



สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)
Technology Promotion Association (Thailand-Japan)



สถาบันเทคโนโลยีไทย-ญี่ปุ่น
Thai-Nichi Institute of Technology
泰日工業大学

พื้นฐานการเขียนโปรแกรม python และการ ประยุกต์ใช้งานทางด้าน AI (การเรียนรู้ของเครื่องและการทำนายผลลัพธ์)

19-20 สิงหาคม 2566

แนะนำทีมงาน

2

- รศ. ดร.วรากร ศรีเชวงทร์พย়
- ผศ. ดร.กันติชา กิตติพิรชล
- อ.ชาตรี ทองวรณ

หัวข้อการอบรม

3

- แนะนำภาษา Python
- หลักภาษาและไวยกรณ์ทั่วไปของภาษา Python
- การเขียนคำสั่งเกี่ยวกับเงื่อนไขทางเลือก
- การเขียนคำสั่งวนลูป
- การใช้งาน String และ List
- การสร้างและใช้งานฟังก์ชัน (Function)
- การประยุกต์ใช้ในงานทางด้าน AI (การเรียนรู้ของเครื่องและทำนายผลลัพธ์)



สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)
Technology Promotion Association (Thailand-Japan)



สถาบันเทคโนโลยีไทย-ญี่ปุ่น
Thai-Nichi Institute of Technology
泰日工業大学

แนะนำภาษา PYTHON

หลักภาษาและไวยกรณ์ทั่วไป

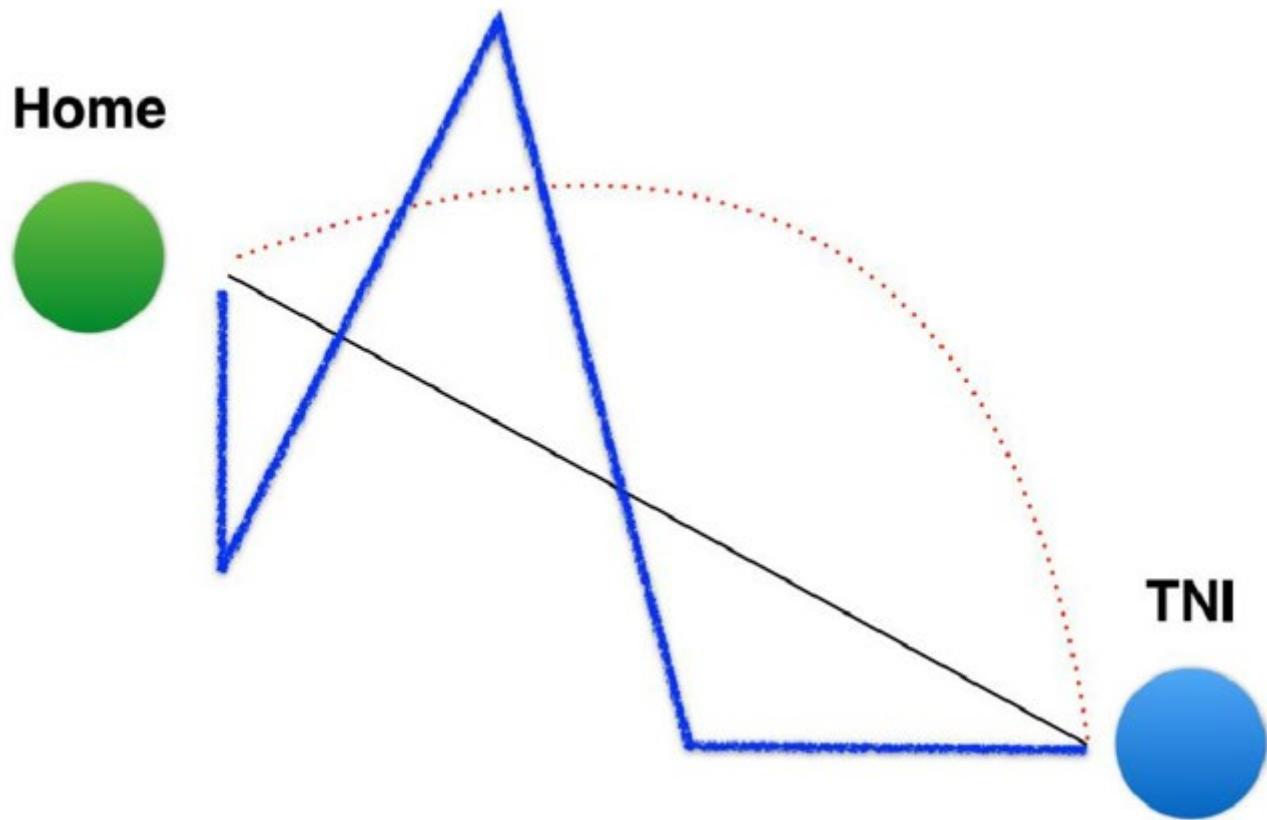
Program

5

**“A computer program is a
set of instructions...”**

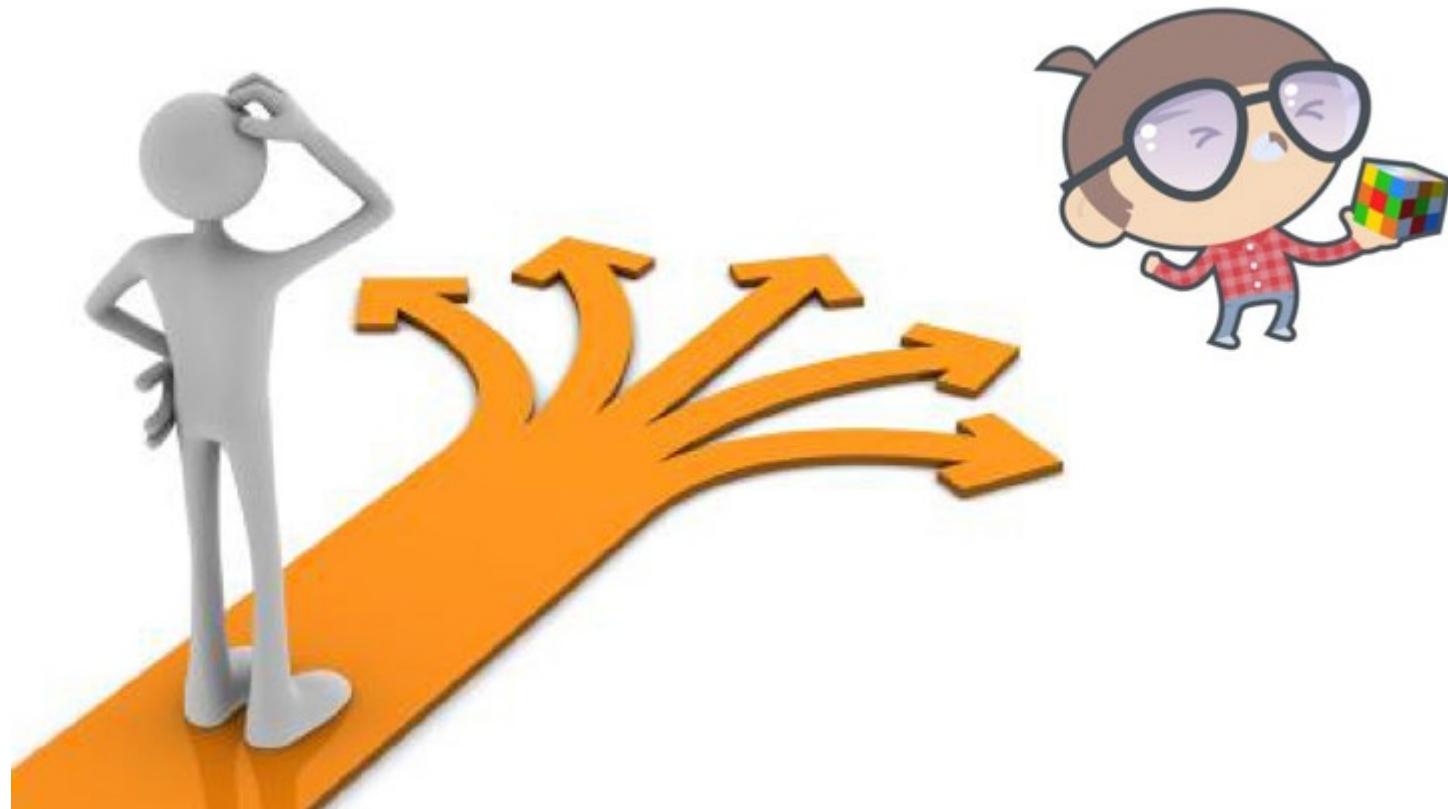
Program

6



Program

7



Program

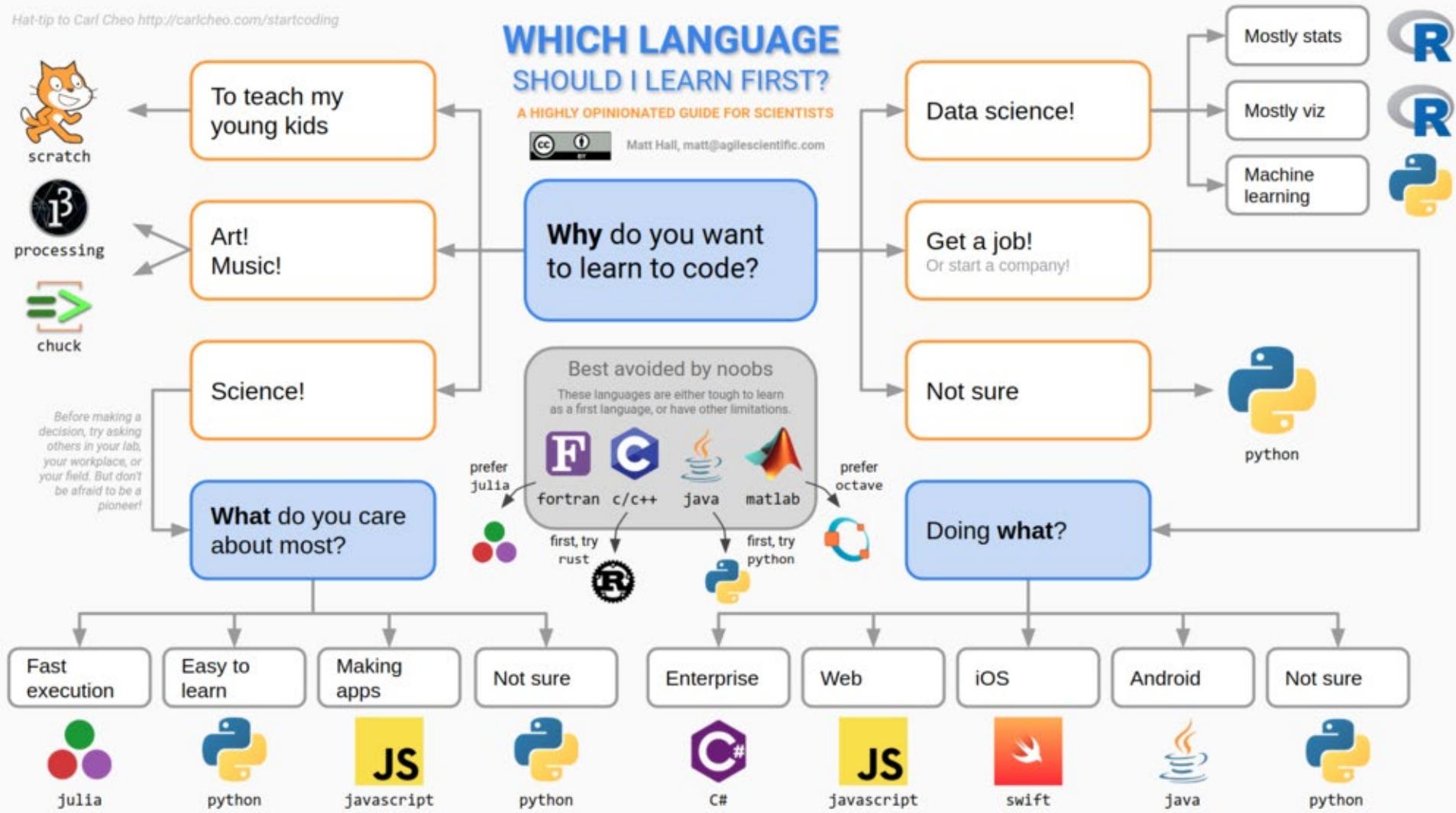
8



Program

9

Hat-tip to Carl Cheo <http://carlcheo.com/startcoding>

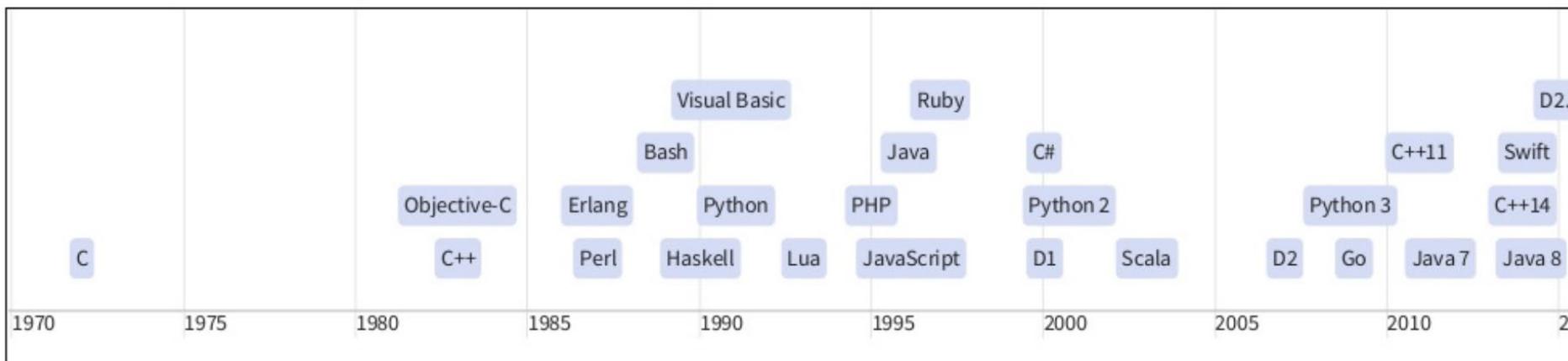


History of Popular Programming Languages: A timeline



Program

11



PYTHON 2



Legacy

