

01076105, 01076106

Object Oriented Programming

Object Oriented Programming Project

**From C to Python**

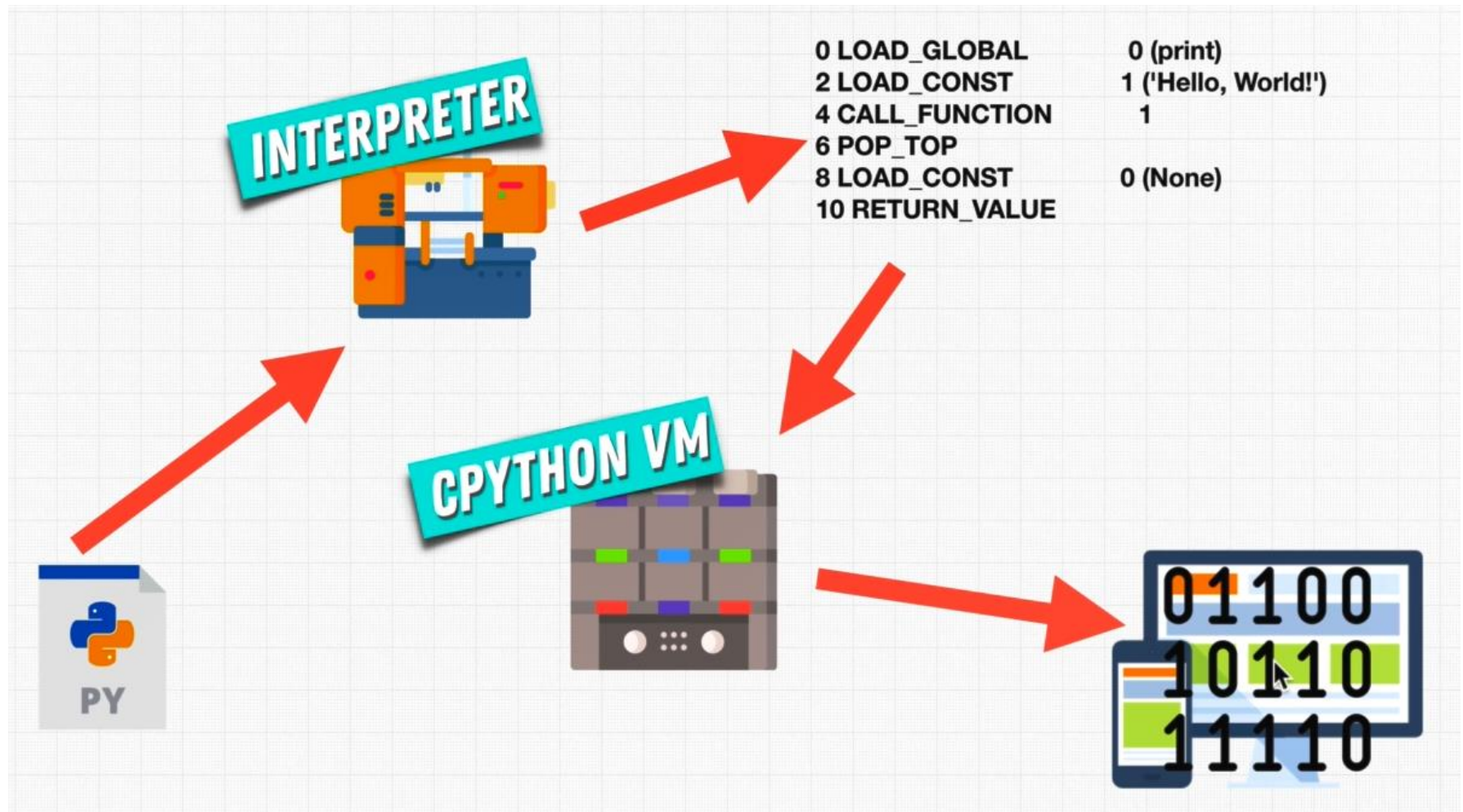
# Differences Between C and Python

Comparison Parameter	C	Python
Programming model	Procedural programming language	Object oriented programming language.
Type of language	Compilation	Interpretation
Speed	Faster	Slower
Memory Management	Manually	Automatically handled
Robustness	Less robust	More robust

# Differences Between C and Python

Comparison Parameter	C	Python
Applications	Hardware applications	General purpose
Built-in functions	Very limited	A lot
Usage	Syntax is harder than Python	Easier to write a code in Python
variables	Strong Type	Dynamic Type
Pointer	Yes	No

# การทำงานของ Interpreter



# From C to Python

```
#include <stdio.h>

int main(void) {
    int n;
    n = 10;

    for (int row = 0; row < n; row++) {
        for (int col = 0; col < n; col++) {
            if (row + col < n) {
                printf(" ");
            } else {
                printf("#");
            }
        }
        printf("#\n");
    }
}
```

```
# Draw Triangle
n = 10

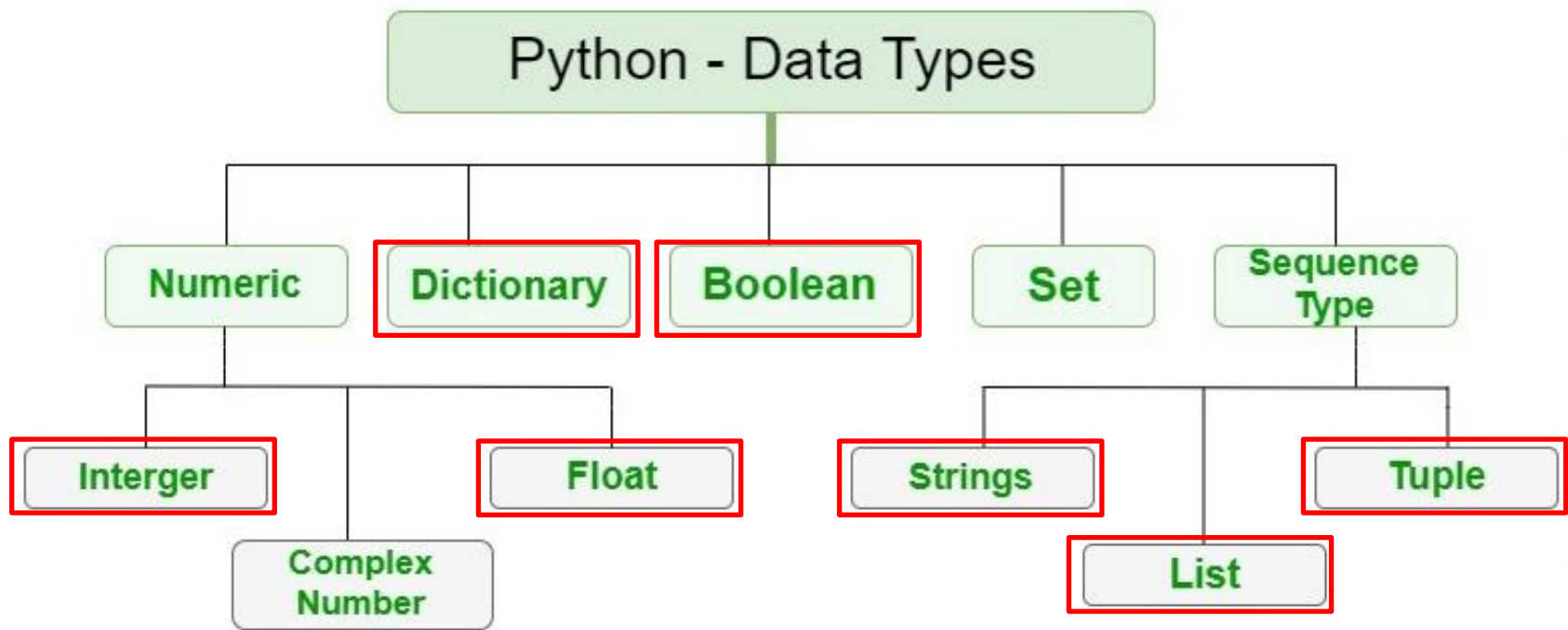
for row in range(n):
    for col in range(n):
        if row+col < n:
            print(' ', end='')
        else:
            print("#", end='')
    print('#')
```

# From C to Python

- Code Block ของ C จะใช้เครื่องหมาย { และ } แต่ Python จะใช้วิธี Indent
- การ Indent ในภาษา Python จึง Strict มาก
- ในภาษา Python มี Comment 2 แบบ
  - comment แบบ 1 บรรทัด ขึ้นต้นด้วย #  
# Say hello to everyone.  
  
print("Hello Python people!")
  - comment แบบหลายบรรทัด ขึ้นต้นด้วย """

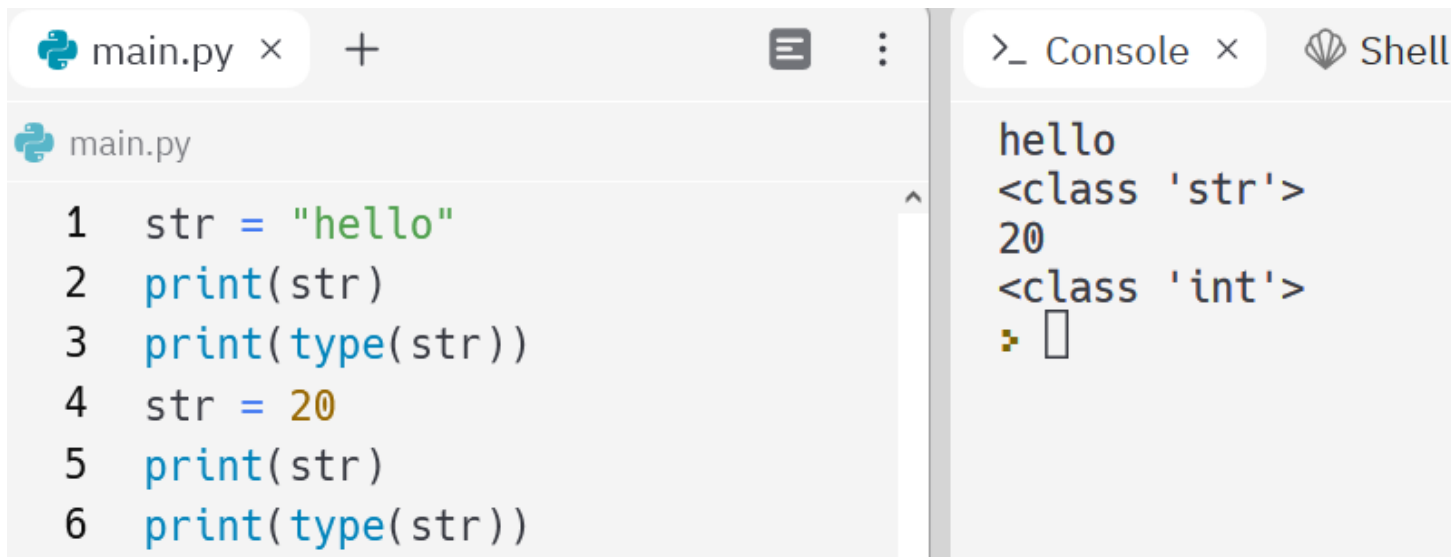
# From C to Python

- ข้อมูลในภาษา Python มีประเภทดังนี้ (ในกรอบ หมายถึง ใช้บ่อย)



# From C to Python

- ตัวแปรในภาษา Python จะต่างจากภาษา C โดยภาษา Python ตัวแปร คือ ตัวชี้ไปยังหน่วยความจำที่เก็บข้อมูลนั้นๆ ไม่ใช่ตัวข้อมูลเอง
- จากรูปจะเห็นว่าตัวแปร ชี้ไปยังข้อมูลใดก็ได้
- จะเรียกตัวแปรแบบนี้ว่า duck typing หรือ dynamic typing ซึ่งตรงข้ามกับ Strong Typing หรือ Static Typing



