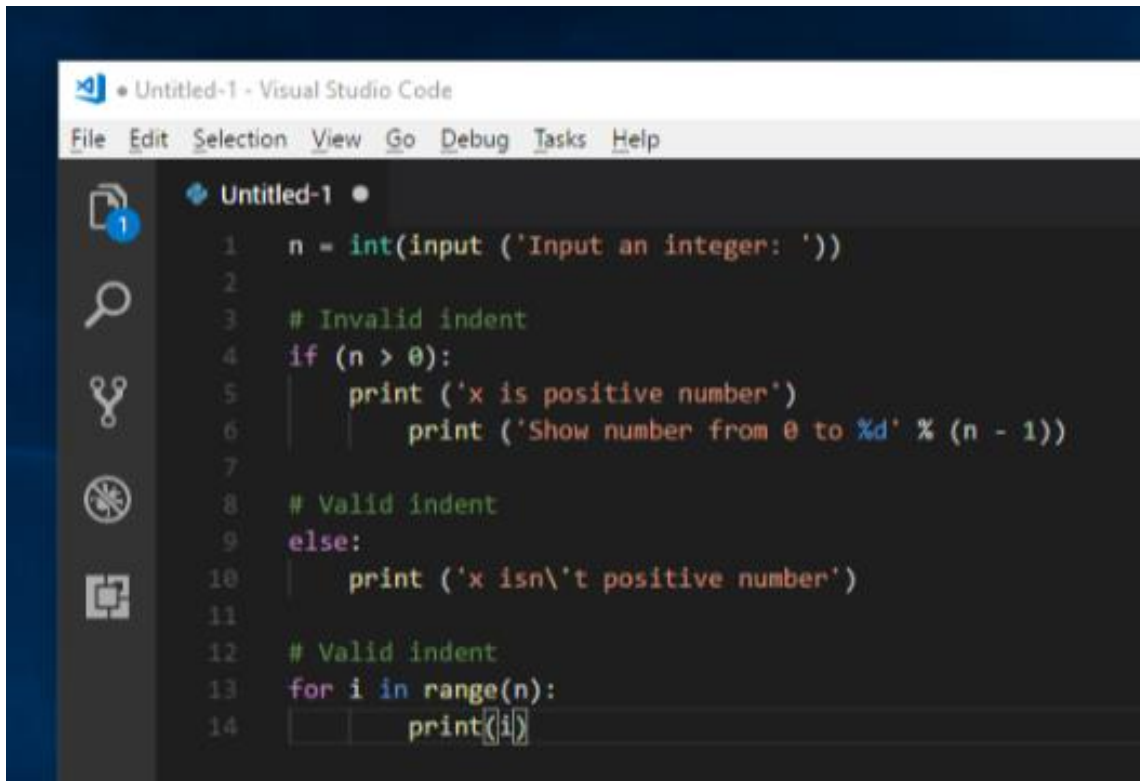


```

Untitled-1 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

Untitled-1
1  n = int(input('Input an integer: '))
2
3  if (n > 0):
4      print('x is positive number')
5      print('Show number from 0 to %d' % (n - 1))
6
7  else:
8      print('x isn\'t positive number')
9
10 for i in range(n):
11     print(i)
  
```

ในตัวอย่าง เป็นบล็อกของโปรแกรมจาก 3 คำสั่ง ในคำสั่งแรกคือ If ในบล็อกนี้มีสองคำสั่งย่อยอยู่ภายใน ที่หัวของบล็อกนั้นจะต้องมีเครื่องหมาย : กำหนดหลังคำสั่งในการเริ่มต้นบล็อกเสมอ อีกสองบล็อกสุดท้ายนั้นเป็นคำสั่ง Else และ For ซึ่งมีหนึ่งคำสั่งย่อยอยู่ภายใน ในภาษา Python นี้เข้มงวดกับช่องว่างภายในบล็อกมาก นั่นหมายความว่าทุกคำสั่งย่อยภายในบล็อกนั้นต้องมีจำนวนช่องว่างเท่ากันเสมอ



The screenshot shows the Visual Studio Code editor interface. The title bar reads "Untitled-1 - Visual Studio Code". The menu bar includes "File", "Edit", "Selection", "View", "Go", "Debug", "Tasks", and "Help". The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The main editor area displays a Python script with the following code:

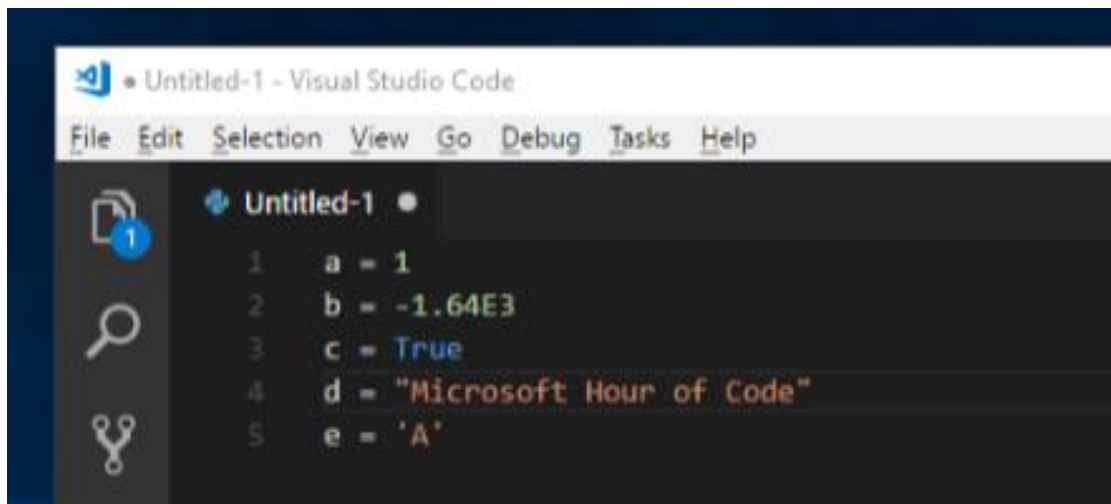
```
1  n = int(input ('Input an integer: '))
2
3  # Invalid indent
4  if (n > 0):
5      print ('x is positive number')
6      print ('Show number from 0 to %d' % (n - 1))
7
8  # Valid indent
9  else:
10     print ('x isn\'t positive number')
11
12 # Valid indent
13 for i in range(n):
14     print(i)
```

Comments within the code indicate indentation issues: "# Invalid indent" on line 3 and "# Valid indent" on lines 8 and 12.

ในตัวอย่าง นี่เป็นตัวอย่างการใช้งานช่องว่างที่ถูกต้องและไม่ถูกต้องภายในบล็อก ไส้คำสั่ง If นั้นไม่ถูกเพราะทั้งสองคำสั่งมีจำนวนช่องว่างที่ไม่เท่ากัน สำหรับในคำสั่ง Else และ For นั้นถูกต้อง

Literals

ในการเขียนโปรแกรม Literal คือเครื่องหมายที่ใช้แสดงค่าของค่าคงที่ในโปรแกรม ในภาษา Python นั้นมี Literal ของข้อมูลประเภทต่างๆ เช่น Integer Floating-point number และ String หรือแม้กระทั่งตัวอักษรและ boolean นี่เป็นตัวอย่างของการกำหนด Literal ให้กับตัวแปรในภาษา Python

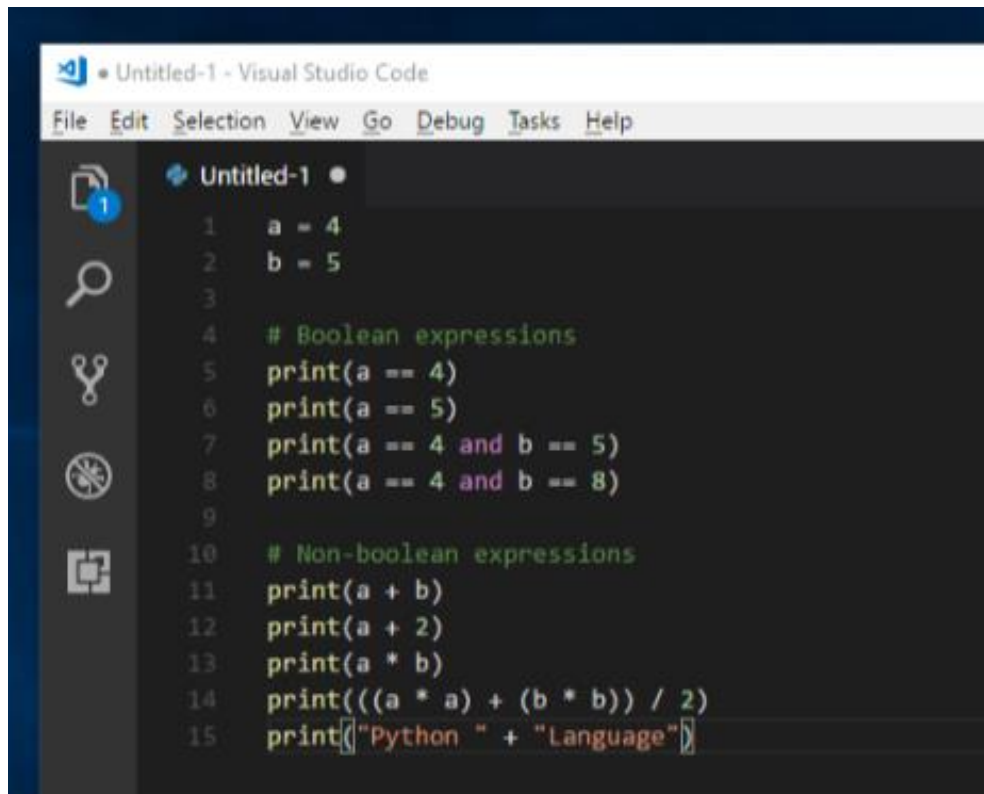


```
1 a = 1
2 b = -1.64E3
3 c = True
4 d = "Microsoft Hour of Code"
5 e = 'A'
```

ในตัวอย่าง เป็นการกำหนด Literal ประเภทต่างๆ ให้กับตัวแปร ในค่าที่เป็นแบบตัวเลขนั้นสามารถกำหนดค่าลงไปโดยตรงได้ทันทีและสามารถกำหนดในรูปแบบสั้นได้อย่างในตัวแปร b และสำหรับ Boolean นั้นจะเป็น True ส่วน String หรือ Character นั้นจะต้องอยู่ภายในเครื่องหมาย double quote หรือ single quote เสมอ