It is still entrenched in the software at certain companies



Library

Many older libraries built for Python 2 are not forwards-compatible

0100
0001

ASCII

Strings are stored as ASCII by default



PYTHON 3

Future

It will take over Python 2 by 2020



Library

Many of today's developers are creating libraries strictly for use with Python 3



Unicode

0000
0000
0100
0001

Text strings are Unicode by default

PYTHON 2

$$5/2=2$$

It rounds your calculation down
to the nearest whole number

`print "hello"`

Python 2 print statement

PYTHON 3

$$5/2=2.5$$



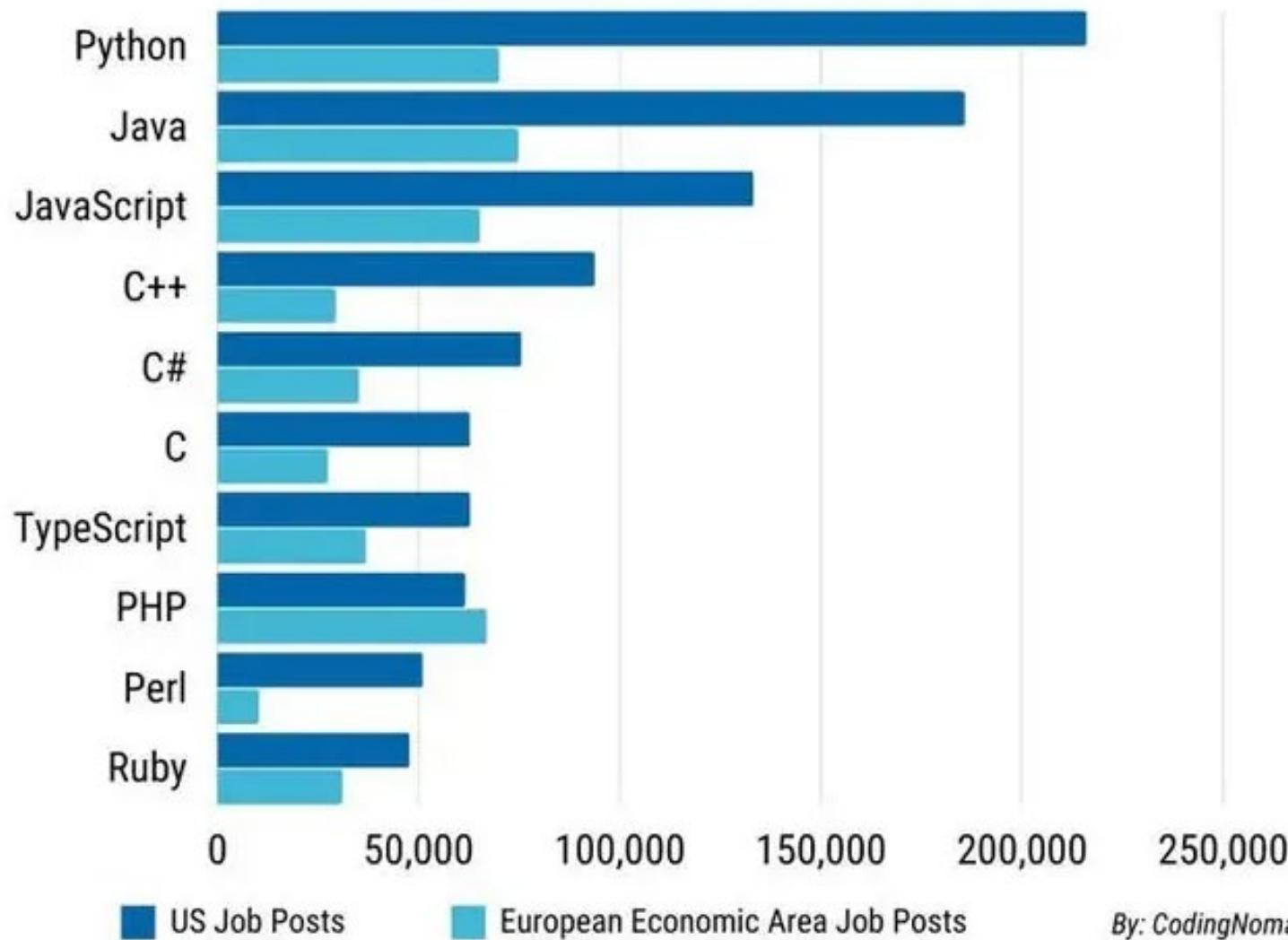
The expression `5 / 2` will return
the expected result