The screenshot shows a Python IDE with two panels. The left panel, titled 'main.py', contains the following code:

```
1 str = "hello"
2 print(str)
3 print(type(str))
4 str = 20
5 print(str)
6 print(type(str))
```

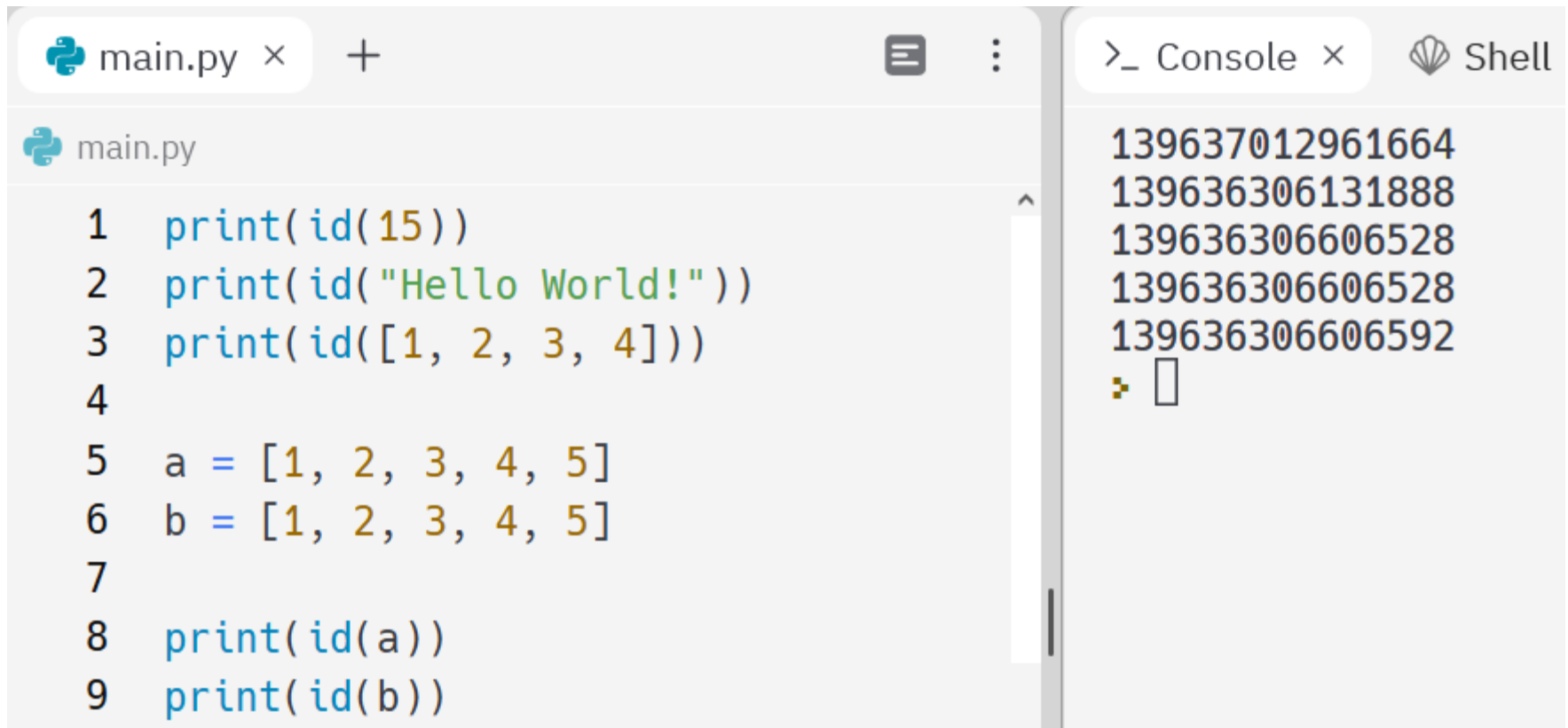
The right panel, titled 'Console', shows the output of the script:

```
hello
<class 'str'>
20
<class 'int'>
>
```



# From C to Python

- เราสามารถใช้ฟังก์ชัน `id()` ในการหา id หรือ Address ของข้อมูลได้



The screenshot shows a Python IDE with a file named `main.py` and a console window. The script in `main.py` is as follows:

```
1 print(id(15))
2 print(id("Hello World!"))
3 print(id([1, 2, 3, 4]))
4
5 a = [1, 2, 3, 4, 5]
6 b = [1, 2, 3, 4, 5]
7
8 print(id(a))
9 print(id(b))
```

The console window displays the output of the script, showing the memory addresses for each object:

```
139637012961664
139636306131888
139636306606528
139636306606528
139636306606592
>
```

# From C to Python

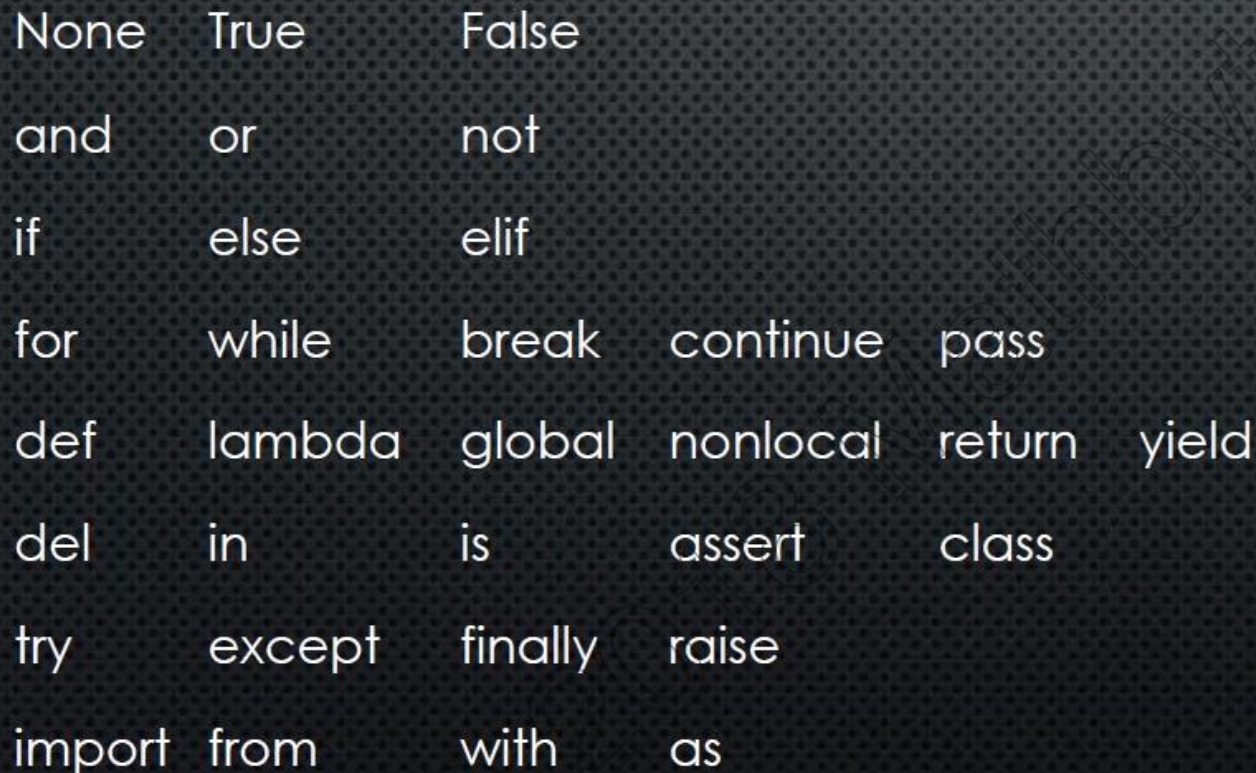
- ข้อมูลใดที่มีการกำหนด หากไม่มีตัวแปรชี้ python จะ clear ข้อมูลนั้นออก เพื่อคืนพื้นที่หน่วยความจำให้กับระบบ
- ตัวแปรหนึ่ง อาจถูกเปลี่ยนไปชี้ข้อมูลในตำแหน่งอื่นก็ได้ เมื่อมีการเปลี่ยนค่า และอาจชี้ไปที่เดียวกัน หากเป็นข้อมูลเดียวกัน

```
main.py
1  num = 1
2  print(id(num))
3  num = 200
4  print(id(num))
5  str1 = "Hello"
6  str2 = "Hello"
7  print(id(str1))
8  print(id(str2))
9  str1 = "Hello, CE"
10 print(id(str1))
11 print(id(str2))
```

```
139821457655744
139821457662112
139820683951088
139820683951088
139820683951280
139820683951088
✖
```

# From C to Python

- ข้อมูลใน Python จะไม่มีขนาด เพราะ Interpreter สามารถขยายขนาดข้อมูลได้
- ชื่อตัวแปรต้องไม่เป็น reserve word



None	True	False			
and	or	not			
if	else	elif			
for	while	break	continue	pass	
def	lambda	global	nonlocal	return	yield
del	in	is	assert	class	
try	except	finally	raise		
import	from	with	as		

# From C to Python

- การตั้งชื่อตัวแปร จะมีหลักนิยมอยู่ 3 แบบ
  - Snake Case จะใช้คำภาษาอังกฤษเขียนเป็นตัวเล็กทั้งหมด แล้วคั่นด้วยเครื่องหมาย `_` เช่น `sum_of_student_score` จะเห็นว่าใช้คำเขียนต่อกันคล้ายงู จึงเรียกว่า snake case
  - Camel Case จะใช้ตัวอักษรตัวใหญ่เฉพาะตัวอักษรแรกของคำ “ยกเว้น” คำแรก เช่น `sumOfStudentScore`
  - Pascal Case จะใช้ตัวอักษรตัวใหญ่เฉพาะตัวอักษรแรกของคำ โดยไม่มียกเว้น เช่น `SumOfStudentScore`
  - สำหรับวิชานี้ให้ใช้ Snake Case เพราะอ่านง่ายที่สุด
- ตัวอักษรตัวใหญ่เล็ก ถือว่าเป็นคนละตัวกัน (Case Sensitive)
- ตัวแรกต้องเป็นตัวอักษร หรือ `_` เท่านั้น ตัวถัดมาเป็นตัวอักษร ตัวเลข และ `_`

# From C to Python

- ข้อมูลประเภท Integer และ Float จะมี Operator ดังนี้

Operators	Meaning	Example	Result
+	Addition	$4 + 2$	6
-	Subtraction	$4 - 2$	2
*	Multiplication	$4 * 2$	8
/	Division	$4 / 2$	2
%	Modulus operator to get remainder in integer division	$5 \% 2$	1
**	Exponent	$5 ** 2 = 5^2$	25
//	Integer Division/ Floor Division	$5 // 2$ $-5 // 2$	2 -3

# From C to Python

- ลำดับการทำงานของ Operator ทางคณิตศาสตร์

Operator	Description	Associativity
()	Parentheses	left-to-right
**	Exponent	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right

# From C to Python

- ใน Python ไม่มีชนิดข้อมูล Char มีแต่ชนิดข้อมูล String
- String จะใช้เครื่องหมาย " หรือ ' ครอบเอาไว้ โดยเครื่องหมายเปิดและปิด ต้องเป็นเครื่องหมายเดียวกัน และใน String ต้องไม่มีเครื่องหมายแบบเดียวกัน
- สามารถใช้เครื่องหมาย "" "" ครอบ String ใดๆ ได้ (multiline)