Expression

การทำงานร่วมกันระหว่างค่าตั้งแต่หนึ่งไปจนถึงหลายค่า โดยค่าเหล่านี้จะมีตัวดำเนินการสำหรับควบคุมการทำงาน ในภาษา Python นั้น Expression จะมีสองแบบคือ Boolean expression เป็นการกระทำกันของตัวแปรและตัวดำเนินการและจะได้ผลลัพธ์เป็นค่า Boolean โดยทั่วไปแล้วมักจะเป็นตัวดำเนินการเปรียบเทียบค่าและตัวดำเนินการตรรกศาสตร์ และ Expression ทางคณิตศาสตร์ คือการกระทำกันกับตัวดำเนินการและได้ค่าใหม่ที่ไม่ใช่ Boolean นี่เป็นตัวอย่างของ Expressions ในภาษา Python

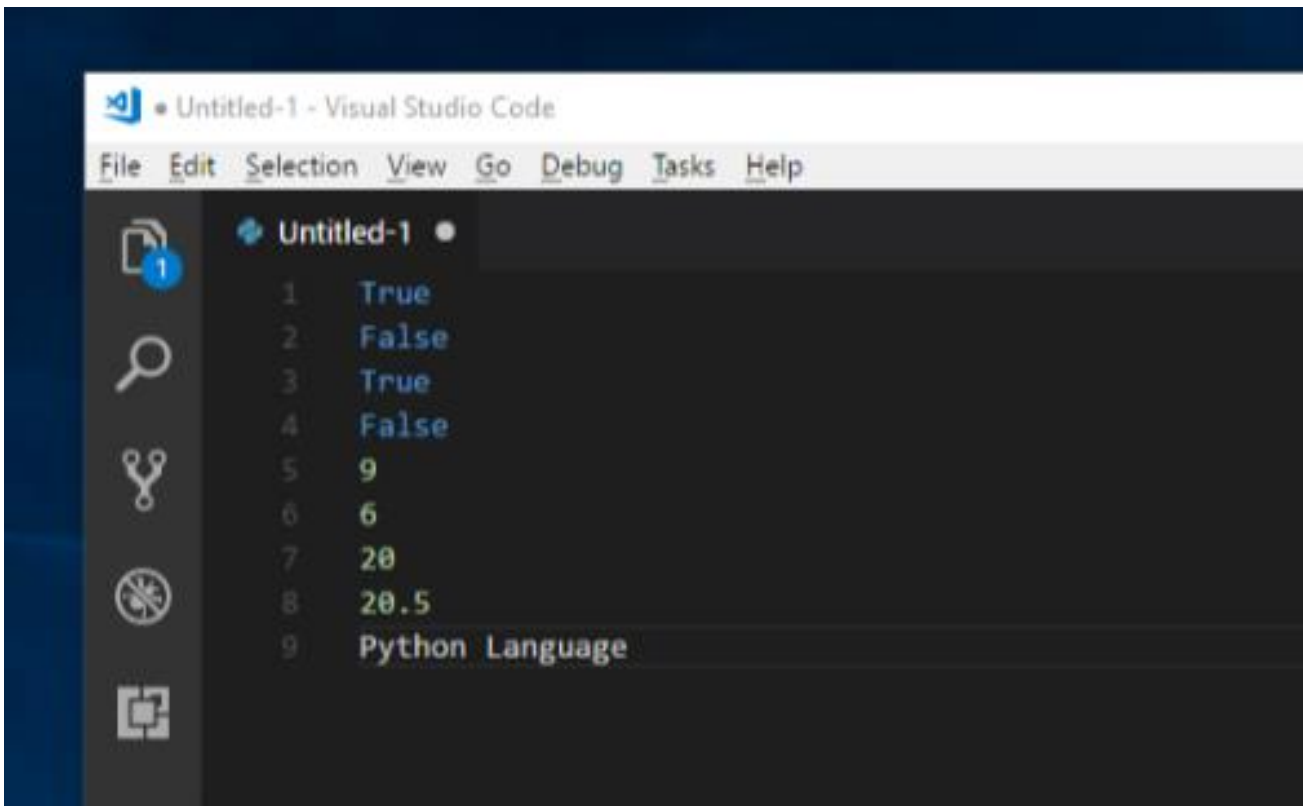


```
Untitled-1 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

Untitled-1
1 a = 4
2 b = 5
3
4 # Boolean expressions
5 print(a == 4)
6 print(a == 5)
7 print(a == 4 and b == 5)
8 print(a == 4 and b == 8)
9
10 # Non-boolean expressions
11 print(a + b)
12 print(a + 2)
13 print(a * b)
14 print(((a * a) + (b * b)) / 2)
15 print("Python " + "Language")
```

ในตัวอย่าง เรามีตัวแปร a และ b และกำหนดค่าให้กับตัวแปรเหล่านี้และทำงานกับตัวดำเนินการประเภทต่างๆ ที่แสดง Expression ในรูปแบบของ Boolean expression ที่จะได้ผลลัพธ์สุดท้ายเป็นเพียงค่า True และ False เท่านั้น ส่วน Non-Boolean expression นั้นสามารถเป็นค่าใดๆ ที่ไม่ใช่ Boolean

นี่เป็นผลลัพธ์การทำงานของโปรแกรมในการทำงานของ Expression ในภาษา Python



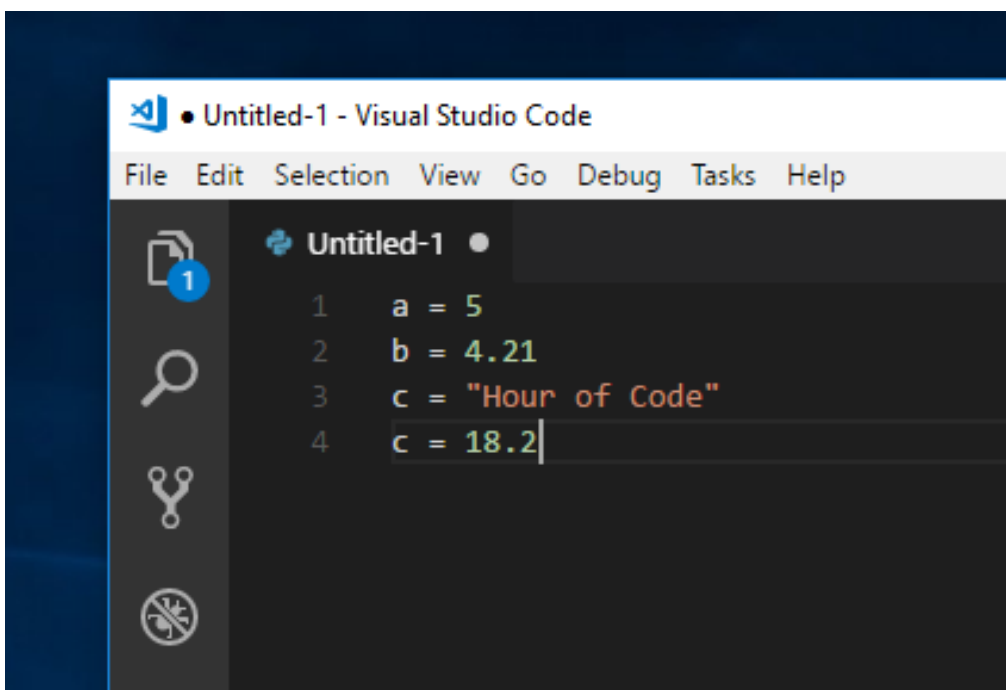
```
1 True
2 False
3 True
4 False
5 9
6 6
7 20
8 20.5
9 Python Language
```

Variable and Data Type

Variable

ตัวแปร (Variable) คือ ชื่อหรือเครื่องหมายที่กำหนดขึ้นสำหรับใช้เก็บค่าในหน่วยความจำ ตัวแปรจะมีชื่อ (identifier) สำหรับใช้ในการอ้างถึงข้อมูลของมัน ในการเขียนโปรแกรม ค่าของตัวแปรสามารถที่จะกำหนดได้ใน run-time หรือเปลี่ยนแปลงอยู่ตลอดเวลาในขณะที่โปรแกรมทำงาน (executing)

ในการเขียนโปรแกรมคอมพิวเตอร์นั้น ตัวแปรจะแตกต่างจากตัวแปรในทางคณิตศาสตร์ ค่าของตัวแปรนั้นไม่จำเป็นต้องประกอบไปด้วยสูตรหรือสมการที่สมบูรณ์เหมือนกับในคณิตศาสตร์ ในคอมพิวเตอร์ ตัวแปรนั้นอาจจะมีการทำงานซ้ำๆ เช่น การกำหนดค่าในที่หนึ่ง และนำไปใช้อีกที่หนึ่งในโปรแกรม และนอกจากนี้ยังสามารถกำหนดค่าใหม่ให้กับตัวแปรได้ตลอดเวลา ต่อไปเป็นตัวอย่างของการประกาศตัวแปรในภาษา Python



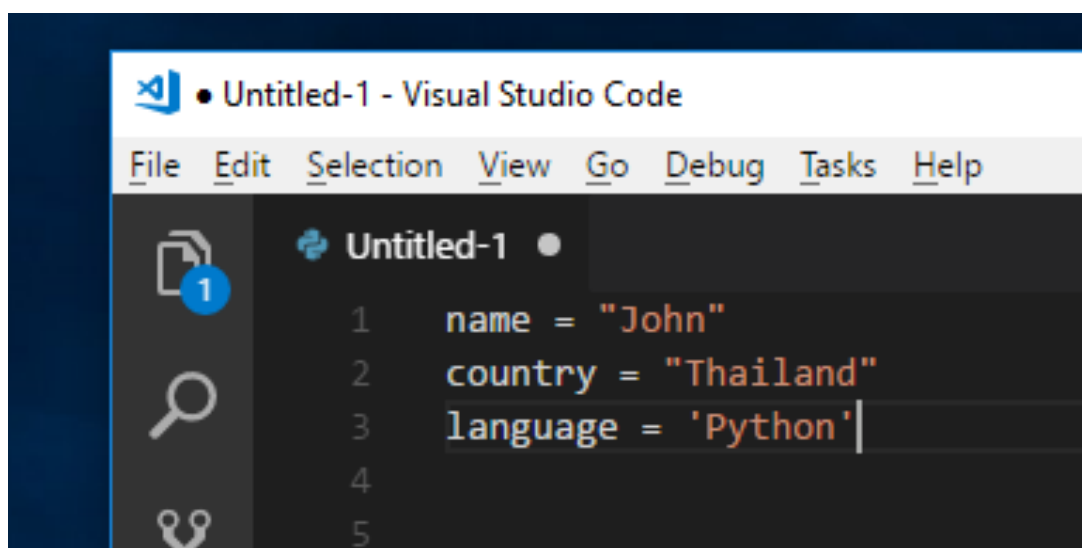
The screenshot shows the Visual Studio Code interface. The title bar reads 'Untitled-1 - Visual Studio Code'. The menu bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Debug', 'Tasks', and 'Help'. The left sidebar contains icons for Explorer (with a '1' badge), Search, Source Control, and Run and Debug. The main editor area displays a Python script with the following code:

```
1 a = 5
2 b = 4.21
3 c = "Hour of Code"
4 c = 18.2
```