`print ("hello")`

The print statement has been
replaced with a `print()` function

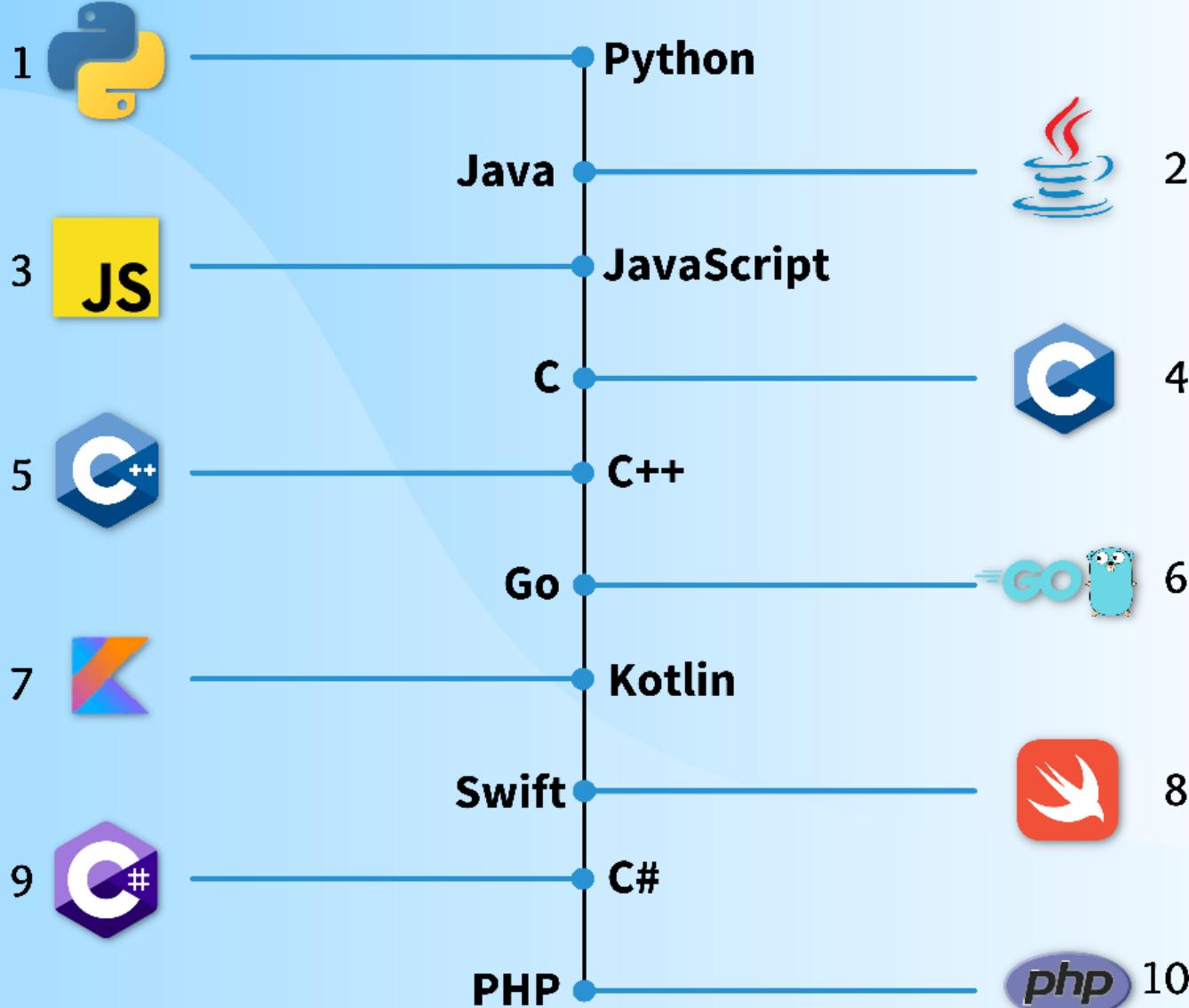
Most in-demand programming languages of 2022

Based on LinkedIn job postings in the USA & Europe



By: CodingNomads

10 MOST POPULAR CODING LANGUAGES OF 2021



Program

16

1. Python

Released in: **1991**

Career opportunities: **Very high**

Average base salary: **\$108,602** per year (as of June 2021)

Current number of jobs: **40,026** (as of June 2021)

Source: *Indeed.com*

As one of the most popular coding languages globally, Python finds invaluable use in solving problems and gaining insights. The language has recorded a 27% year-over-year user growth, mainly fueled by data analysts and data scientists.

Its application in AI analysis, deep learning, and data analysis is indispensable. Python is definitely among the top 3 popular programming languages of 2021.

Why Should You Learn Python?

17

- Python is a general-purpose, high-level, interpreted language known for its ease of use, extensive libraries and tools, and code readability.
- Python can handle complex algorithms and is excellent for data automation, making it a natural fit for machine learning.

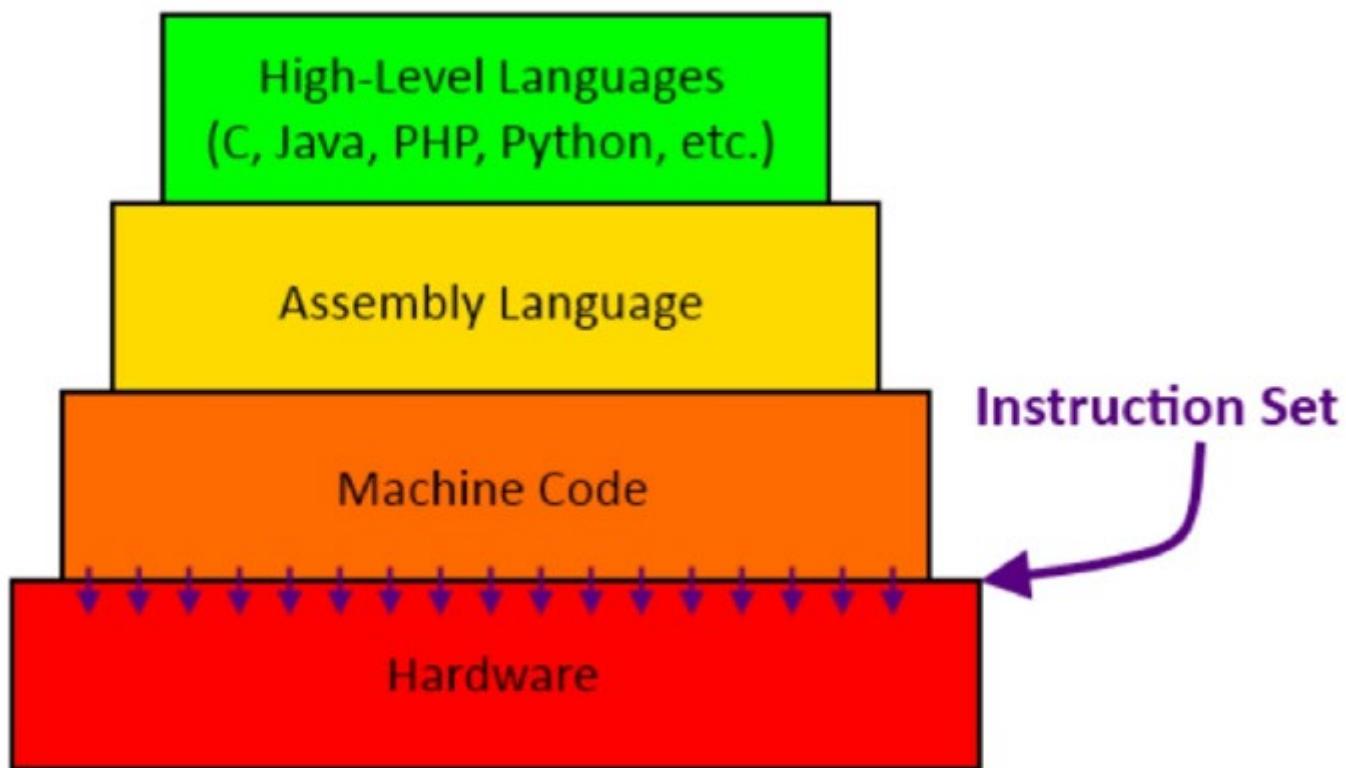
Why Should You Learn Python?