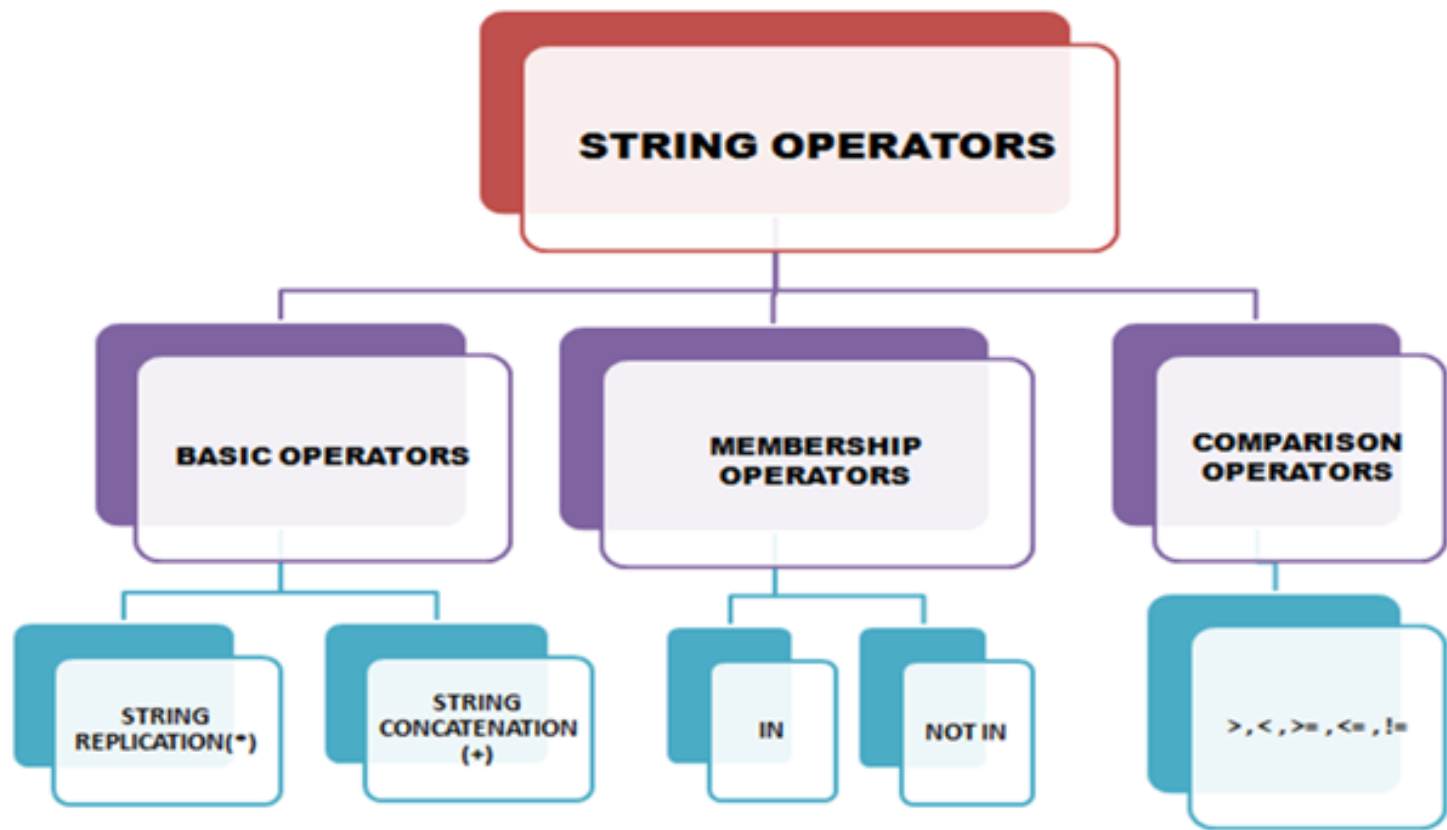
 main.py

```
1 print("Hello Python world!")
2 print('Hello KMITL!')
3 print("King's Mongkut")
4 print('Hello "John"')
5 print("""Hello 'John'""")
6
```

```
Hello Python world!
Hello KMITL!
King's Mongkut
Hello "John"
Hello 'John'
>
```

# From C to Python

- ข้อมูลประเภท String มี Operator ดังนี้





# From C to Python

- ตัวอย่าง Operator ของ String



The image shows a Python IDE with a script editor on the left and a console on the right. The script editor contains a file named `main.py` with the following code:

```
1 print("Hello " + "World")
2 print("CE " * 5)
3 print("H" in "Hello")
4 print("H" not in "Hello")
5 print("H" > "Hello")
6
```

The console on the right shows the output of the script:

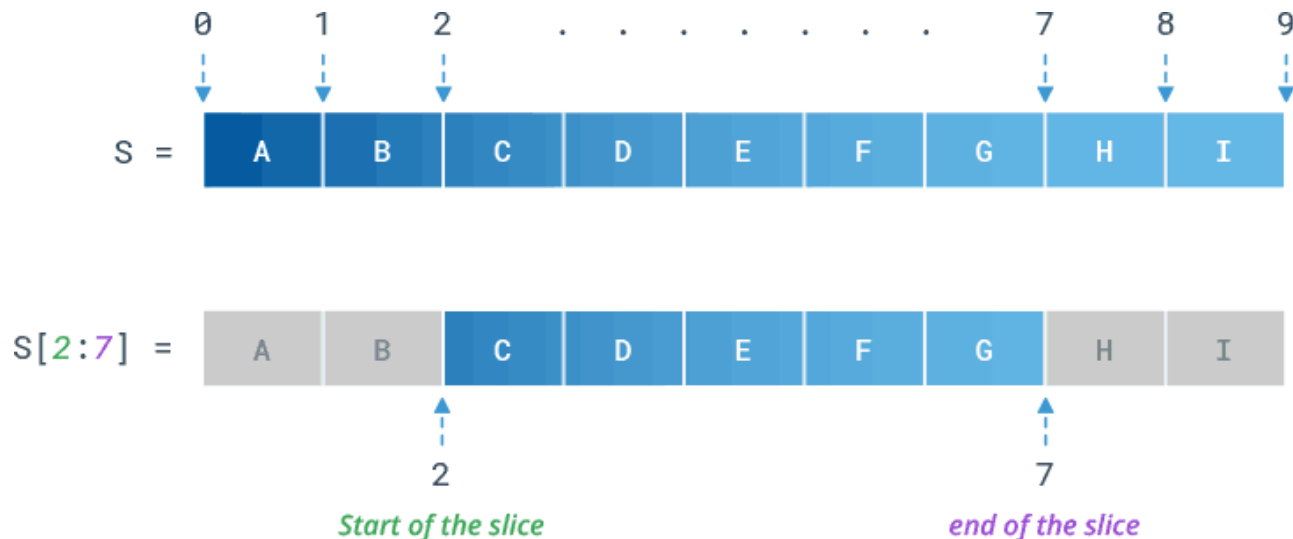
```
>_ Console x Shell x
Hello World
CE CE CE CE CE
True
False
False
>
```

# From C to Python

- ใน Python จะมอง String คล้ายกับ Array ของ C โดยสามารถระบุ Index ได้
- นอกจากนั้นยังสามารถทำ Slicing ได้

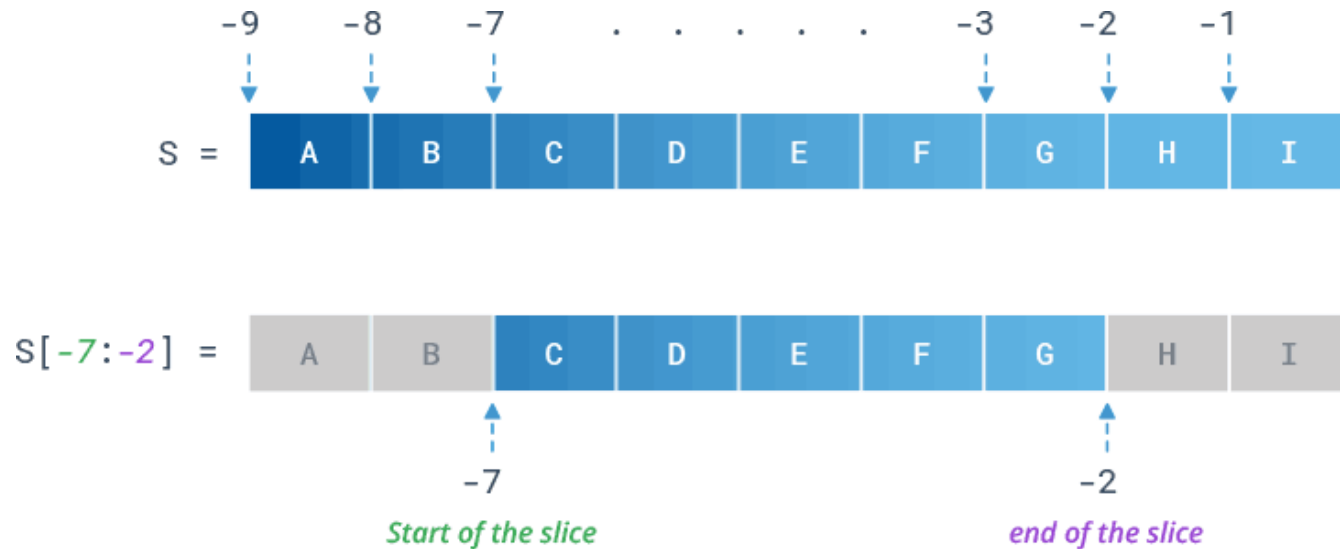
`S[start:stop:step]`

*Start position    End position    The increment*



# From C to Python

- โดย Index สามารถเป็นลบได้



# From C to Python

- ตัวอย่างของ String Slicing

main.py x

```
1 #Accessing string characters in Python
2 str = 'computer'
3 print('str = ', str)
4
5 #first character
6 print('str[0] = ', str[0])
7
8 #last character
9 print('str[-1] = ', str[-1])
10
11 #slicing 2nd to 5th character
12 print('str[1:5] = ', str[1:5])
13
14 #slicing 6th to 2nd last character
15 print('str[5:-2] = ', str[5:-2])
```