ในตัวอย่าง เราได้ทำการประกาศ 3 ตัวแปร ในการประกาศตัวแปรในภาษา Python คุณไม่จำเป็นต้องระบุประเภทของตัวแปรในตอนที่เราประกาศเหมือนในภาษา C ในตัวแปร a มีค่าเป็น 5 และเป็นประเภทเป็น Integer ตัวแปร b มีค่าเป็น 4.21 และเป็นประเภทเป็น Float และตัวแปร c มีค่าเป็น "Hour of Code" และเป็นประเภท String ภายหลังเราได้เปลี่ยนค่าของตัวแปร c เป็น 18.2 ตัวแปรกลายเป็นประเภท Float

Strings

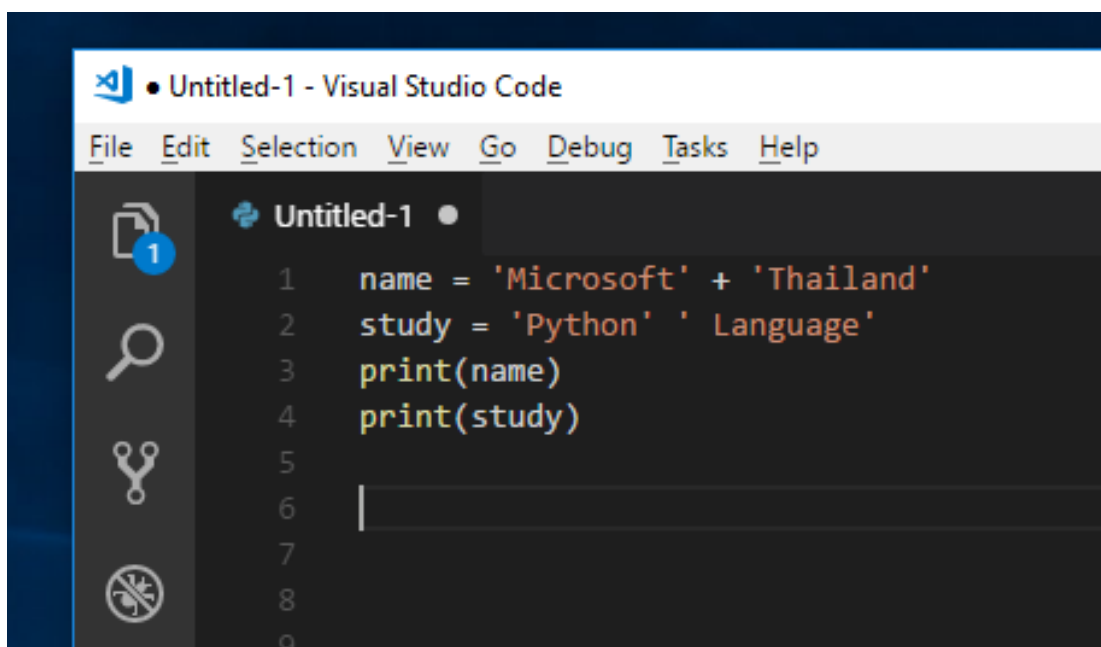
เป็นประเภทข้อมูลที่สำคัญและใช้งานทั่วไปในการเขียนโปรแกรม ในภาษาเขียนโปรแกรมส่วนมากแล้วจะมีประเภทข้อมูลแบบ String และในภาษา Python เช่นกัน String เป็นลำดับของตัวอักษรหลายตัวเรียงต่อกัน ซึ่งในภาษา Python นั้น String จะอยู่ในเครื่องหมาย Double quote หรือ Single quote เท่านั้น



The screenshot shows the Visual Studio Code interface with a file named 'Untitled-1'. The code editor displays three lines of Python code: `name = "John"`, `country = "Thailand"`, and `language = 'Python'`. The code is written in a dark theme with syntax highlighting. The menu bar at the top includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. The left sidebar shows icons for Explorer, Search, and Run and Debug.

```
1 name = "John"
2 country = "Thailand"
3 language = 'Python'|
4
5
```

การทำงานอย่างหนึ่งที่สำคัญเกี่ยวกับ String ก็คือการเชื่อมต่อ String ซึ่งเป็นการนำ String ตั้งต่อสองอันขึ้นไปมาต่อกัน ในภาษา Python คุณสามารถต่อ String ได้โดยใช้เครื่องหมาย + หรือคั่นด้วยช่องว่างหรือบรรทัดใหม่



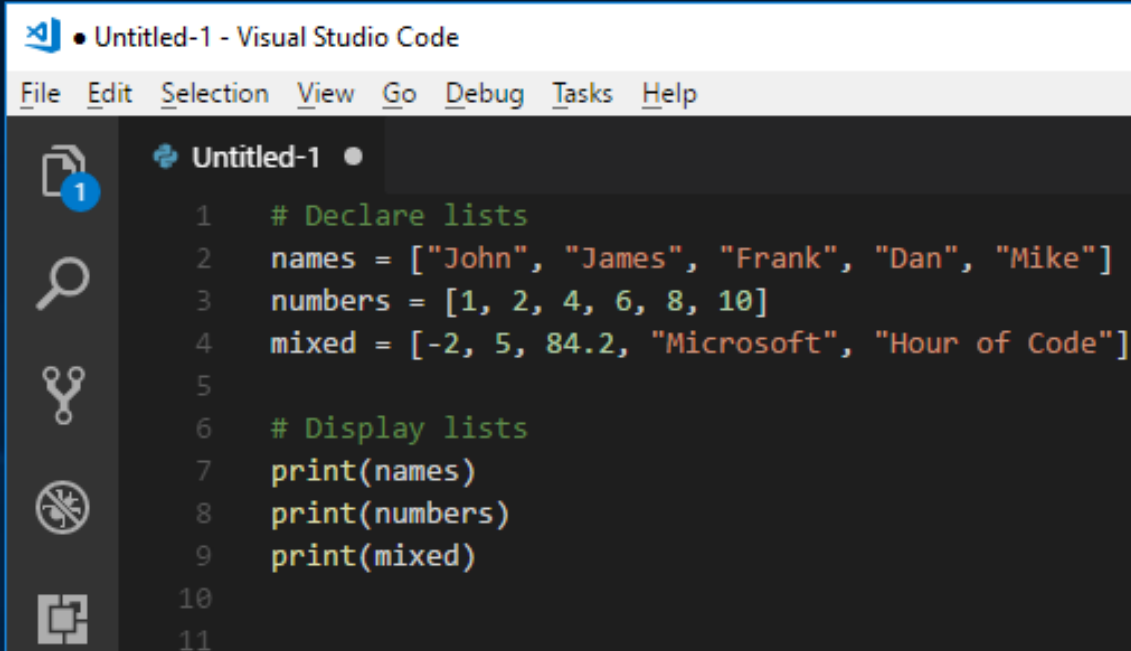
The screenshot shows the Visual Studio Code interface with a file named 'Untitled-1'. The code in the editor is as follows:

```
1 name = 'Microsoft' + 'Thailand'
2 study = 'Python' ' Language'
3 print(name)
4 print(study)
5
6
7
8
9
```

```
>> MicrosoftThailand
>> Python Language
```

Lists

เป็นประเภทข้อมูลที่เก็บข้อมูลแบบเป็นชุดและลำดับ กล่าวคือมันสามารถเก็บข้อมูลได้หลายค่าในตัวแปรเดียว และมี Index สำหรับเข้าถึงข้อมูล มันสามารถเก็บข้อมูลได้หลายตัวและยังสามารถเป็นประเภทข้อมูลที่แตกต่างกันได้อีกด้วย



```

1  # Declare lists
2  names = ["John", "James", "Frank", "Dan", "Mike"]
3  numbers = [1, 2, 4, 6, 8, 10]
4  mixed = [-2, 5, 84.2, "Microsoft", "Hour of Code"]
5
6  # Display lists
7  print(names)
8  print(numbers)
9  print(mixed)
10
11