18

- Being open-source, it has support from a large community of Python users committed to making it better and more efficient.
- Python has several mobile and web development frameworks like Django, TurboGears, Bottle, Pyramid, and Flask, as well as scientific application frameworks like TensorFlow, Keras, and SciPy.
- Software engineers use Python to develop GUI-based applications, handle AI computations, and information science.

Python

19

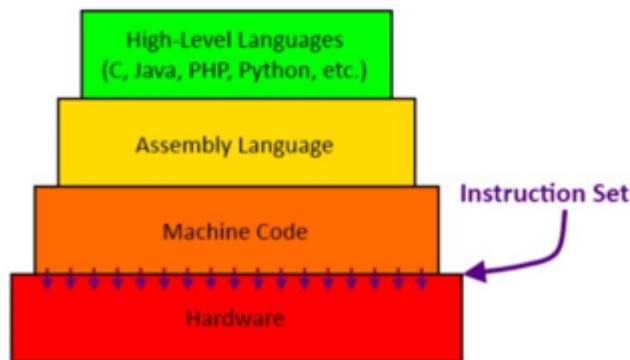


Python

20

Python

```
print ("HELLO WORLD")
```



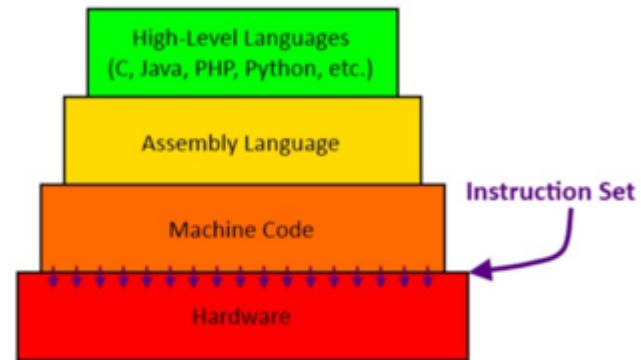
Assembly

```
.or $300  
main    ldy #$00  
.1      lda str,y  
        beq .2  
        jsr $fded  
        iny  
        bne .1  
        rts  
.2      str   .as "HELLO WORLD"  
        .hs 0D00
```

Python

21

Machine Code

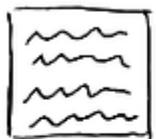


Compiled vs. Interpreted

22

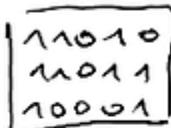
Source code:

hello.c



COMPILER →

Machine code:



Program (also
called binary,
executable ...)

result

run the
program



Source code:

hello.py



INTERPRETER → result



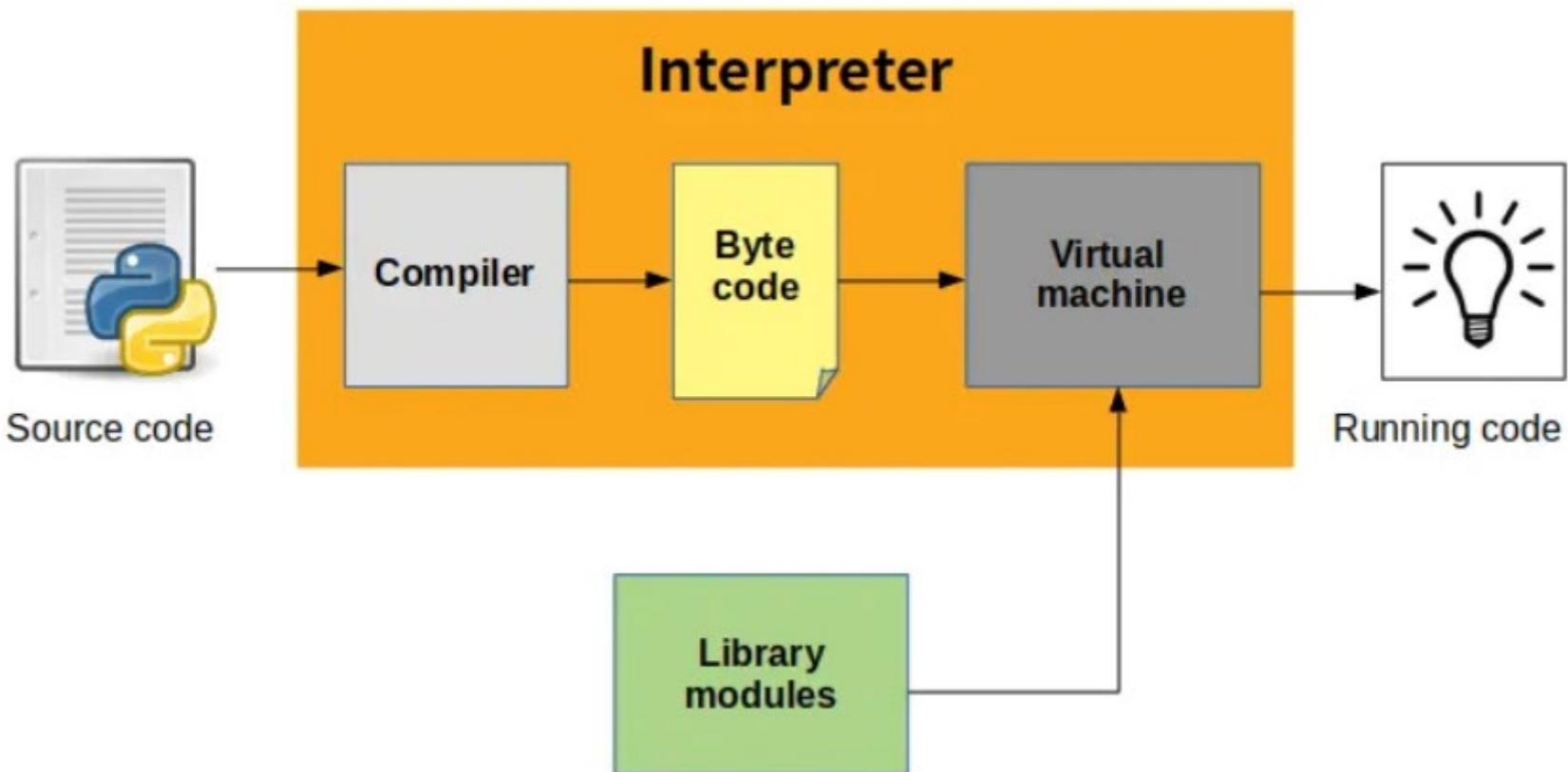
Compiled vs. Interpreted

23

Compiled		Interpreted	
PROS	CONS	PROS	CONS
ready to run	not cross platform	cross-platform	interpreter required
often faster	inflexible	simpler to test	often slower
source code is private	extra step	easier to debug	source code is public