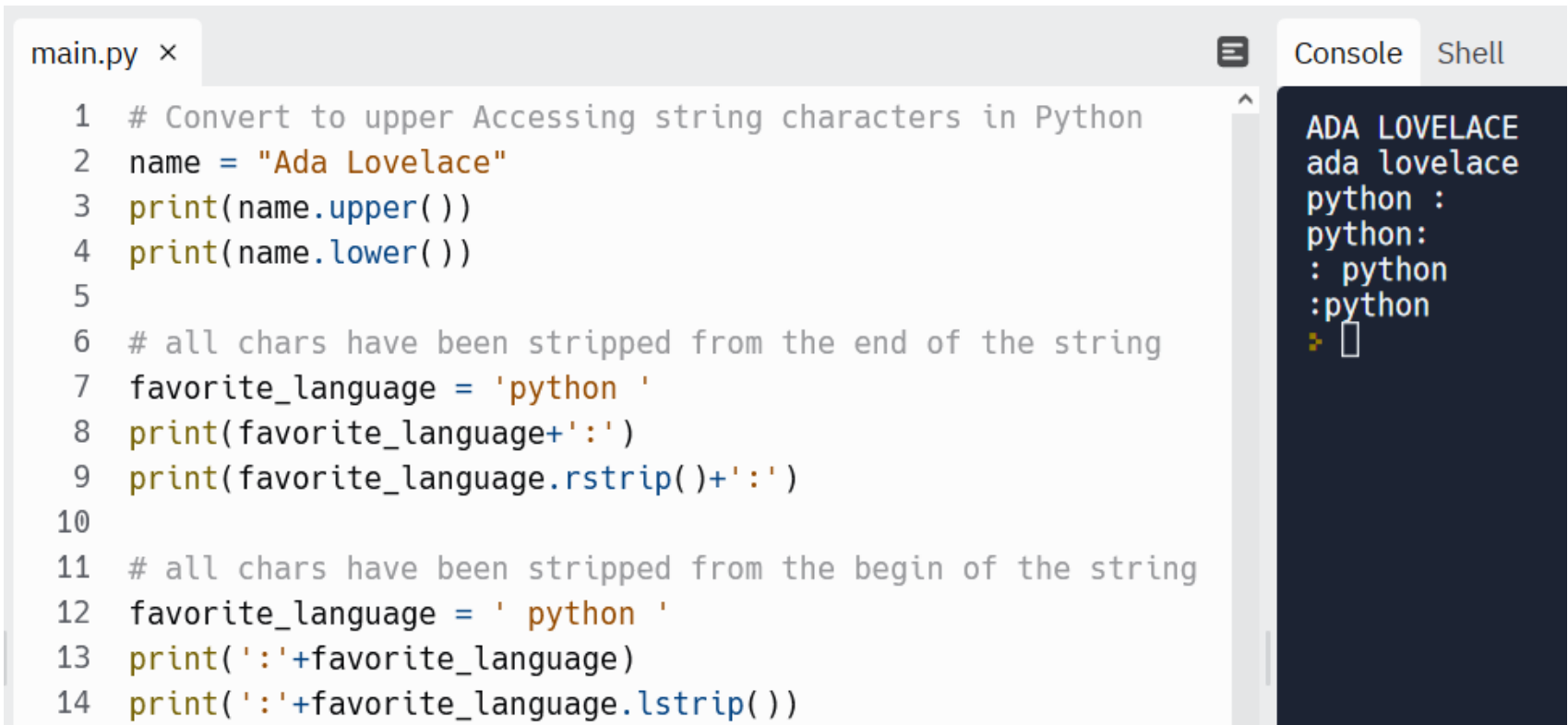
Console

Shell

```
str = computer
str[0] = c
str[-1] = r
str[1:5] = ompu
str[5:-2] = t
>
```

# From C to Python

- สิ่งต่างจาก Array คือ String ไม่สามารถแก้ไขได้
- เนื่องจาก String เป็น Object จึงมี Function (Method) ให้ใช้มากมาย วิธีการใช้ Function ของ String จะใช้ .function หลังตัวแปร



The screenshot shows a Python IDE with a file named 'main.py'. The code in the editor demonstrates string methods: `upper()` and `lower()` for case conversion, and `rstrip()` and `lstrip()` for stripping whitespace. The output in the console shows the results of these operations on the string 'Ada Lovelace' and 'python'.

```
main.py x
1 # Convert to upper Accessing string characters in Python
2 name = "Ada Lovelace"
3 print(name.upper())
4 print(name.lower())
5
6 # all chars have been stripped from the end of the string
7 favorite_language = 'python '
8 print(favorite_language+':')
9 print(favorite_language.rstrip()+':')
10
11 # all chars have been stripped from the begin of the string
12 favorite_language = ' python '
13 print(':'+favorite_language)
14 print(':'+favorite_language.lstrip())
```

Console

```
ADA LOVELACE
ada lovelace
python :
python:
: python
:python
: 
```

# From C to Python

- ฟังก์ชันอื่นๆ ของ String
  - isupper ตัวใหญ่หรือไม่
  - islower ตัวเล็กหรือไม่
  - isnumeric ตัวเลขหรือไม่
  - isalpha ตัวอักษรหรือไม่
  - isalnum Alphanumeric หรือไม่
  - count นับจำนวนของอักษร
  - find หาดำแหน่งตัวอักษรหรือชุด (หาตัวที่ n ได้ด้วย)
  - startswith เริ่มต้นด้วย
  - endswith ปิดท้ายด้วย
  - replace เปลี่ยนข้อความ
  - split แยก String และส่งกลับใน List
  - join รวม List เป็น String (ข้อมูลใน List ต้องเป็น String)

# From C to Python

- Boolean เป็นชนิดข้อมูลหนึ่งของ Python
- มีค่าเพียง 2 ค่า คือ True และ False
- ค่า True สามารถตีความเป็นตัวเลขได้ โดยมีค่า 1 (อะไรที่  $\neq 0$  เป็น True หมด)
  - เช่น ถ้ามีประโยค `a = 1 == True` ; a จะมีค่า = True
  - ประโยค `a = "1" == True` มีค่าเป็นอะไร?
- สำหรับค่า False ถ้าตีความเป็นตัวเลข มีค่า 0
  - เช่น ถ้ามีประโยค `a = 0 == False` ; a จะมีค่า = True

# From C to Python

- การกระทำแบบเปรียบเทียบ จะให้ผลลัพธ์เป็น Boolean

Operator	Meaning
<b>==</b>	Equal to
<b>!=</b>	Not equal to
<b>&gt;</b>	Greater than
<b>&lt;</b>	Less than
<b>&gt;=</b>	Greater than or equal to
<b>&lt;=</b>	Less than or equal to



# From C to Python

- การกระทำแบบเปรียบเทียบ จะให้ผลลัพธ์เป็น Boolean

main.py ×

```
1 spam = True
2 print(spam)
3 print(42 == 42)
4 print(42 == 99)
5 print(2 != 3)
6 print(2 != 2,end='\n\n')
7
8 print('hello' == 'hello')
9 print('hello' == 'Hello')
10 print('dog' != 'cat')
11 print(True == True)
12 print(True != False)
13 print(42 == 42.0)
14 print(42 == '42')
15
```

Console Shell

```
True
True
False
True
False
```

```
True
False
True
True
True
True
False
>
```

# From C to Python

- การกระทำแบบเปรียบเทียบ ยังมีอีก 1 ตัว โดยใช้ตรวจสอบการเป็นสมาชิก ได้แก่ operator **in**
- เช่น 'a' in 'abc' จะได้ผลลัพธ์เป็น True
- การใช้ in จะมีบทบาทมากในการเขียนโปรแกรมต่อไปในอนาคต
- นอกจากนั้นยังมีการใช้แบบตรงกันข้าม คือ not in อีกด้วย

# From C to Python

- ในภาษา Python การกระทำแบบเปรียบเทียบสามารถเขียนในรูปแบบ “ลูกโซ่” หรือ Chained Comparison ได้
  - โดย  $a == b == c$   $\Rightarrow$   $a == b$  and  $b == c$
  - และ  $a < b < c$   $\Rightarrow$   $a < b$  and  $b < c$
- การเขียนในลักษณะ “ลูกโซ่” แบบนี้ จะทำให้ นิพจน์ สั้นลง
- เช่น หากต้องการทราบว่าตัวแปร  $a$  อยู่ระหว่าง 1 – 10 หรือไม่ ก็สามารถเขียนเป็น  $1 < a < 10$

# From C to Python

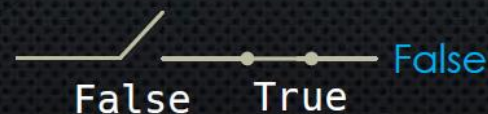
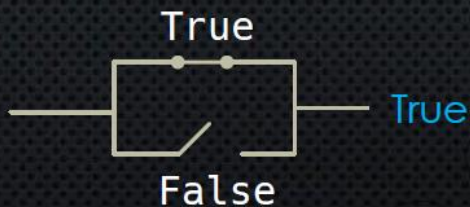
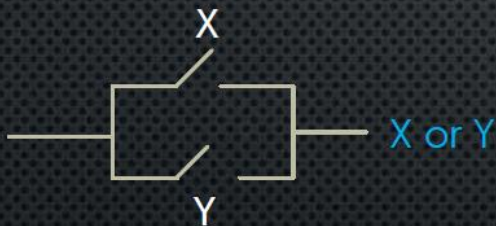
- การกระทำทาง Logic จะให้ผลลัพธ์เป็น Boolean

เช่นกัน

X	Y	not X	X and Y	X or Y
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

 open False

 closed True

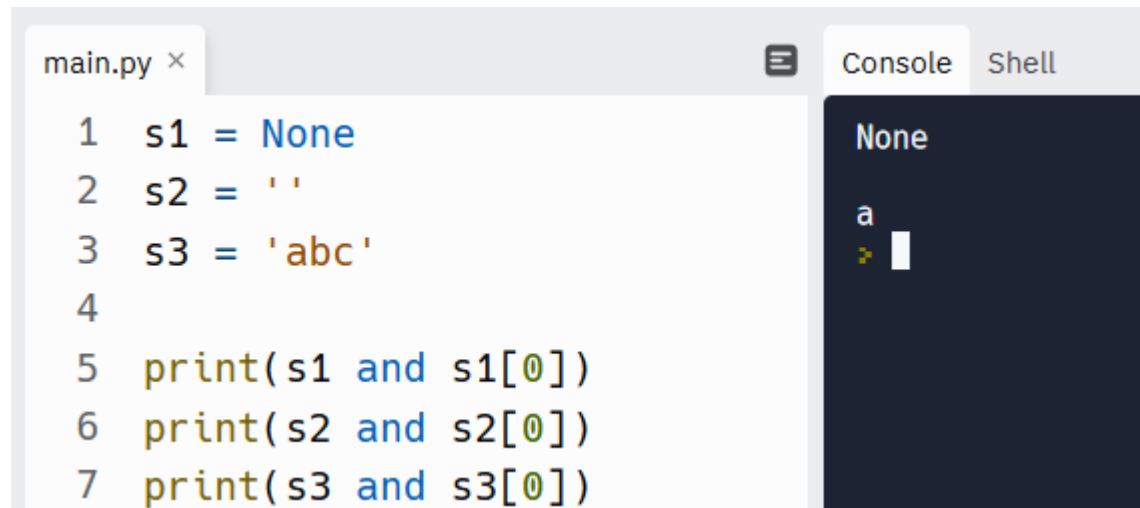


# From C to Python

- การทำงานของ AND คือ

`X and Y`: If X is falsy, returns X, otherwise evaluates and returns Y

- ถ้า X เป็น False จะคืนค่า X โดยไม่ประเมินค่า Y เลย แต่ถ้าเป็น True จึงประเมินค่า Y



The screenshot shows a Python IDE with a file named `main.py`. The code in the editor is as follows:

```
1 s1 = None
2 s2 = ''
3 s3 = 'abc'
4
5 print(s1 and s1[0])
6 print(s2 and s2[0])
7 print(s3 and s3[0])
```

The right-hand side of the IDE shows the 'Console' tab, which displays the output of the script:

```
None
a
>
```

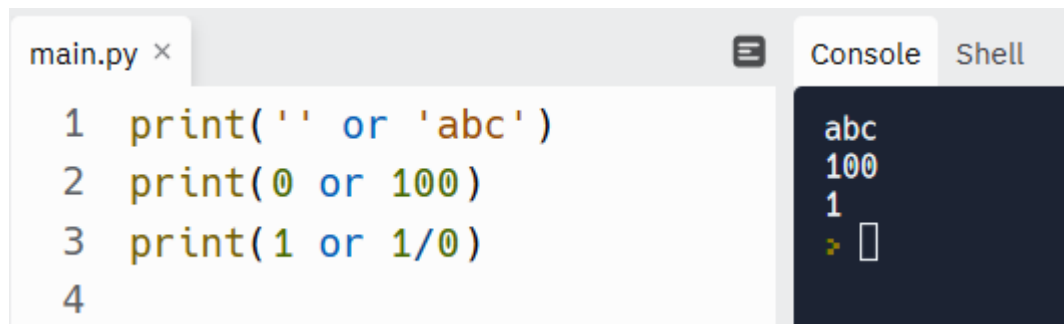
The output demonstrates the behavior of the `and` operator: for `s1` (None), it returns `s1`; for `s2` (empty string), it returns `s2`; and for `s3` ('abc'), it evaluates `s3[0]` and returns 'a'.