```

```

>> John James Frank Dan Mike
>> 1 2 4 6 8 10
>> -2 5 84.2 Microsoft Hour of Code

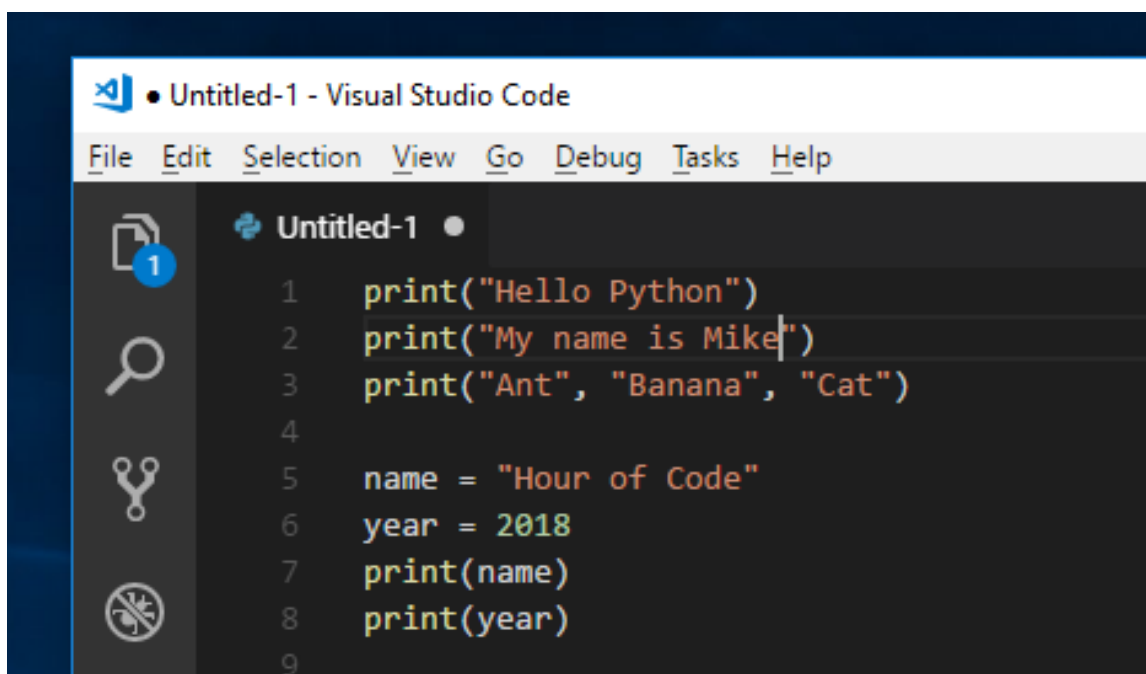
```

Lists นั้นทำงานกับ Index ดังนั้นเราสามารถเข้าถึงข้อมูลของ List โดยการใช้ Index ของมันได้ ในตัวอย่างเป็นการเข้าถึงข้อมูลภายใน Index ซึ่ง Index ของ List นั้นจะเริ่มจาก 0 ไปจนถึงจำนวนทั้งหมดของมันลบด้วย 1

Input and Output

Input

ในการแสดงผลในภาษา Python นั้นจะใช้ฟังก์ชัน `print()` เพื่อแสดงผลข้อความ ตัวเลข หรือข้อมูลประเภทอื่นๆ ออกทางหน้าจอ



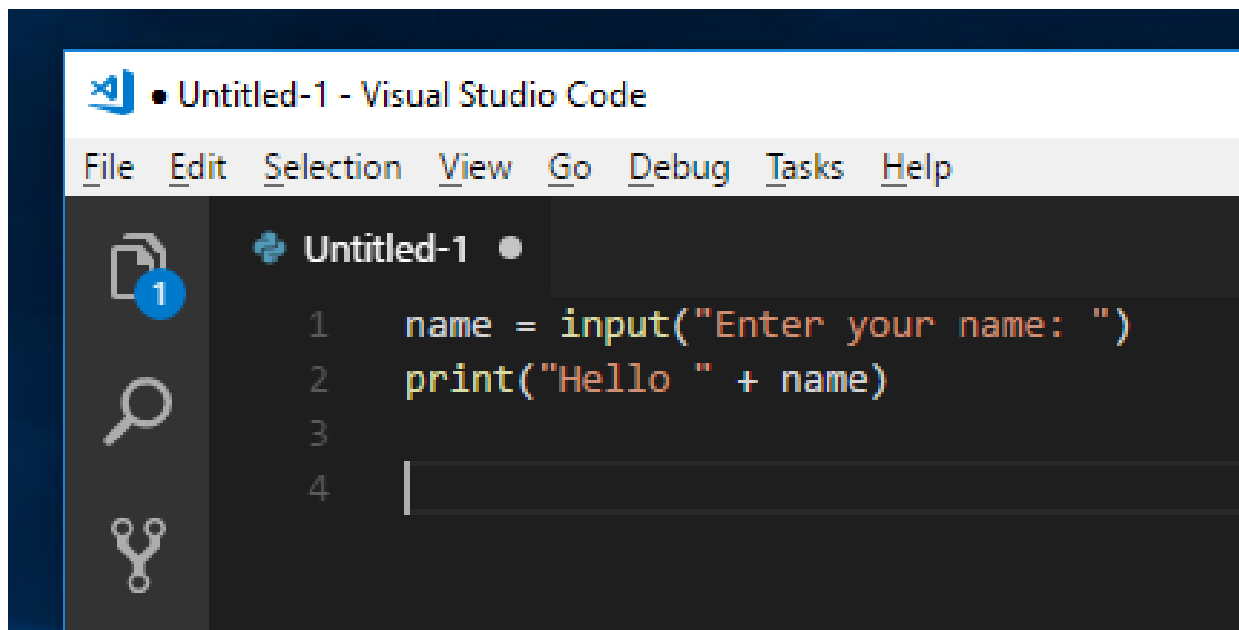
```
• Untitled-1 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

1 print("Hello Python")
2 print("My name is Mike")
3 print("Ant", "Banana", "Cat")
4
5 name = "Hour of Code"
6 year = 2018
7 print(name)
8 print(year)
9
```

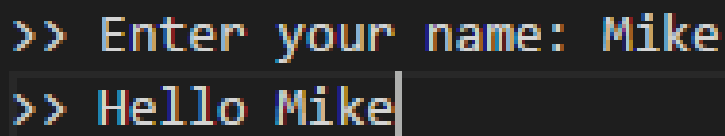
```
>> Hello Python
>> My name is Mike
>> Ant Banana Cat
>>
>> Hour of Code
>> 2018
```


Output

นอกจากการแสดงผลแล้วนั้น การติดต่อกับผู้ใช้ในอีกรูปแบบหนึ่งคือการรับค่า โดยทั่วไปแล้วมักจะเป็นการรับค่าทางคีย์บอร์ด ในภาษา Python เราใช้ฟังก์ชัน `input()` สำหรับการรับค่า String จากทางคีย์บอร์ด มาดูตัวอย่างการรับค่าจากผู้ใช้ในภาษา Python

A screenshot of the Visual Studio Code editor interface. The title bar shows 'Untitled-1 - Visual Studio Code'. The menu bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Debug', 'Tasks', and 'Help'. The left sidebar shows a file explorer with one file 'Untitled-1' and a search icon. The main editor area shows a Python script with two lines of code: `name = input("Enter your name: ")` and `print("Hello " + name)`. The cursor is at the end of the second line.

```
1 name = input("Enter your name: ")
2 print("Hello " + name)
3
4
```

A screenshot of a terminal window showing the execution of the Python script. The first line shows the prompt `>>` followed by the input `Enter your name: Mike`. The second line shows the prompt `>>` followed by the output `Hello Mike`.

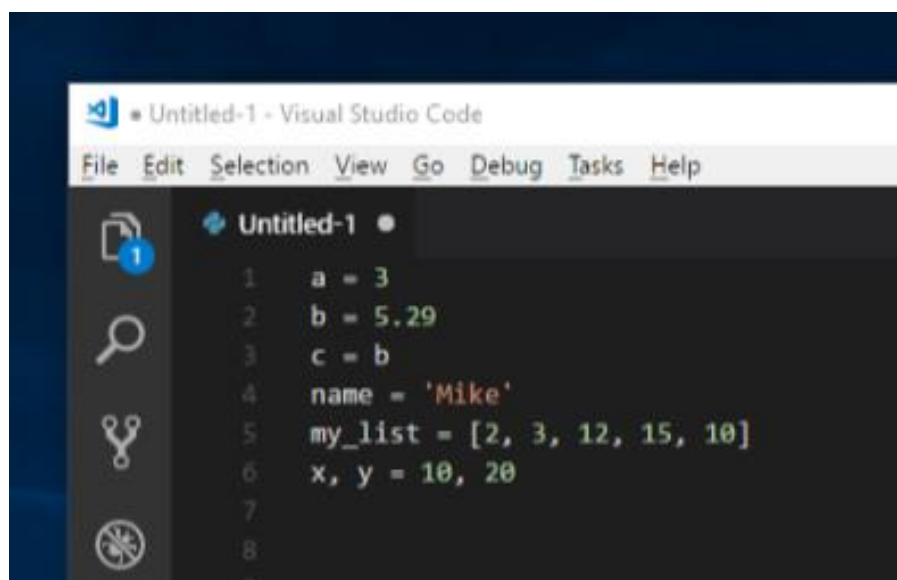
```
>> Enter your name: Mike
>> Hello Mike
```

Operators

กลุ่มของเครื่องหมายหรือสัญลักษณ์ที่ใช้ทำงานเหมือนกับฟังก์ชัน แต่แตกต่างกันตรงไวยากรณ์หรือความหมายในการใช้งาน ในภาษา Python นั้นสนับสนุนตัวดำเนินการประเภทต่างๆ สำหรับการเขียนโปรแกรม เช่น ตัวดำเนินการ + เป็นตัวดำเนินการทางคณิตศาสตร์ที่ใช้สำหรับการบวกตัวเลขเข้าด้วยกัน หรือตัวดำเนินการ > เป็นตัวดำเนินการเพื่อให้เปรียบเทียบค่าสองค่า

Assignment Operator

ตัวดำเนินการที่เป็นพื้นฐานที่สุดสำหรับการเขียนโปรแกรมในทุกๆ ภาษาก็คือ ตัวดำเนินการกำหนดค่า (Assignment operator) ตัวดำเนินการนี้แสดงโดยใช้เครื่องหมายเท่ากับ (=) มันใช้สำหรับกำหนดค่าให้กับตัวแปร