Program

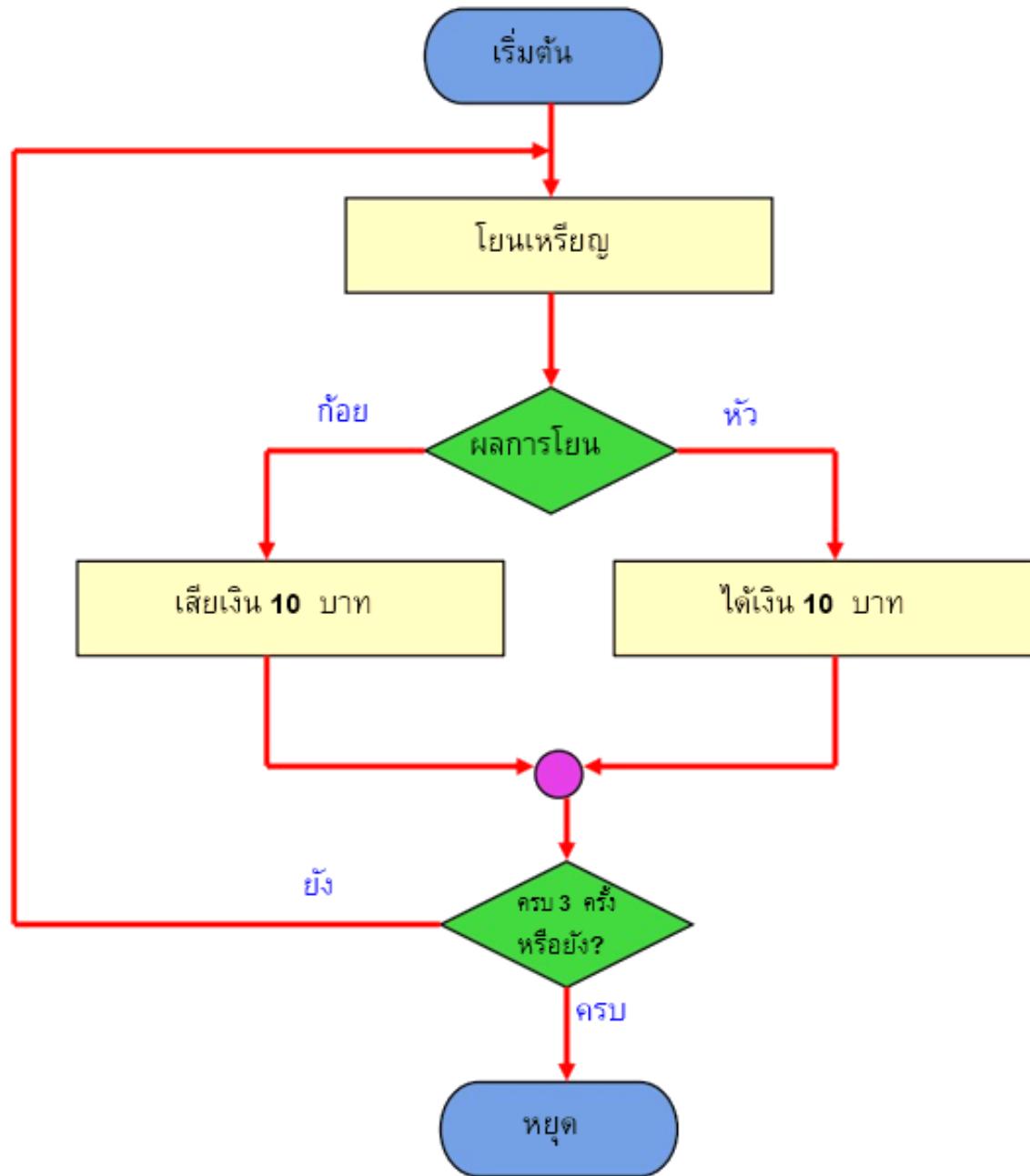
24

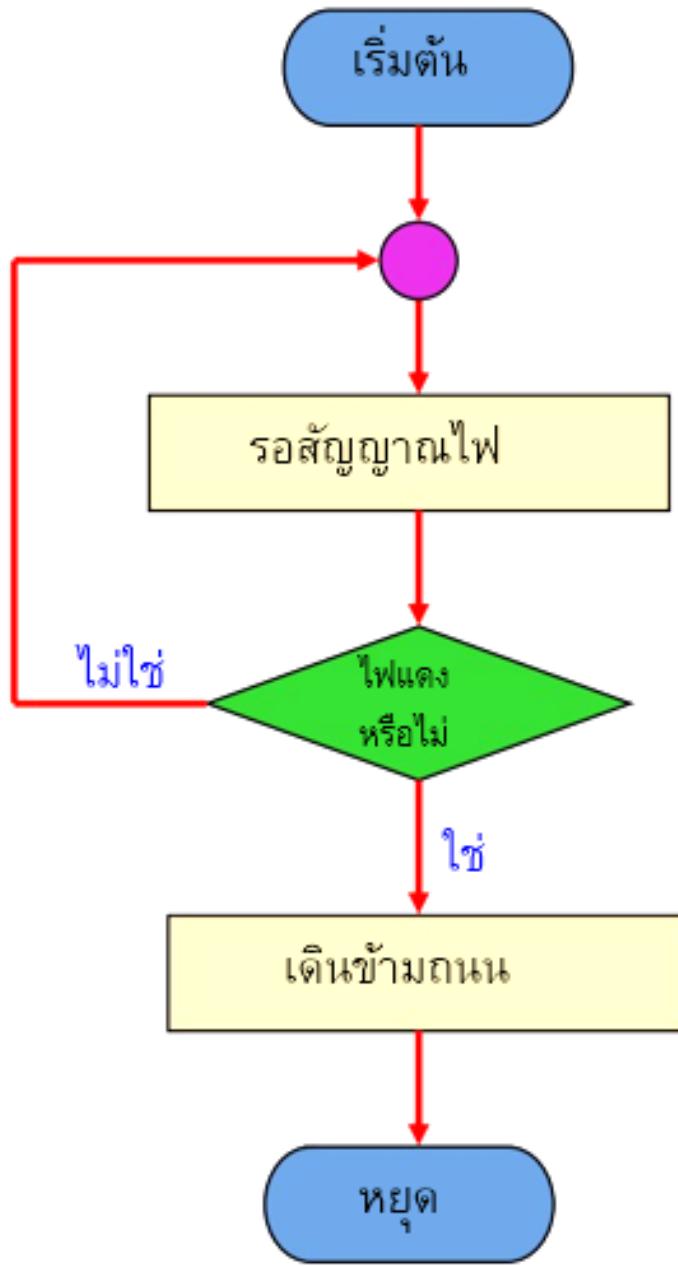


Flowchart

25

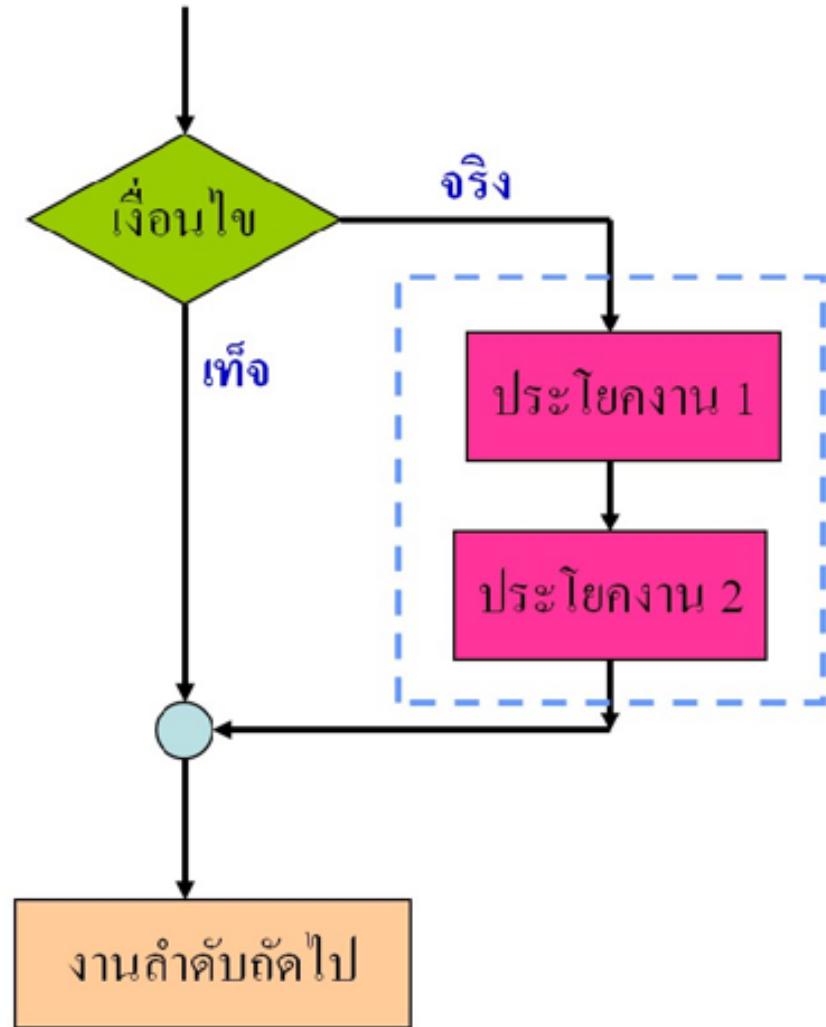
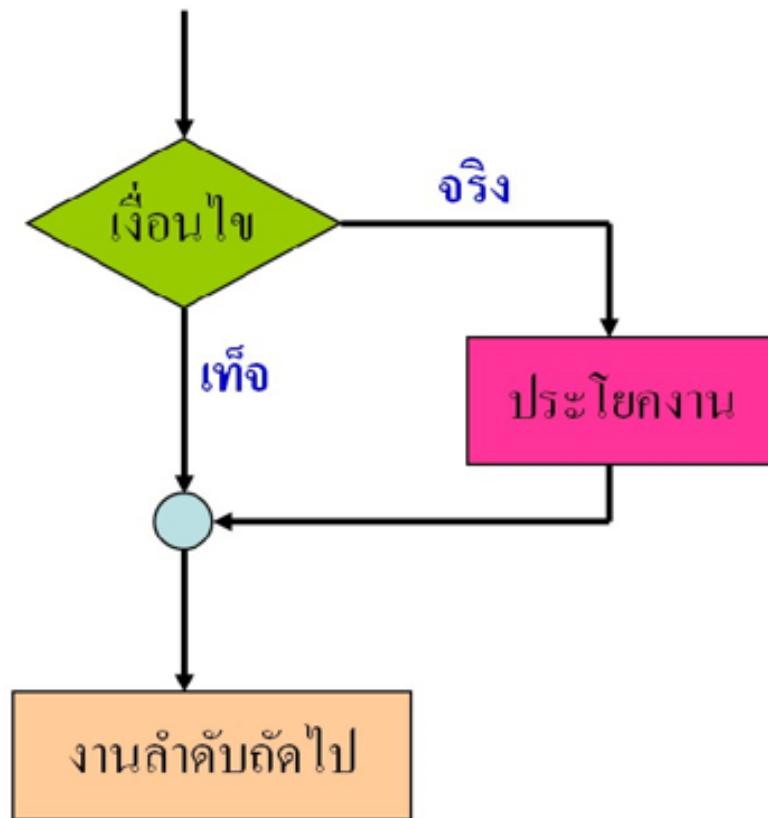
- เครื่องมือที่ช่วยในการเขียนโปรแกรม
- เครื่องหมายแสดงลำดับขั้นตอน
- รูปภาพหรือสัญลักษณ์
- เข้าใจลำดับขั้นตอนการเขียนโปรแกรม
- ง่ายต่อการตรวจสอบความถูกต้อง