# From C to Python

- การทำงานของ OR คือ

`X or Y`: If X is falsy, returns Y, otherwise evaluates and returns X

- ถ้า X เป็น False จะประเมินค่า Y แต่ถ้าเป็น True จะประเมิน X



The screenshot shows a Python IDE with a file named 'main.py' open. The code in the editor is as follows:

```
1 print('' or 'abc')
2 print(0 or 100)
3 print(1 or 1/0)
4
```

To the right of the editor is a 'Console' tab. The output in the console is:

```
abc
100
1
>
```

The output shows that the 'or' operator returns the first truthy value it encounters. In the third line, '1' is truthy, so it is returned before the division '1/0' is evaluated, thus avoiding a ZeroDivisionError.

- ในบรรทัดที่ 3 ปกติ 1/0 จะ Error แต่กรณีนี้ไม่ Error เพราะไม่ทำงาน

# From C to Python

- Short Circuit Evaluation ใน Logical Operation
- จากที่กล่าวว่า การทำงานของ AND คือ

`X and Y`: If X is falsy, returns X, otherwise evaluates and returns Y

– ถ้า X เป็น False จะคืนค่า X โดยไม่ประเมินค่า Y เลย  
แต่ถ้าเป็น True จึงประเมินค่า Y

- และการทำงานของ OR คือ

`X or Y`: If X is falsy, returns Y, otherwise evaluates and returns X

- ถ้า X เป็น False จะประเมินค่า Y แต่ถ้าเป็น True จะประเมิน X

# From C to Python

- โปรแกรมนี้ จะมีปัญหาหาก b เป็น 0 (ตัวหารเป็น 0)

```
a = 10
b = 2

if a/b > 2:
    print('a is at least double b')
```

- แต่หากเขียนแบบนี้ จะสามารถแก้ปัญหาก็ได้

```
a = 10
b = 0

if b and a/b > 2:
    print('a is at least double b')
```



# From C to Python

- จะมีบางกรณีที่เราใช้ประโยชน์จาก **Short Circuit Evaluation**
- สมมติว่าเรามีระบบหนึ่ง เก็บข้อมูลจาก sensor จำนวนหนึ่ง โดยมีเงื่อนไขว่าหาก sensor ที่เราสนใจ ถ้ามีระดับสูงกว่าที่กำหนดให้แจ้งเตือน สามารถเขียนเป็นโปรแกรม

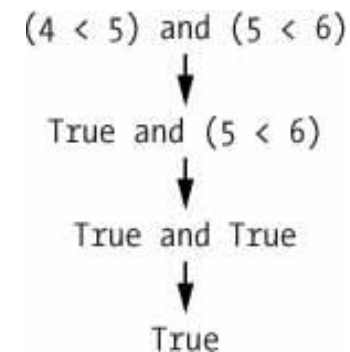
```
if sensor in watch_list:  
    if value(sensor) > threshold:  
        # do something
```

- จากเงื่อนไข AND จะเห็นว่า ถ้าเงื่อนไขแรกเป็น False จะไม่ทำงานในเงื่อนไขหลังเลย ดังนั้นหากเขียนแบบนี้ ก็ยังทำงานเหมือนเดิมทุกประการ

```
if sensor in watch_list and value(sensor) > threshold:  
    # do something
```

# From C to Python

- กรณีที่มีการทำงานทางลอจิกหลายๆ ตัวอยู่ด้วยกัน จะมีลำดับการทำงานดังนี้
  - ( ) ถ้ามีวงเล็บ ทำในวงเล็บก่อน
  - < > <= >= != in ลำดับที่ 2
  - not ลำดับที่ 3
  - and ลำดับที่ 4
  - or ลำดับที่ 5

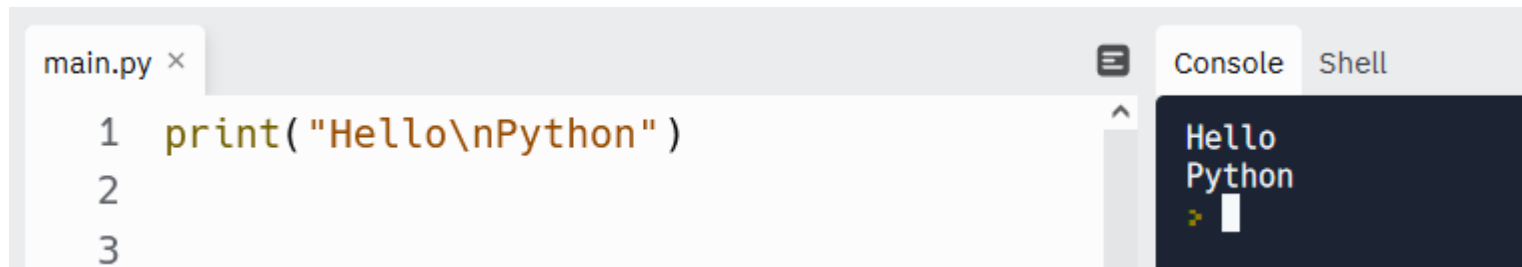


# From C to Python

Operator	Description	
( )	Parentheses (grouping)	Highest   Lowest
**	Exponentiation	
~x	Bitwise nor	
+x,-x	Positive , negative (unary +,-)	
*,/,//,%	Multiplication , division , floor division , remainder	
+, -	Addition , subtraction	
&	Bitwise and	
^	Bitwise XOR	
	Bitwise OR	
<,<=,>,>=,<>,!','=', ,is,isnot	Comparisons (Relational operators),identity operators	
not x	Boolean NOT	
And	Boolean AND	
or	Boolean OR	
		Lowest

# From C to Python

- ในการแสดงผลจะใช้คำสั่ง print โดยกรณีที่ต้องการให้ขึ้นบรรทัดใหม่ จะใช้อักขระพิเศษ \n



The screenshot shows a code editor window titled 'main.py' with the following code:

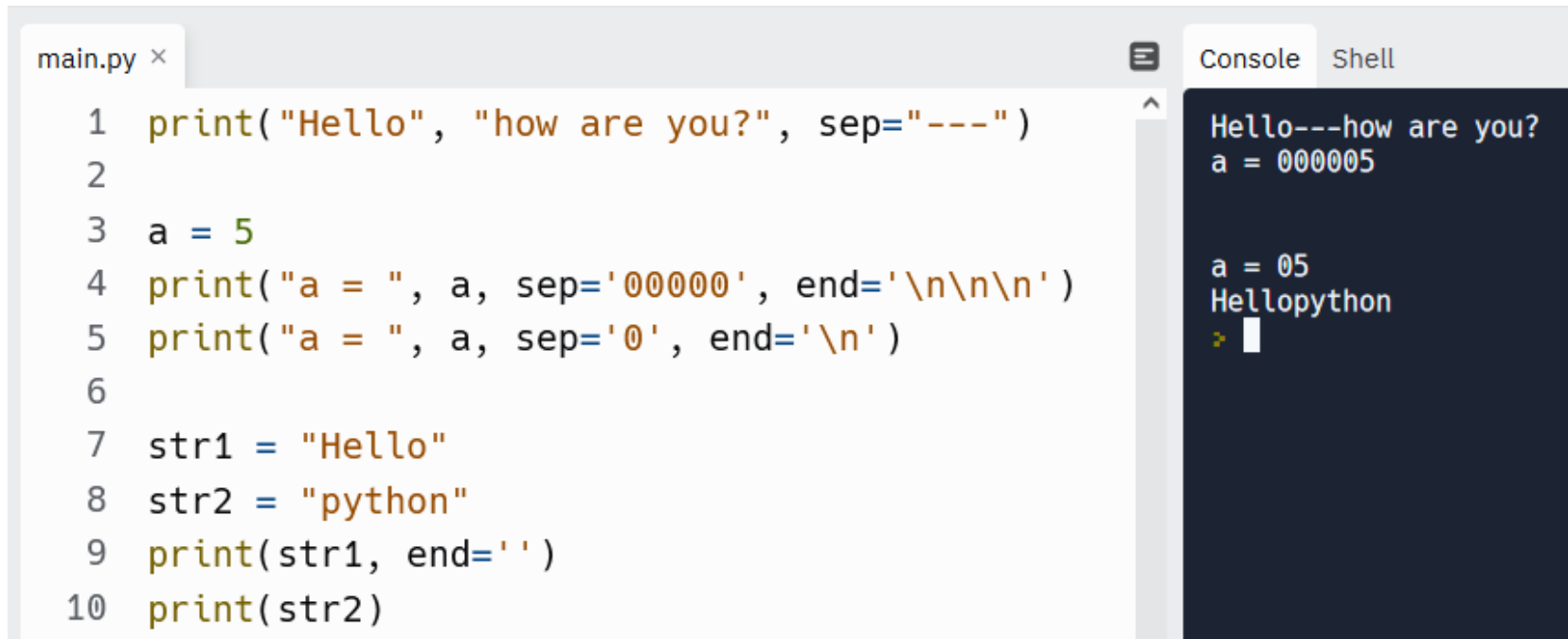
```
1 print("Hello\nPython")
2
3
```

To the right of the code editor is a 'Console' window. It displays the output of the code: 'Hello' on the first line and 'Python' on the second line, with a cursor on the third line. This demonstrates the effect of the '\n' escape character.

- ปกติคำสั่ง print จะขึ้นบรรทัดใหม่เสมอ แต่สามารถกำหนดให้ปิดท้าย String ด้วยอักขระอื่นได้ โดยใช้ end=' ' ดังนั้นถ้าใช้ end='' คือไม่ต้องขึ้นบรรทัดใหม่
- ปกติหากมีการพิมพ์หลายค่าในคำสั่งเดียว print จะแทรก space คั่นระหว่างค่า แต่สามารถกำหนดให้ใช้อย่างอื่นแทนได้ โดยใช้ sep=

# From C to Python

- ตัวอย่างการใช้ end และ sep ในคำสั่ง print



The screenshot shows a Python IDE with a file named `main.py` and a console window. The script in `main.py` is as follows:

```
1 print("Hello", "how are you?", sep="---")
2
3 a = 5
4 print("a = ", a, sep='00000', end='\n\n\n')
5 print("a = ", a, sep='0', end='\n')
6
7 str1 = "Hello"
8 str2 = "python"
9 print(str1, end='')
10 print(str2)
```

The console output shows the results of the script:

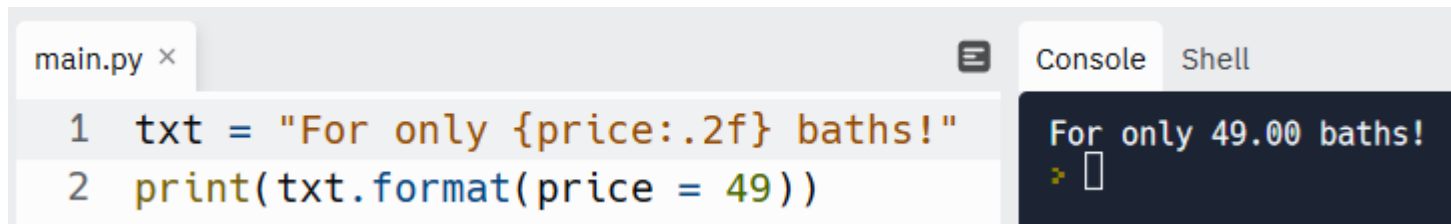
```
Hello---how are you?
a = 000005

a = 05
Hellopython
>
```

- sep คือ ใช้อย่างอื่นแทนช่องว่างระหว่างการแสดงผล
- end คือ ปิดท้ายบรรทัดด้วยอะไร ถ้าไม่บอก คือ การขึ้นบรรทัดใหม่