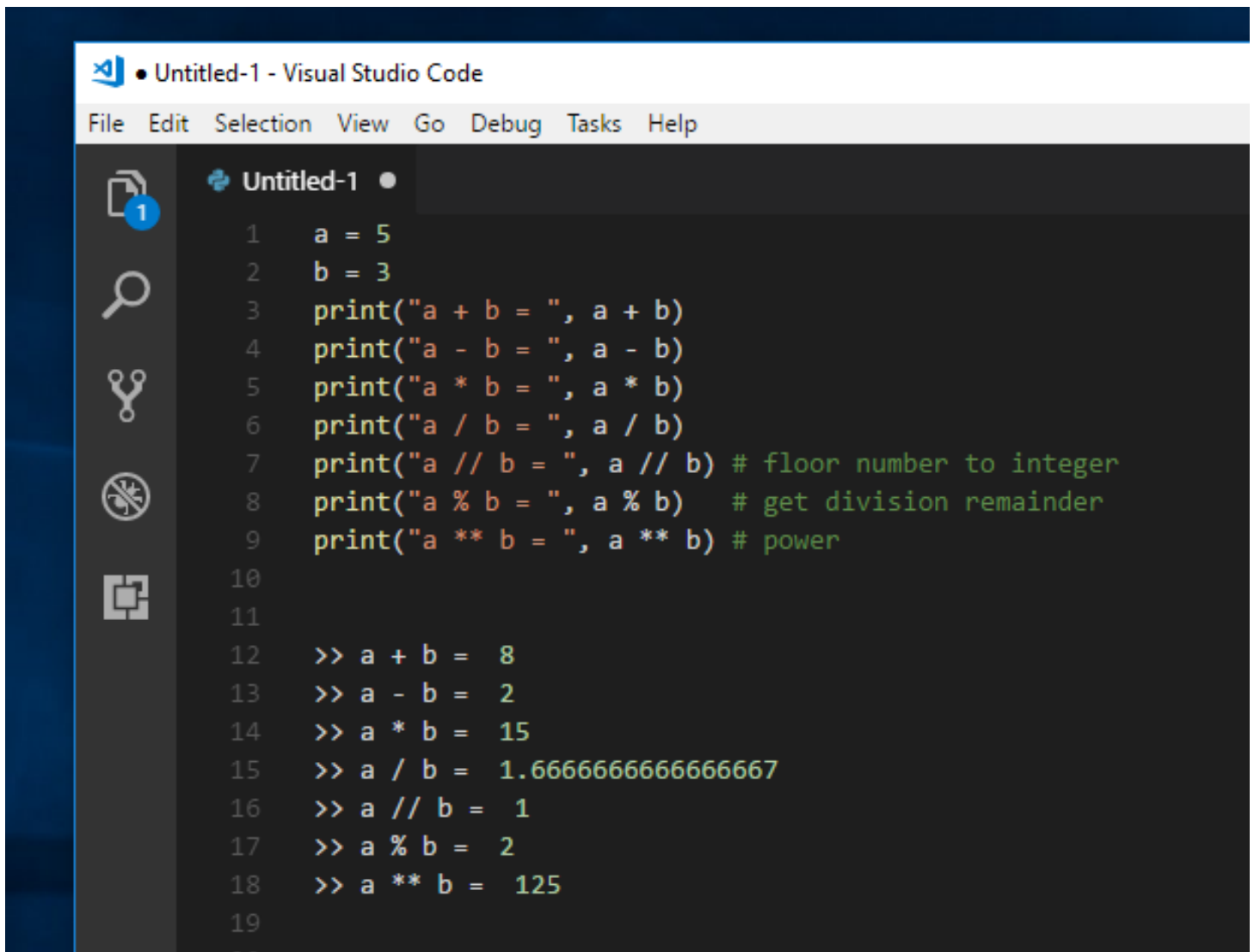
A screenshot of the Visual Studio Code editor interface. The title bar shows 'Untitled-1 - Visual Studio Code'. The menu bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Debug', 'Tasks', and 'Help'. The left sidebar shows a file explorer with 'Untitled-1' selected. The main editor area displays the following Python code:

```
1 a = 3
2 b = 5.29
3 c = b
4 name = 'Mike'
5 my_list = [2, 3, 12, 15, 10]
6 x, y = 10, 20
7
8
```

Arithmetic operators

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic operators) คือตัวดำเนินการที่ใช้สำหรับการคำนวณทางคณิตศาสตร์ในพื้นฐาน เช่น การบวก การลบ การคูณ และการหาร มากไปกว่านั้น ในภาษา Python ยังมีตัวดำเนินการทางคณิตศาสตร์เพิ่มเติม เช่น การหารเอาเศษ (Modulo) การหารแบบเลขจำนวนเต็ม และการยกกำลัง

Operator	Name	Example
+	Addition	$a + b$
-	Subtraction	$a - b$
*	Multiplication	$a * b$
/	Division	a / b
//	Division and floor	$a // b$
%	Modulo	$a \% b$
**	Power	$a ** b$



The image shows a screenshot of the Visual Studio Code editor interface. The title bar at the top reads "Untitled-1 - Visual Studio Code". Below the title bar is a menu bar with the following options: File, Edit, Selection, View, Go, Debug, Tasks, and Help. The editor area is divided into two panes. The left pane shows the Explorer view with a file named "Untitled-1" and a search icon. The right pane shows the code editor with the following Python code:

```
1 a = 5
2 b = 3
3 print("a + b = ", a + b)
4 print("a - b = ", a - b)
5 print("a * b = ", a * b)
6 print("a / b = ", a / b)
7 print("a // b = ", a // b) # floor number to integer
8 print("a % b = ", a % b) # get division remainder
9 print("a ** b = ", a ** b) # power
```

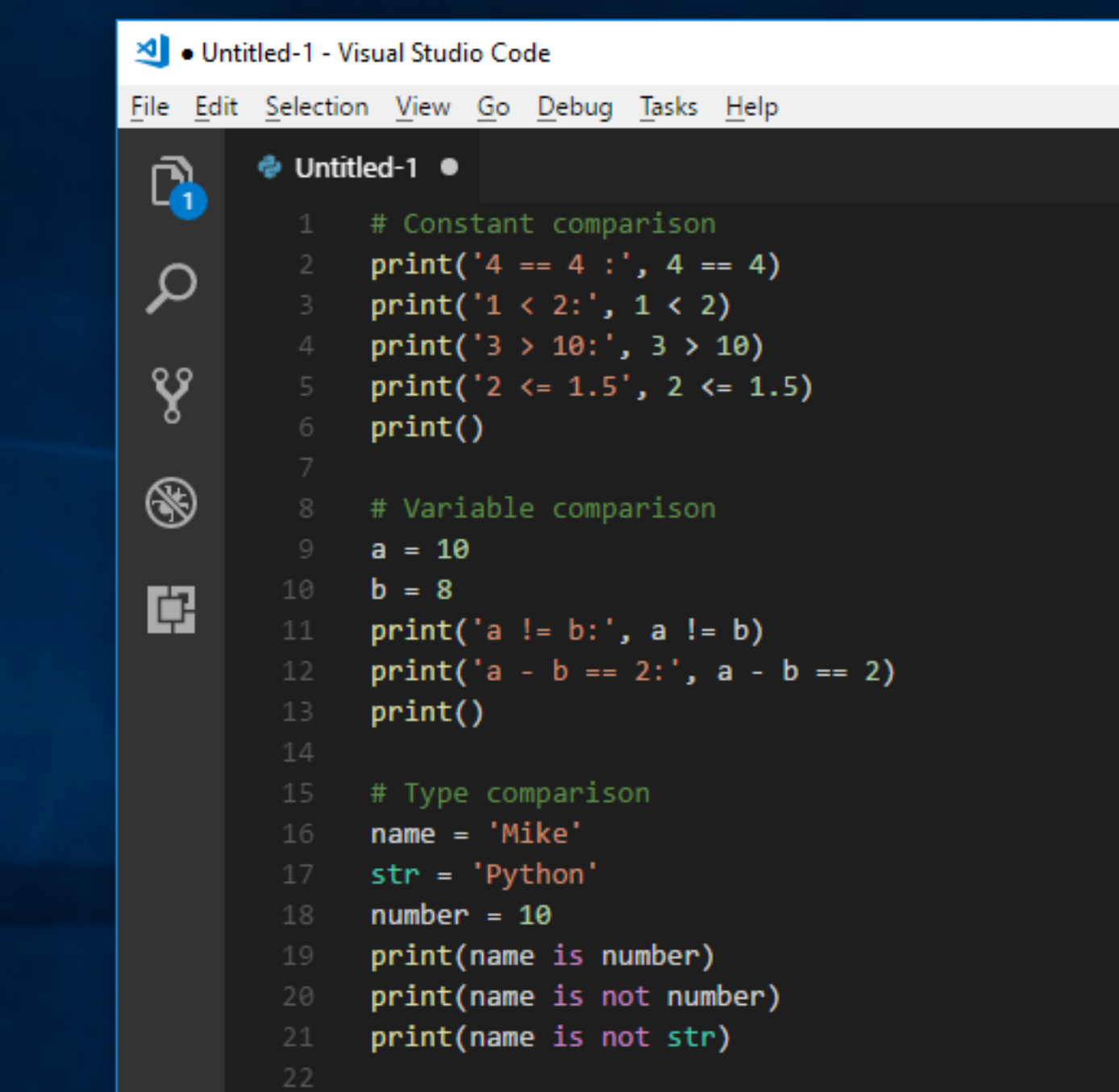
The output of the code is displayed in the bottom pane:

```
10
11
12 >> a + b = 8
13 >> a - b = 2
14 >> a * b = 15
15 >> a / b = 1.6666666666666667
16 >> a // b = 1
17 >> a % b = 2
18 >> a ** b = 125
19
```