# From C to Python

- ในบางครั้งเราต้องการแสดงผลในรูปแบบที่เราต้องการ เราสามารถจัดรูปแบบการแสดงผลโดยใช้ `.format` เช่น



The screenshot shows a code editor window titled 'main.py' with two lines of Python code. To the right of the editor is a 'Console' tab showing the output of the code. The code defines a string with a placeholder for a price and then prints it with the value 49. The output shows the string with the price formatted to two decimal places.

```
main.py ×  
1 txt = "For only {price:.2f} baths!"  
2 print(txt.format(price = 49))
```

Console Shell  
For only 49.00 baths!  
>

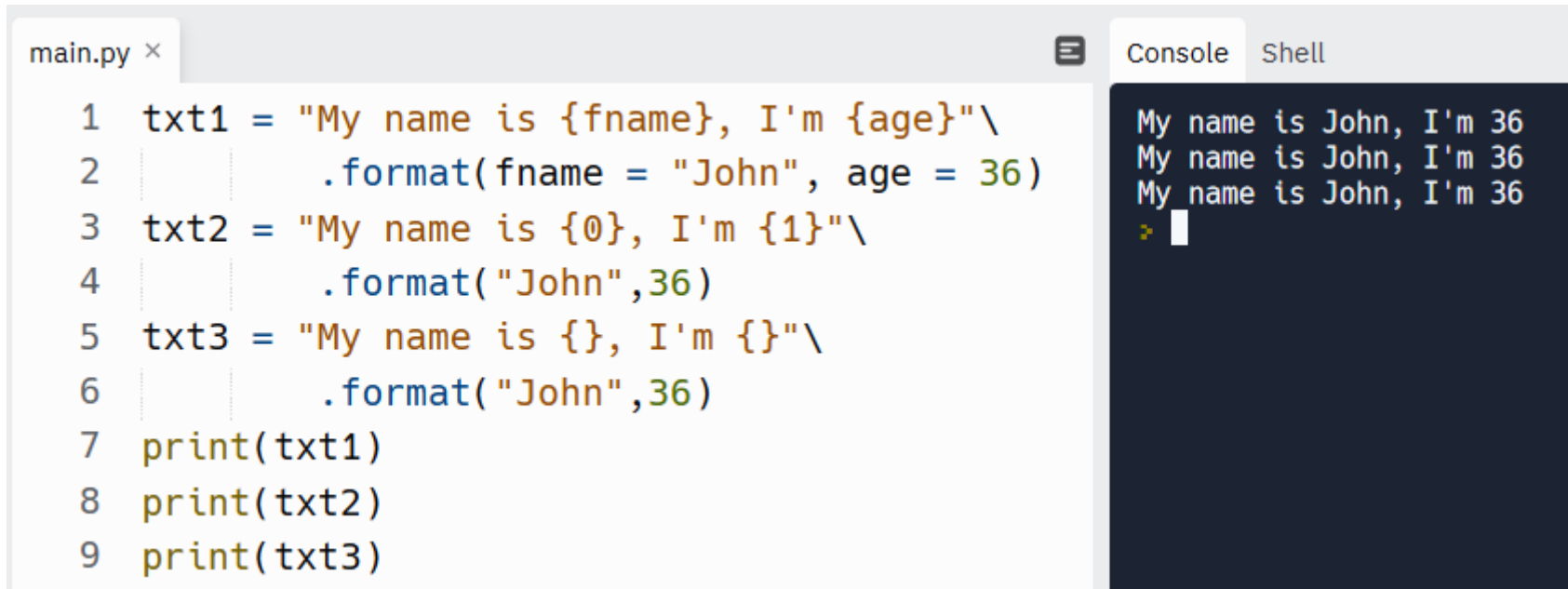
- รูปแบบการใช้งานเป็นดังนี้

`string.format(value1, value2...)`

- ส่วนที่ต้องการให้จัดรูปแบบจะอยู่ข้างใน { } และส่วนข้อมูลที่ต้องการแสดงผลและจัดรูปแบบจะอยู่ในวงเล็บของ `.format( )`

# From C to Python

- การระบุการจับคู่ระหว่างรูปแบบ กับ ข้อมูล สามารถทำได้หลายรูปแบบดังนี้



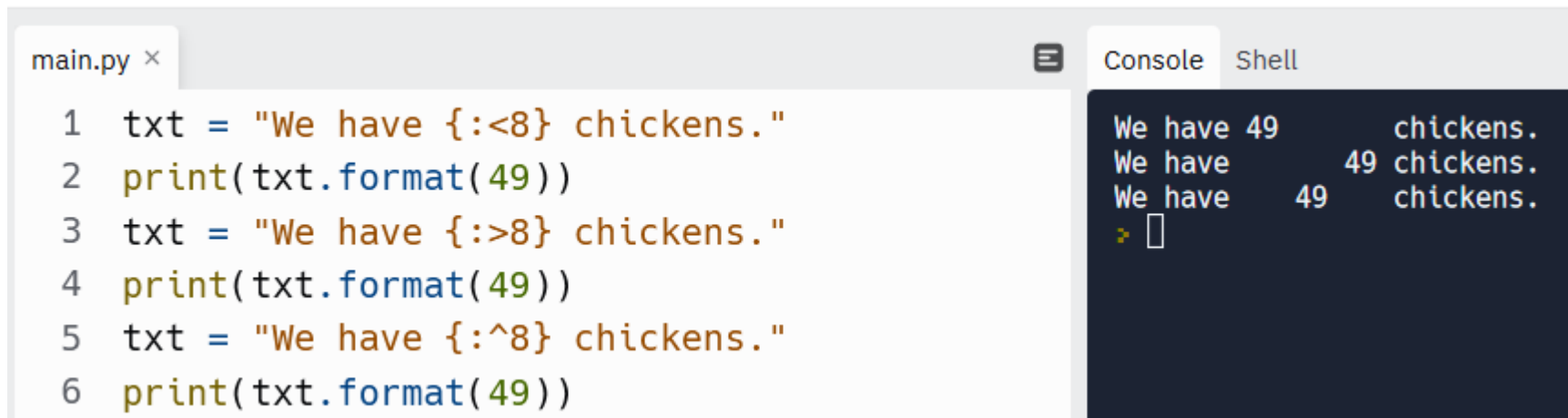
```
main.py x
1 txt1 = "My name is {fname}, I'm {age}"\
2     .format(fname = "John", age = 36)
3 txt2 = "My name is {0}, I'm {1}"\
4     .format("John",36)
5 txt3 = "My name is {}, I'm {}"\
6     .format("John",36)
7 print(txt1)
8 print(txt2)
9 print(txt3)
```

Console Shell

```
My name is John, I'm 36
My name is John, I'm 36
My name is John, I'm 36
>
```

# From C to Python

- ข้างในเครื่องหมาย { } เราสามารถใส่สัญลักษณ์จัดรูปแบบได้
  - :< จัดชิดซ้าย
  - :> จัดชิดขวา
  - :^ จัดตรงกลาง



The screenshot shows a Python IDE with a file named `main.py`. The code in the editor is as follows:

```
1 txt = "We have {:<8} chickens."  
2 print(txt.format(49))  
3 txt = "We have {:>8} chickens."  
4 print(txt.format(49))  
5 txt = "We have {:^8} chickens."  
6 print(txt.format(49))
```

To the right of the editor is a console window. It displays the output of the code: the number 49 is formatted to be left-aligned, right-aligned, and center-aligned within an 8-character field. The prompt character is a yellow greater-than sign.

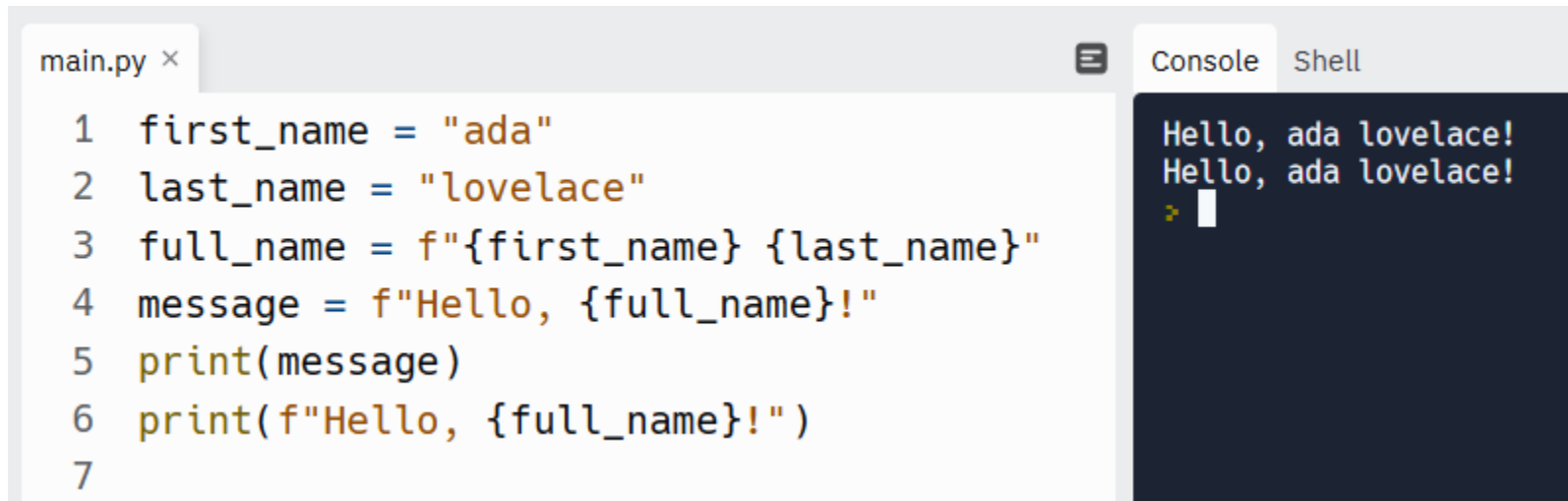
```
We have 49      chickens.  
We have          49 chickens.  
We have      49      chickens.  
> 
```

- รูปแบบอื่น [https://www.w3schools.com/python/ref\\_string\\_format.asp](https://www.w3schools.com/python/ref_string_format.asp)



# From C to Python

- การจัดรูปแบบอีกแบบหนึ่งเรียกว่า f-string เป็นการรวมเอาส่วนการจัดรูปแบบและส่วนของข้อมูลเอาไว้ด้วยกัน



The screenshot shows a code editor window titled 'main.py' with the following Python code:

```
1 first_name = "ada"
2 last_name = "lovelace"
3 full_name = f"{first_name} {last_name}"
4 message = f"Hello, {full_name}!"
5 print(message)
6 print(f"Hello, {full_name}!")
7
```

To the right of the code editor is a 'Console' window. It displays the output of the code:

```
Hello, ada lovelace!
Hello, ada lovelace!
>
```

# From C to Python

- ในการพิมพ์เราสามารถจัดรูปแบบการพิมพ์ได้ เพื่อความสวยงาม

main.py ×

```
1 print("{}{}".format("th", "Thailand"))
2 print("{:5}|{:15}|".format("th", "Thailand")) # align left
3 print("{:<5}|{:<15}|".format("th", "Thailand")) # align left
4 print("{:>5}|{:>15}|".format("th", "Thailand")) # align right
5 print("{:*>5}|{:>-15}|".format("th", "Thailand")) # align right
6 print("{:^5}|{:^15}|".format("th", "Thailand")) # align center
7
```



Console

Shell

```
thThailand
th      |Thailand      |
th      |Thailand      |
      th|          Thailand|
***th|-----Thailand|
      th|          Thailand |
> []
```