Comparison operators

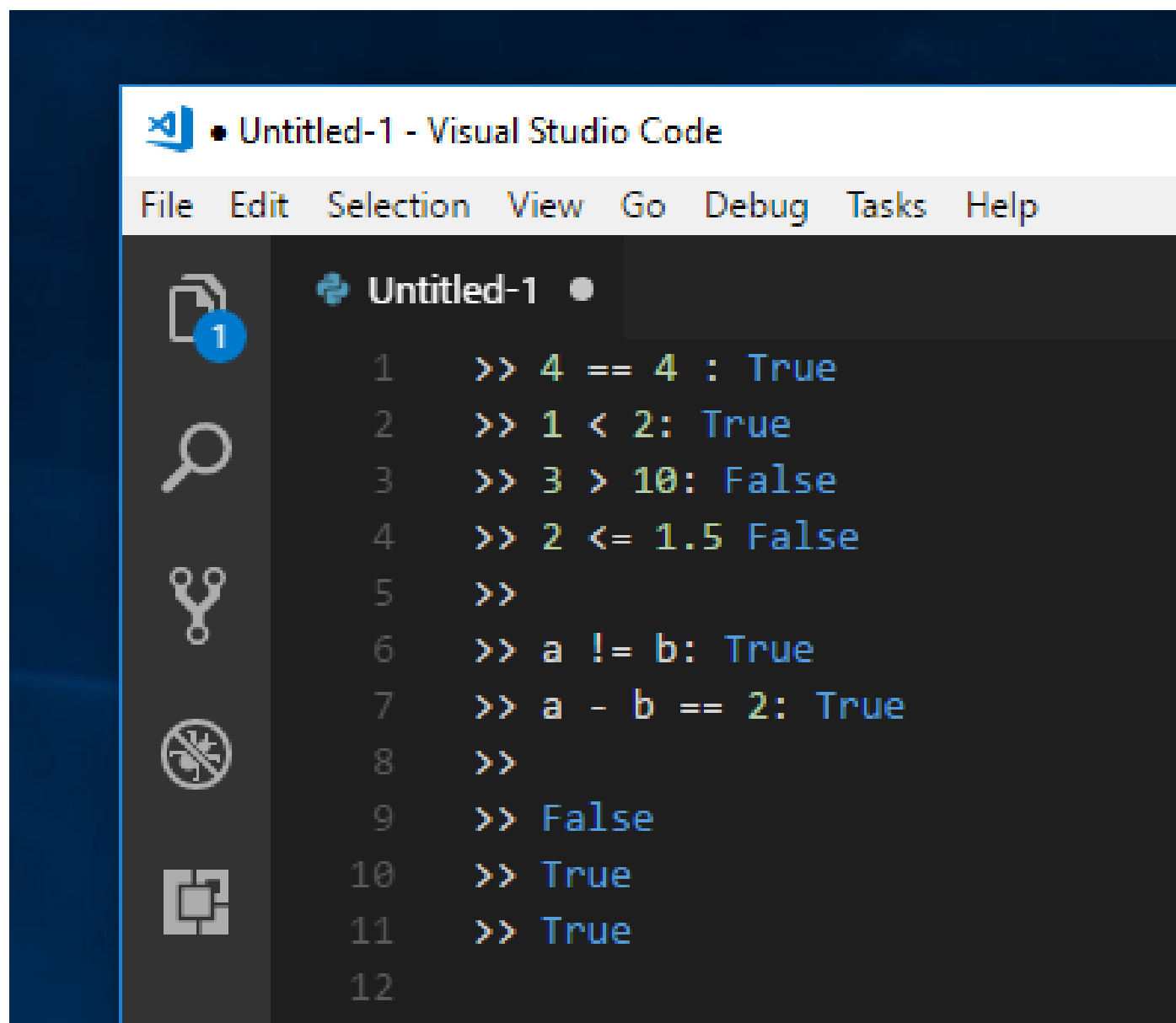
ตัวดำเนินการเปรียบเทียบ (Comparison operators) คือตัวดำเนินการที่ใช้สำหรับเปรียบเทียบค่าหรือค่าในตัวแปร ซึ่งผลลัพธ์ของการเปรียบเทียบนั้นจะเป็น True หากเงื่อนไขเป็นจริง และเป็น False หากเงื่อนไขไม่เป็นจริง ตัวดำเนินการเปรียบเทียบมักจะใช้กับคำสั่งตรวจสอบเงื่อนไข if และคำสั่งวนซ้ำ for while เพื่อควบคุมการทำงานของโปรแกรม

Operator	Name	Example
<	Less than	<code>a < b</code>
<=	Less than or equal	<code>a <= b</code>
>	Greater than	<code>a > b</code>
>=	Greater than or equal	<code>a >= b</code>
==	Equal	<code>a == b</code>
!=	Not equal	<code>a != b</code>
is	Object identity	<code>a is b</code>
is not	Negated object identity	<code>a is not b</code>



The image shows a screenshot of the Visual Studio Code editor interface. The title bar at the top reads "Untitled-1 - Visual Studio Code". Below the title bar is a menu bar with the following options: File, Edit, Selection, View, Go, Debug, Tasks, and Help. The left sidebar contains several icons: a file explorer icon with a blue circle containing the number "1", a search icon, a source control icon, a run and debug icon, and a testing icon. The main editor area displays a Python script in a dark theme. The script is titled "Untitled-1" and contains three sections of code, each starting with a comment line. The first section, labeled "# Constant comparison", contains five lines of code that use the print function to output the results of constant comparisons: '4 == 4', '1 < 2', '3 > 10', and '2 <= 1.5'. The second section, labeled "# Variable comparison", contains five lines of code that first assign values to variables 'a' and 'b', and then use the print function to output the results of comparisons involving these variables: 'a != b', 'a - b == 2'. The third section, labeled "# Type comparison", contains five lines of code that first assign values to variables 'name', 'str', and 'number', and then use the print function to output the results of type comparisons: 'name is number', 'name is not number', and 'name is not str'. The line numbers 1 through 22 are visible on the left side of the code editor.

```
1  # Constant comparison
2  print('4 == 4 :', 4 == 4)
3  print('1 < 2:', 1 < 2)
4  print('3 > 10:', 3 > 10)
5  print('2 <= 1.5', 2 <= 1.5)
6  print()
7
8  # Variable comparison
9  a = 10
10 b = 8
11 print('a != b:', a != b)
12 print('a - b == 2:', a - b == 2)
13 print()
14
15 # Type comparison
16 name = 'Mike'
17 str = 'Python'
18 number = 10
19 print(name is number)
20 print(name is not number)
21 print(name is not str)
22
```



Conditional Statement

ในบทนี้เราจะพูดถึงการควบคุมการทำงานของโปรแกรมด้วยคำสั่ง if, if else และ elif เพื่อให้โปรแกรมสามารถทำงานซับซ้อนและมีประสิทธิภาพมากขึ้น ยกตัวอย่างเช่น เครื่องปรับอากาศจะทำงานอัตโนมัติถ้าหากอุณหภูมิในห้องสูงหรือต่ำเกินไป หรือรถยนต์จะแสดงสัญญาณเตือนหากน้ำมันกำลังจะหมด เป็นต้น ซึ่งทั้งหมดนี้เกิดจากการกำหนดเงื่อนไขการทำงานให้โปรแกรม มาเริ่มกับคำสั่ง if ในภาษา Python

If Statement

คำสั่ง if เป็นคำสั่งที่ใช้ควบคุมการทำงานของโปรแกรมที่เป็นพื้นฐานและง่ายที่สุด เราใช้คำสั่ง if เพื่อสร้างเงื่อนไขให้โปรแกรมทำงานตามที่เราต้องการเมื่อเงื่อนไขนั้นตรงกับที่เรากำหนด เช่น การตรวจสอบค่าในตัวแปรกับตัวดำเนินการประเภทต่างๆ

```
if expression:  
    # statements
```

ในตัวอย่าง เป็นรูปแบบของการใช้งานคำสั่ง if และ expression เป็นเงื่อนไขที่สร้างจากตัวดำเนินการประเภทต่างๆ ที่เป็น boolean expression โดยโปรแกรมจะทำงานในบล็อกคำสั่ง if ถ้าหากเงื่อนไขเป็น True ไม่เช่นนั้นโปรแกรมจะข้ามการทำงานไป ในบล็อกของคำสั่ง if จะประกอบไปด้วยคำสั่งการทำงานของโปรแกรม คำสั่งทั้งหมดในบล็อกต้องมีระยะเว้นช่องว่างที่เท่ากัน

```

Untitled-1
1  n = 10
2  if n == 10:
3      print('n equal to 10')
4
5  logged_in = False
6  if not logged_in:
7      print('You must login to continue')
8
9  m = 4
10 if m % 2 == 0 and m > 0:
11     print('m is even and positive numbers')
12
13 if 3 > 10:
14     print('This block isn\'t executed')

```

ในตัวอย่าง เป็นการใช้งานคำสั่ง if เพื่อกำหนดให้โปรแกรมทำงานตามเงื่อนไขต่างๆ ในบล็อกแรกเป็นการตรวจสอบค่าในตัวแปร n ว่าเท่ากับ 10 หรือไม่ เนื่องจากค่าในตัวแปรนั้นเท่ากับ 10 ทำให้เงื่อนไขเป็นจริง และโปรแกรมทำงานในบล็อกของคำสั่ง if และต่อมาเรามีตัวแปร boolean logged_in เก็บค่าสถานะการเข้าสู่ระบบ เราได้ทำการตรวจสอบโดยใช้ตัวดำเนินการ not สำหรับตรวจสอบว่าถ้าหากผู้ใช้ไม่เข้าสู่ระบบ จะแสดงข้อความบอกว่าต้องเข้าสู่ระบบก่อนที่จะใช้งาน

ต่อมาเป็นการตรวจสอบค่าในตัวแปร m ว่าเป็นทั้งจำนวนเต็มบวกและจำนวนคู่หรือไม่ เราได้ใช้ตัวดำเนินการ and เพื่อเชื่อม expression ย่อยทั้งสอง และเงื่อนไขเป็นจริงทำให้ในบล็อกคำสั่ง if ทำงาน สุดท้ายเป็นเปรียบเทียบค่าของตัวเลข เราได้เปรียบเทียบว่า 3 มากกว่า 10 หรือไม่ เนื่องจากเงื่อนไขเป็น False ทำให้โปรแกรมข้ามการทำงานบล็อกนี้ไป

Untitled-1

```
1  >> n equal to 10
2  >> You must login to post
3  >> m is even and positive numbers
4
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม คุณจะเห็นว่าในสามบล็อกแรกของคำสั่ง if นั้นทำงานเพราะว่าเงื่อนไขเป็นจริงหรือ True และในบล็อกสุดท้ายไม่ทำงานเพราะเงื่อนไขไม่เป็นจริงหรือ False

If – else Statement

หลังจากที่คุณได้รู้จักกับคำสั่ง if ไปแล้ว อีกคำสั่งหนึ่งที่ทำงานควบคู่กับคำสั่ง if คือ คำสั่ง else clause โดยโปรแกรมจะทำงานในคำสั่ง else ถ้าหากเงื่อนไขในคำสั่ง if นั้นไม่เป็นจริง กล่าวอีกนัยหนึ่ง มันจะทำงานเมื่อเงื่อนไขก่อนหน้านี้ไม่เป็นจริงหรือเป็นเงื่อนไข Default มาดูตัวอย่างการใช้งาน if else ในภาษา Python

Untitled-1

```
1  n = 5
2  if n == 10:
3      print('n equal to 10')
4  else:
5      print('n is something else except 10')
6
7  name = 'James'
8  if name == 'Mike':
9      print('Hi, Mike.')
10 else:
11     print('Who are you?')
12
13 money = 300
14 if money >= 350:
15     print('You can buy an iPad')
16 else:
17     print('You don\'t have enough money to buy an iPad')
18
```

ในตัวอย่าง เป็นโปรแกรมเพื่อทดสอบการทำงานของคำสั่ง else เราได้เพิ่มบล็อกของคำสั่ง else เข้ามาหลังจากคำสั่ง if ซึ่งโค้ดในบล็อกของคำสั่ง else จะทำงาน ถ้าหากเงื่อนไขใน if ไม่เป็นจริง นั่นหมายถึงโปรแกรมของเราสามารถทำงานได้เพียงหนึ่งทางเลือกเท่านั้น

ในการตรวจสอบครั้งแรก เป็นการตรวจสอบค่าในตัวแปร n ว่าเท่ากับ 10 หรือไม่ เพราะค่าในตัวแปรนั้นเป็น 5 ทำให้เงื่อนไขไม่เป็นจริง และโปรแกรมทำงานในบล็อกคำสั่ง else แทน ต่อมาเป็นการตรวจสอบชื่อในตัวแปร name ว่าเป็น "Mike" หรือไม่ เพราะค่าในตัวแปรนั้นเป็น "James" ทำให้โปรแกรมทำงานในบล็อกคำสั่ง else และสุดท้ายนั้นเราตรวจสอบว่าหากมีเงินในตัวแปร money มากกว่าหรือเท่ากับ 350 จะได้ซื้อ iPad เพราะว่ามีเงินไม่พอ โปรแกรมจึงบอกว่าเงินไม่พอที่จะซื้อ

Untitled-1

```
1  >> n is something else except 10
2  >> Who are you?
3  >> You don't have enough money to buy an iPad
4
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมซึ่งจะทำงานในบล็อกของคำสั่ง else ทั้งหมดเพราะเงื่อนไขไม่เป็นจริงทั้งหมด

If-elif Statement

คำสั่ง elif นั้นเป็นคำสั่งที่ใช้สำหรับสร้างเงื่อนไขแบบหลายทางเลือกให้กับโปรแกรมที่มีการทำงานเช่นเดียวกับ switch case ในภาษาอื่นๆ คำสั่ง elif นั้นต้องใช้หลังจากคำสั่ง if เสมอและสามารถมี else ได้ในเงื่อนไขสุดท้าย มาดูตัวอย่างการใช้งานคำสั่ง elif ในภาษา Python

Untitled-1

```
1  print('Welcome to Hour of Code game')
2  level = input('Enter level (1 - 4): ')
3
4  if level == '1':
5      print('Easy')
6  elif level == '2':
7      print('Medium')
8  elif level == '3':
9      print('Hard')
10 elif level == '4':
11     print('Expert')
12 else:
13     print('Invalid level selected')
14
```

ในตัวอย่าง เป็นโปรแกรมจำลองในการเลือกโหมดของการเล่นเกม เราได้ให้ผู้ใช้กรอกค่าระหว่าง 1 -4 เพื่อใช้ในการเปรียบเทียบกับระดับความยากของเกม โดยที่ 1 เป็นระดับที่ง่ายที่สุด และ 4 นั้นเป็นระดับที่ยากที่สุด คุณจะเห็นว่าเราได้ให้คำสั่ง elif เพราะเรามีเงื่อนไข 4 แบบ และคำสั่ง else ในการกรณีที่ตัวเลขที่ผู้เล่นกรอกเข้ามานั้นไม่ตรงกับเงื่อนไขใดๆ ก่อนหน้าเลย

```

Untitled-1
1  >> Welcome to Hour of Code game
2  >> Enter level (1 - 4): 4
3  >> Expert

```

```

Untitled-1
1  Welcome to Hour of Code game
2  Enter level (1 - 4): 7
3  Invalid level selected
4

```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมเมื่อเรากรอก 4 และ 7 ตามลำดับ เมื่อเรากรอก 4 นั้นโปรแกรมตรงกับเงื่อนไขของ elif ที่ `level == 4` และเมื่อเรากรอก 7 โปรแกรมไม่ตรงกับเงื่อนไขใดๆ เลยทำให้ทำงานในบล็อกของคำสั่ง else

Iteration Statement

ในบทนี้เราจะพูดถึงการควบคุมการทำงานโดยใช้คำสั่ง while loop และ for loop คำสั่งเหล่านี้สามารถควบคุมโปรแกรมให้ทำงานซ้ำๆ ในเงื่อนไขที่กำหนดและเพิ่มความสามารถของการเขียนโปรแกรม ตัวอย่างของการทำงานซ้ำๆ นั้นพบเห็นเห็นได้ทั่วไปในชีวิตประจำวัน เช่น โปรแกรมพยากรณ์สภาพอากาศที่เกิดขึ้นในทุกๆ วัน หรือ การไปทำงานของคุณในทุกๆ เช้า เป็นต้น ดังนั้นแนวคิดเหล่านี้จึงถูกนำมาใช้กับการเขียนโปรแกรม

While loop

เป็นคำสั่งวนซ้ำที่ง่ายและพื้นฐานที่สุดในภาษา Python คำสั่ง while loop นั้นใช้ควบคุมโปรแกรมให้ทำงานบางอย่างซ้ำๆ ในขณะที่เงื่อนไขของลูปนั้นยังคงเป็นจริงอยู่

```
Untitled-1
1 while expression:
2     # statements
3
```

ในรูปแบบการใช้งานคำสั่ง while loop นั้น เราสร้างลูปด้วยคำสั่ง while และตามด้วยการกำหนด expression ซึ่งเป็นเงื่อนไขที่จะให้โปรแกรมทำงาน ซึ่งโปรแกรมจะทำงานจนกว่าเงื่อนไขจะเป็น False และสิ้นสุดการทำงานของลูป ภายในบล็อกคำสั่ง while นั้นประกอบไปด้วยคำสั่งการทำงานของโปรแกรม

```

Untitled-1
1  i = 1
2
3  while i <= 10:
4      print(i, end = ', ')
5      i = i + 1
6
7

```

ในตัวอย่าง โปรแกรมในการแสดงตัวเลข 1 ถึง 10 โดยใช้คำสั่ง while loop ในตอนแรก เราได้ประกาศตัวแปร i และกำหนดค่าให้กับตัวแปรเป็น 1 หลังจากนั้นเราสร้างเงื่อนไขสำหรับ while loop เป็น $i \leq 10$ นั้นหมายความว่าโปรแกรมจะทำงานในขณะที่ค่าในตัวแปร i ยังคงน้อยกว่าหรือเท่ากับ 10 และเราแสดงผลค่าของ i ในบล็อคของคำสั่ง while และเราเพิ่มค่าของตัวแปรขึ้นทุกครั้งหลังจากที่แสดงผลเสร็จ ถ้าหากคุณไม่เพิ่มค่าของ i ลูปจะทำงานไม่มีวันหยุดหรือเรียกว่า Infinite loop

```

Untitled-1
1  1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
2
3

```

นี่เป็นผลลัพธ์การทำงานของโปรแกรมในการแสดงตัวเลข 1 ถึง 10 โดยใช้คำสั่งวนซ้ำ คุณจะเห็นว่าเราสามารถเขียนโปรแกรมได้ง่ายขึ้นโดยที่คุณไม่จำเป็นต้องใช้ฟังก์ชัน print() เพื่อแสดงผล 10 ครั้ง

For loop

คำสั่งวนซ้ำที่ใช้ควบคุมการทำงานซ้ำๆ ในจำนวนรอบที่แน่นอน ในภาษา Python นั้นคำสั่ง for loop จะแตกต่างจากภาษาอื่นๆ อย่างภาษา C มันมักจะใช้สำหรับการวนอ่านค่าภายในออบเจ็กต์ เช่น ลิสต์หรือออบเจ็กต์จากฟังก์ชัน range() เป็นต้น

Untitled-1

```
1  # loop through string
2  site = 'HourofCode'
3  for n in site:
4      print(n)
5
6  # loop through list
7  names = ['Ant', 'Bee', 'Cat', 'Dog']
8  for n in names:
9      print(n)
10
11  numbers = [10, 20, 30, 40, 50, 60, 70, 80]
12  for n in numbers:
13      print(n)
14
```

ในตัวอย่าง เป็นการใช้คำสั่ง for loop ในการวนอ่านค่าในตัวแปร String และอ่านข้อมูลภายในลิสต์ ในรูปแบบการวนอ่านค่าตัวอักษรในตัวแปร String site โดยโปรแกรมจะวนอ่านค่าทีละตัวมาเก็บไว้ในตัวแปร n ซึ่งเป็นพารามิเตอร์ของคำสั่ง for loop และวนอ่านค่าจนครบทุกตัวอักษรและจบการทำงานของ loop และอีกในสอง loop ต่อมาเป็นการใช้คำสั่ง for loop ในการวนอ่านข้อมูลภายในลิสต์ของ String และตัวเลข

```
Untitled-1
1  H
2  o
3  u
4  r
5  o
6  f
7  C
8  o
9  d
10 e
11 Ant
12 Bee
13 Cat
14 Dog
15 10
16 20
17 30
18 40
19 50
20 60
21 70
22 80
23
```

แบบฝึกหัด

1. หาพื้นที่วงกลมและเส้นรอบวงของวงกลมโดยรับค่ารัศมีจากผู้ใช้งาน และแสดงคำตอบ

```

1  radius = input()
2
3  area = 3.14 * radius * radius
4  circum = 2 * 3.14 * radius
5
6  print("Area is " + area)
7  print("Circumference is " + circum)
8

```

2. เปลี่ยนอุณหภูมิจากองศาเซลเซียสเป็นฟาเรนไฮต์โดยป้อนอุณหภูมิเป็นองศาเซลเซียส ทางแป้นพิมพ์. แล้วแสดงผลลัพธ์ทางจอภาพ
3. เขียนโปรแกรมเป็นรูปสามเหลี่ยมแบบภูเขา (Triangle - Mountain) โดยรับข้อมูลจำนวนบรรทัดจากผู้ใช้งาน

Input number: 5

```

*
* * *
* * * * *
* * * * * *
* * * * * * *

```

Input number: 3

```

*
* * *
* * * * *

```

Recommended Resources

แหล่งการเรียนรู้ศึกษาเพิ่มเติม

Microsoft Virtual Academy : MVA

<https://mva.microsoft.com/>

Microsoft Virtual Academy | Courses ▾ Search all courses 🔍

Free Microsoft training delivered by experts

Developers IT Pros Data Pros

Windows 10 Cloud Development Game Development Web Development Database Development