— :15 = พื้นที่แสดง 15 ช่อง

— < = ซิดซ้าย

— > = ซิดขวา

— ^ = ตรงกลาง

# From C to Python

- ในการพิมพ์เราสามารถจัดรูปแบบการพิมพ์ได้ เพื่อความสวยงาม

```
main.py x
1 print(f'{"th"}{"Thailand"}')
2 print(f'{"th":5}|{"Thailand":15}|')
3 print(f'{"th":<5}|{"Thailand":<15}|')
4 print(f'{"th":>5}|{"Thailand":>15}|')
5 print(f'{"th":*>5}|{"Thailand":->15}|')
6 print(f'{"th":^5}|{"Thailand":^15}|')
7
```

Console Shell

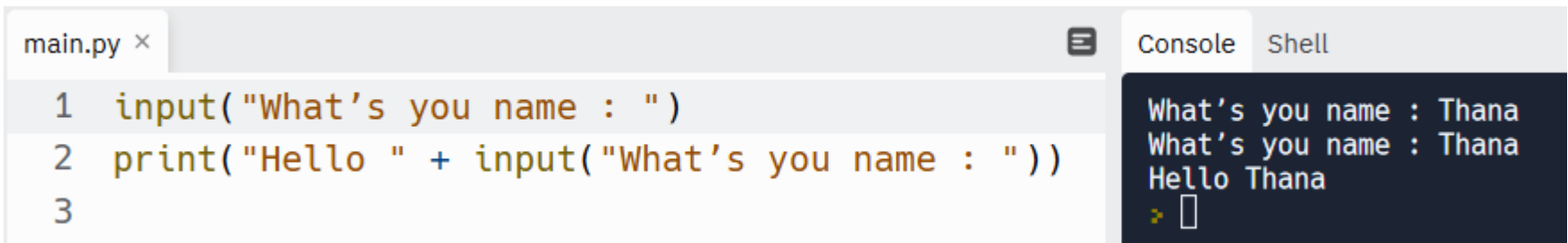
```
thThailand
th |Thailand |
th |Thailand |
  th|      Thailand|
***th|-----Thailand|
  th |      Thailand |
>
```

# From C to Python

- การ Input ในภาษา Python จะมีรูปแบบการใช้งาน (Syntax) ดังนี้

```
input(prompt)
```

- โดย prompt คือ ข้อความที่แสดงว่าต้องการให้ Input ข้อมูลอะไร

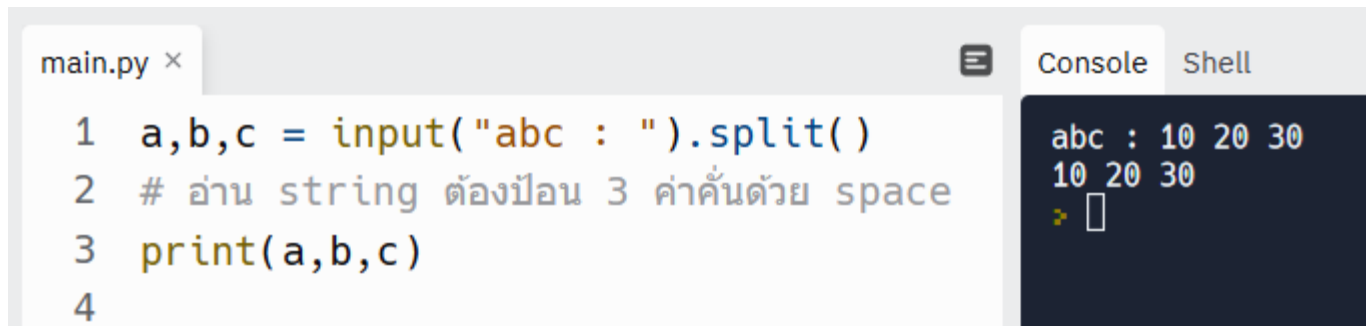


The screenshot shows a code editor with a file named 'main.py'. The code contains two lines: `input("What's you name : ")` and `print("Hello " + input("What's you name : "))`. To the right, there is a 'Console' tab showing the output of the script. The first line of the script prompts for a name, and the second line prompts again and prints 'Hello ' followed by the input. The output in the console shows 'What's you name : Thana' twice, followed by 'Hello Thana'.

```
main.py ×  
1 input("What's you name : ")  
2 print("Hello " + input("What's you name : "))  
3  
Console Shell  
What's you name : Thana  
What's you name : Thana  
Hello Thana  
> □
```

# From C to Python

- เราสามารถรับ Input ครึ่งละหลายค่าได้



The screenshot shows a code editor with a file named 'main.py'. The code contains three lines: `1 a,b,c = input("abc : ").split()`, `2 # อ่าน string ต้องป้อน 3 ค่าคั่นด้วย space`, and `3 print(a,b,c)`. To the right of the editor is a 'Console' window showing the execution output: `abc : 10 20 30` followed by `10 20 30` on the next line. A cursor is visible on the line `10 20 30` in the console.

```
main.py x
1 a,b,c = input("abc : ").split()
2 # อ่าน string ต้องป้อน 3 ค่าคั่นด้วย space
3 print(a,b,c)
4
```

Console Shell

```
abc : 10 20 30
10 20 30
> 
```

- การทำงานของคำสั่งข้างต้น คือ Input จะรับมาเป็น string ยาวๆ จากนั้นคำสั่ง `split()` จะทำหน้าที่แยกข้อมูลเป็นส่วนๆ (ข้อมูลต้องคั่นด้วยช่องว่าง) และจึงกำหนดค่าให้กับ `a`, `b`, `c` ตามลำดับ

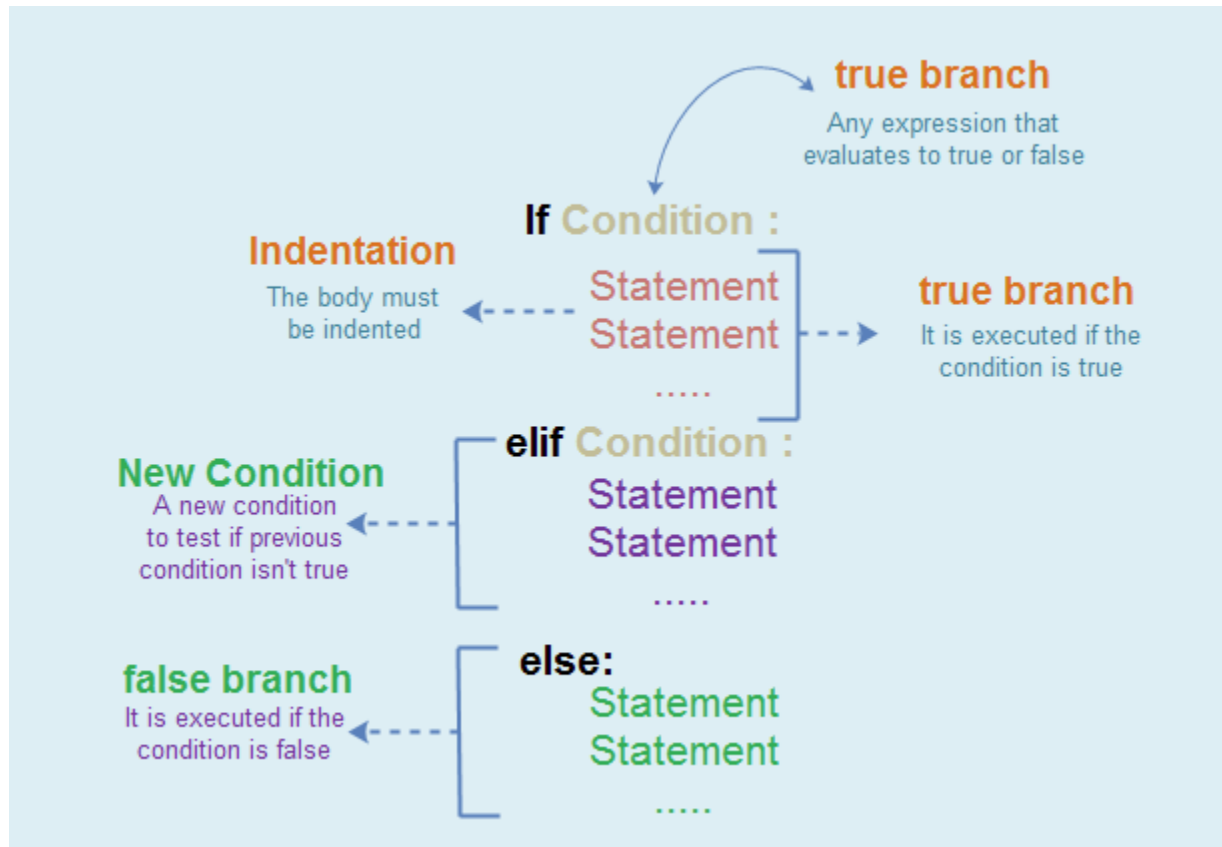
# From C to Python

- ในการรับ Input ครึ่งละหลายค่า มีวิธีการเพิ่มอีก 1 วิธี วิธีนี้เรียกว่า  
List Comprehension จะอธิบายรายละเอียดภายหลัง
- ให้จำรูปแบบการใช้งานไปก่อน ดังนี้  

```
p1,p2,p3,n1,n2 = [int(e) for e in input("Input : ").split()]
```
- จากตัวอย่างเป็นการ Input ข้อมูลจำนวน 5 ตัว โดยป้อนในบรรทัดเดียว และ คั่นด้วยช่องว่าง

# From C to Python

- If else ในภาษา Python มีรูปแบบดังนี้



# From C to Python

- Loop ในภาษา Python จะมีแค่ While กับ For (ไม่มี Do While Loop)
- While Loop มีรูปแบบดังนี้

```
while test_expression:  
    Body of while
```

- โดยสามารถเขียนลดรูป กรณีที่คำสั่งสั้นๆ ได้

main.py ×

```
1 # Python program to illustrate  
2 # Single statement while block  
3 count = 0  
4 ▼ while (count < 5): count += 1; print("Hello CE")  
5  
6
```



Console

Shell

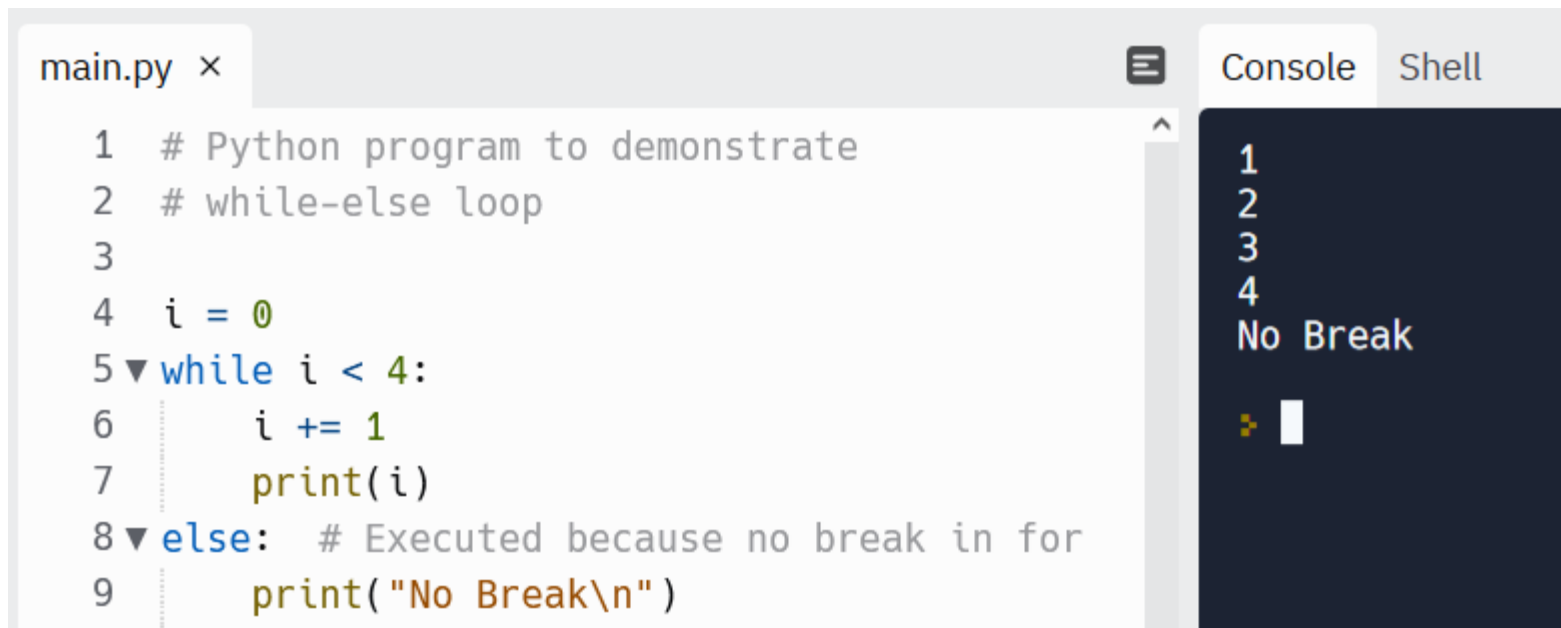
```
Hello CE  
Hello CE  
Hello CE  
Hello CE  
Hello CE
```

```
>
```