C# / XAML Visual Studio For Beginners Mobile App Development

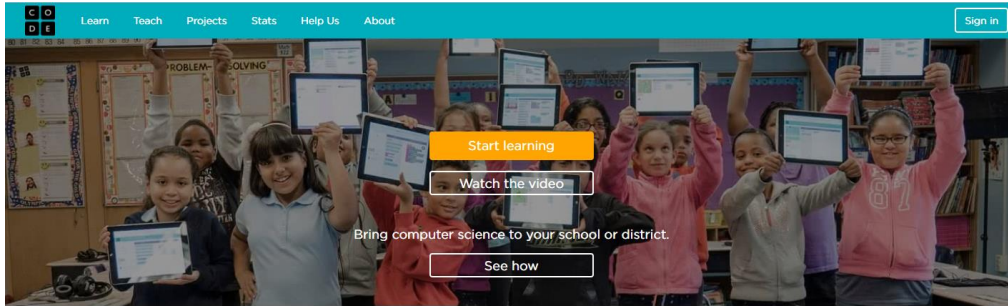
Browse all developer courses ➔

Learning Paths
Curated collections of courses to help you build skills. Complete a learning path to earn a badge you can share with others.
[Learn more](#)

MICROSOFT VIRTUAL ACADEMY
Microsoft
Complete a Learning Path to Earn this Badge
Learning Path Completed

Code.org

<https://code.org/>



Learn Teach Projects Stats Help Us About Sign in

Start learning

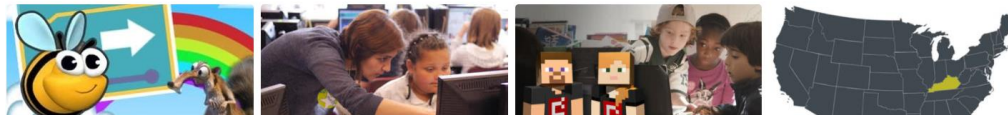
Watch the video

Bring computer science to your school or district.

See how

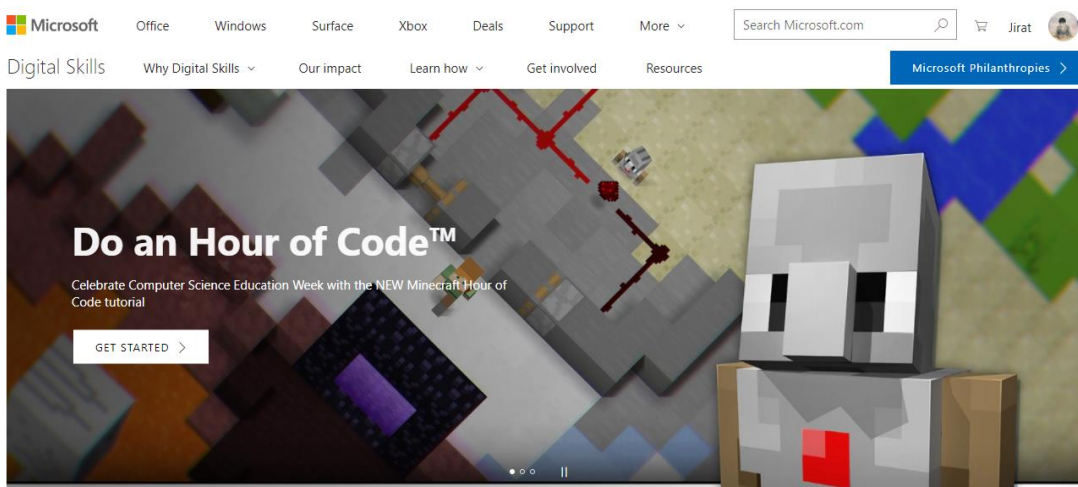
WE'RE MAKING A DIFFERENCE

25%	11M	750K	10%	40
of US students have accounts on Code.org	of our students are female	teachers use Code.org	of the world's students have tried the Hour of Code	states changed policy to support computer science



Microsoft Digital Skills

<https://www.microsoft.com/en-us/digital-skills>



Microsoft Office Windows Surface Xbox Deals Support More Search Microsoft.com Jirat

Digital Skills Why Digital Skills Our impact Learn how Get involved Resources Microsoft Philanthropies

Do an Hour of Code™

Celebrate Computer Science Education Week with the NEW Minecraft Hour of Code tutorial

GET STARTED >

Why digital skills?

From basic digital literacy to advanced computer science, digital skills are often out of reach for the young people who need them most. In a world being transformed by technology, all youth should have the opportunity to develop the creativity, critical thinking, and problem-solving skills gained by learning

edX Microsoft – Intro Python

<https://www.edx.org/course/introduction-python-absolute-beginner-microsoft-dev236x-1>



[Courses](#) ▾ [Programs](#) ▾ [Schools & Partners](#) [About](#) ▾

Search:

[Sign In](#) [Register](#)

[Home](#) > [All Subjects](#) > [Computer Science](#) > [Introduction to Python: Absolute Beginner](#)



Introduction to Python: Absolute Beginner

In this course that's perfect for true beginners, learn Python basics and start coding right away.



Self-Paced

[Enroll Now](#)

☐ I would like to receive email from Microsoft and learn about other offerings related to Introduction to Python: Absolute Beginner.

This course is part of a
Professional Certificate Program

About this course

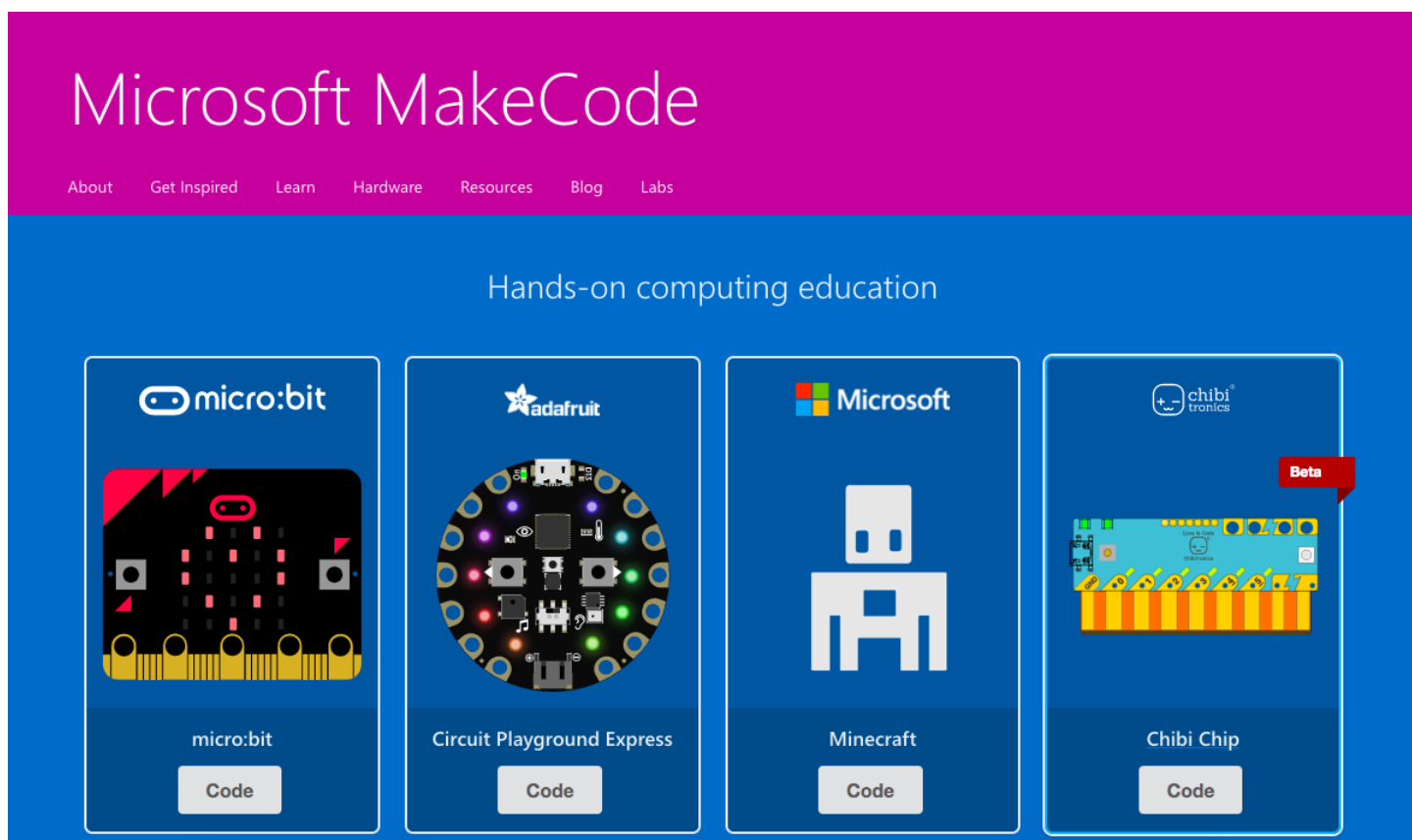
Brand new to text-based programming? Check out this hands-on course for an in-depth look at the details of Python layers and concepts. Get ample practice drills and projects, using Jupyter Notebooks on Azure, which require only a browser and an Internet connection. Learn best practices and begin coding almost immediately.

After you explore data types and variables, take a look at strings, input, testing, and formatting. From there, learn about arguments and parameters, along with conditionals and nested conditionals. By the end of the course, you'll be able to create programs that prompt users for input and use conditional (True/False) logic and Python methods to manipulate numbers and text to provide responses to the users, in addition to requesting further input. Plus, learn basic troubleshooting for your code. Sign up, and get started coding right away!

	Length:	5 weeks
	Effort:	3 to 4 hours per week
	Price:	FREE Add a Verified Certificate for \$99 USD
	Institution:	Microsoft
	Subject:	Computer Science
	Level:	Introductory

Microsoft Makecode

<https://makecode.com/>



Installer

ตัวติดตั้งภาษา Python

สามารถดาวน์โหลดได้จาก [Python.org/downloads](https://python.org/downloads) และเลือกเป็น Python 3.6

หรือคลิกที่ลิงก์นี้ <https://www.python.org/ftp/python/3.6.4/python-3.6.4.exe>