# From C to Python

- While Loop ของ Python จะมีความพิเศษที่ไม่มีในภาษาอื่น คือ สามารถใช้ Else ได้
- โดย Else Block จะทำงานหากไม่มีการ break เกิดขึ้นใน While Loop



The screenshot shows a code editor with a file named 'main.py'. The code is a Python program demonstrating a while-else loop. The code is as follows:

```
1 # Python program to demonstrate
2 # while-else loop
3
4 i = 0
5 while i < 4:
6     i += 1
7     print(i)
8 else: # Executed because no break in for
9     print("No Break\n")
```

The output is shown in the 'Console' tab on the right. It displays the numbers 1, 2, 3, and 4, followed by the text 'No Break' on a new line. A cursor is visible at the end of the output line.

# From C to Python

- ตัวอย่าง While Loop ที่มี โดยใช้ break เป็นตัวหยุด loop ( โดยมี continue เพื่อกลับไปเริ่มต้น Loop)

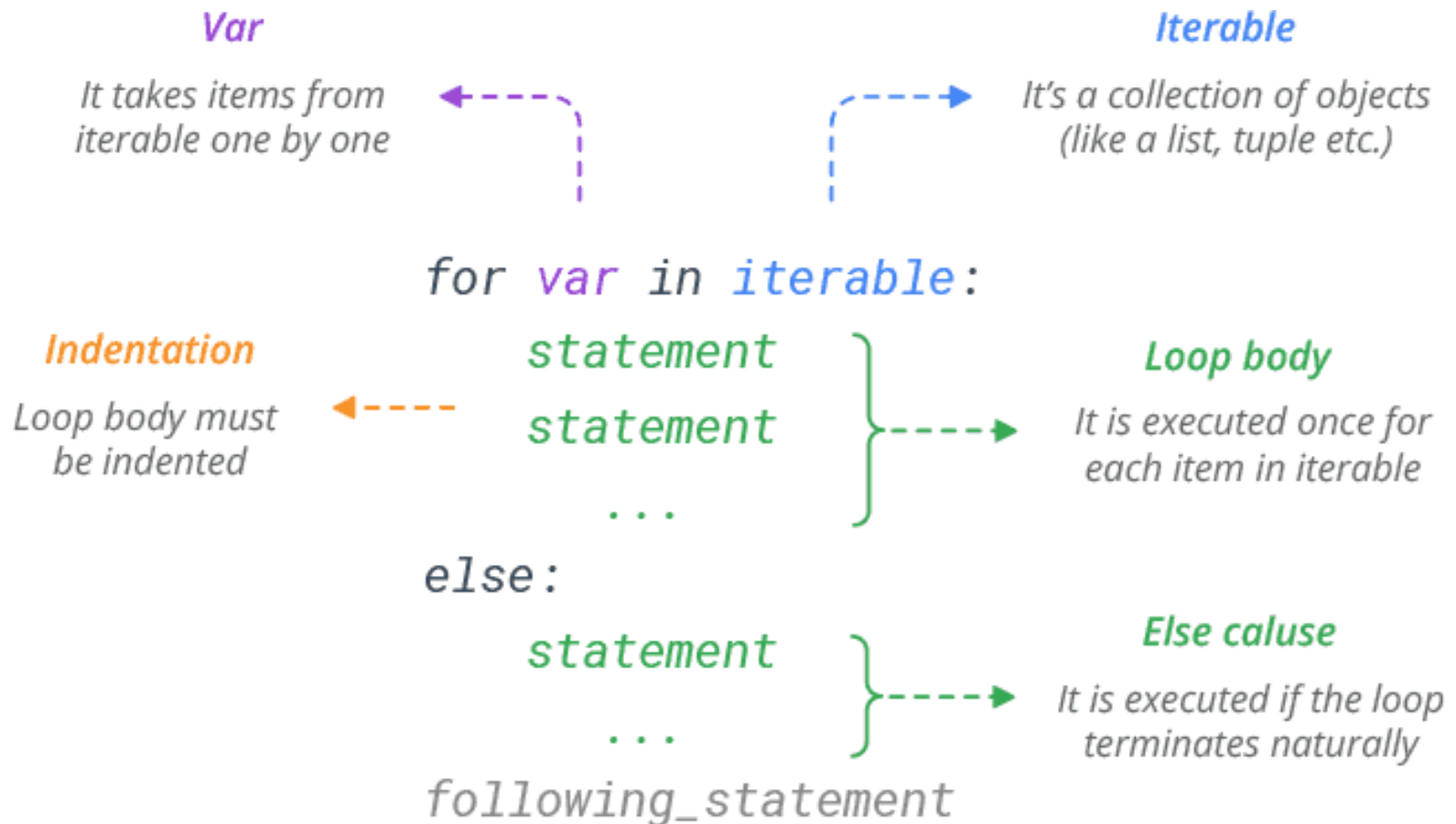
```
main.py x
1 # list of fruits
2 my_list =
  ["papaya", "banana", "pineapple", "mango", "grapes"]
3
4 size = len(my_list) #length/size of the list
5 i=0
6
7 # iterating through the fruit list
8 while i<size:
9     if my_list[i] == 'mango':
10         print("mango found!")
11         break
12     i+=1
13 else:
14     print("mango not found!")
```

Console Shell

mango found!

# From C to Python

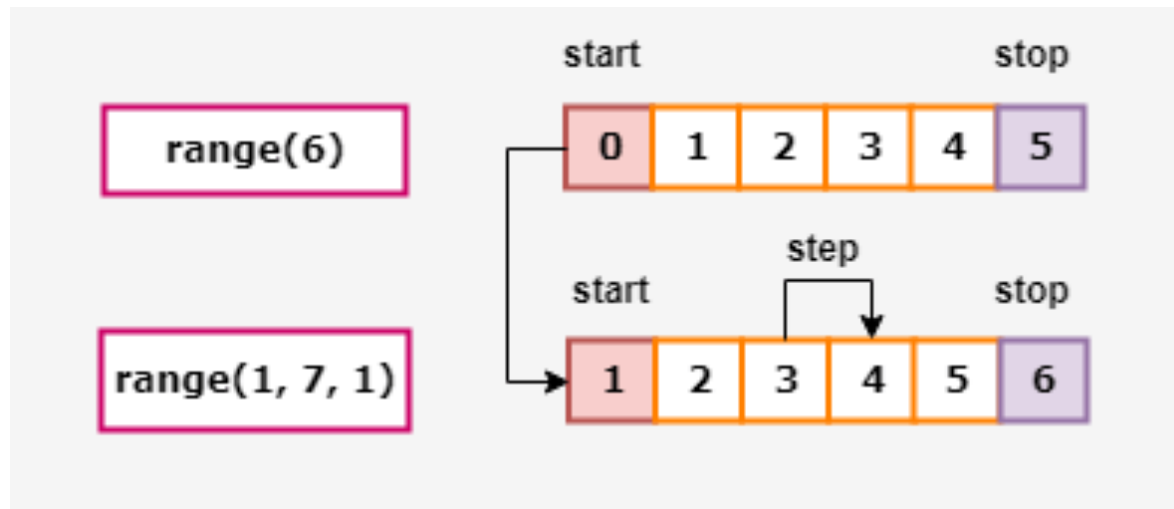
- Loop For ของ Python มีความแตกต่างจาก C โดยมีโครงสร้างดังนี้



# From C to Python

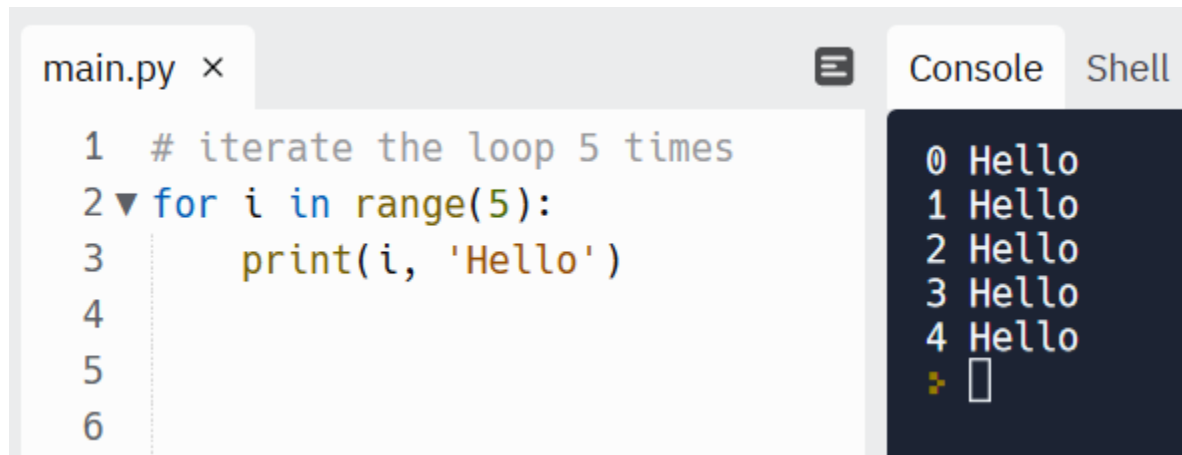
- Iterable หมายถึง Object ที่สามารถส่งค่าออกมาได้เป็นลำดับ
- range เป็นฟังก์ชันที่มีคุณสมบัติเป็น Iterator โดยสามารถคืนค่าเป็นลำดับเลข โดยมีรูปแบบการใช้งานดังนี้

```
range(start, stop, step)
```



# From C to Python

- การใช้ range ร่วมกับ For



```
main.py x
1 # iterate the loop 5 times
2 ▼ for i in range(5):
3     print(i, 'Hello')
4
5
6
```

Console

```
0 Hello
1 Hello
2 Hello
3 Hello
4 Hello
❖
```

- String ก็เป็น Iterator ตัวหนึ่ง ดังนั้นสามารถใช้ For กับ String ได้

# From C to Python

```
#include <stdio.h>

int main(void) {
    int n;
    n = 10;

    for (int row = 0; row < n; row++) {
        for (int col = 0; col < n; col++) {
            if (row + col < n) {
                printf(" ");
            } else {
                printf("#");
            }
        }
        printf("#\n");
    }
}
```

```
# Draw Triangle
n = 10

for row in range(n):
    for col in range(n):
        if row+col < n:
            print(' ', end='')
        else:
            print("#", end='')
    print('#')
```



***For your  
attention***