Flowchart

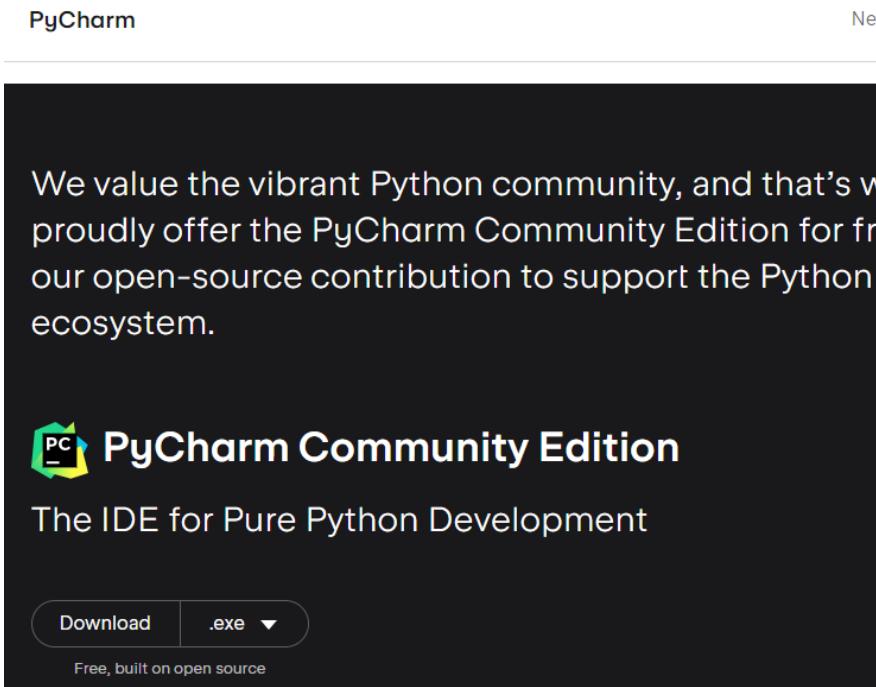
28



Python IDE

29

- Reference : <http://www.jetbrains.com/pycharm/download/>
<https://www.python.org>



Variables (Types)

30

10

Integer

5.5

Floating point

Hello

String

TRUE & FALSE

Boolean

Variables (Types)

31



l0101.py ×

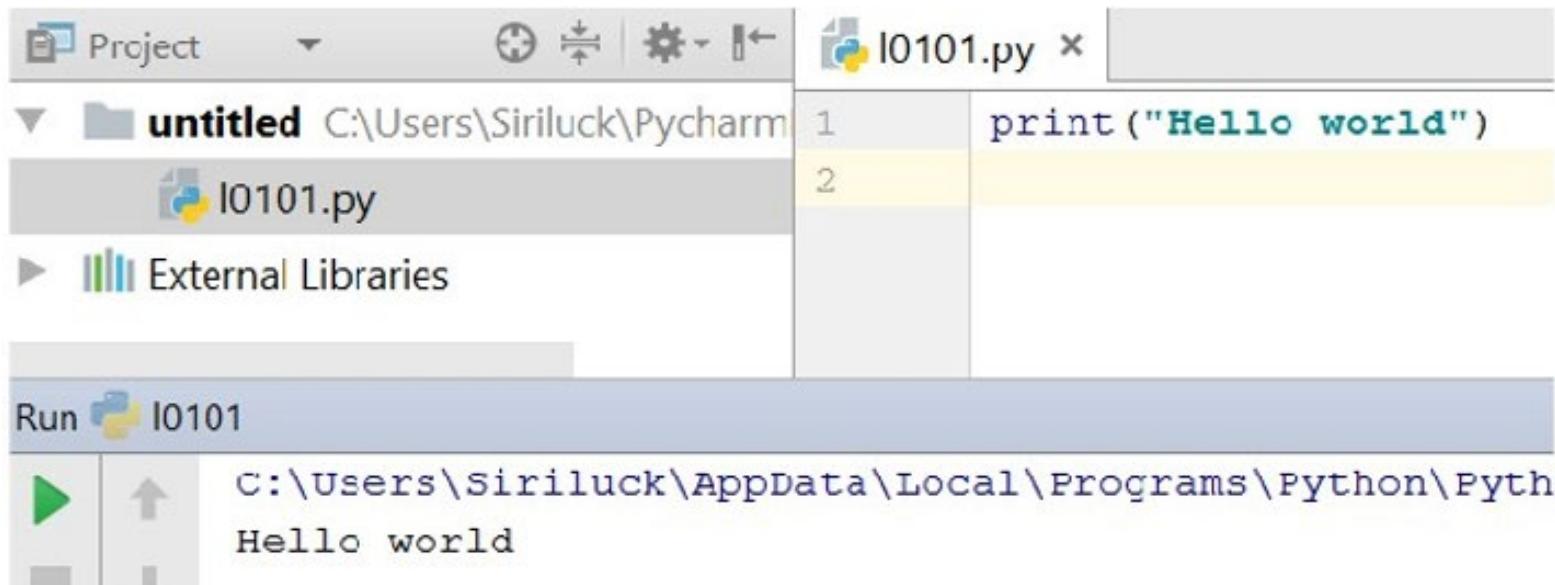
```
1 myInt = 10           #integer
2 myFloat = 5.5        #Floating point
3 myString = "Python"  #String
```

print

32

```
print ("Hello word")
```

```
print (name_data)
```



The screenshot shows the PyCharm IDE interface. The top navigation bar includes 'Project', a search bar, and tool icons. Below the navigation bar, the 'untitled' project structure is visible, containing a file named 'l0101.py'. The code editor shows two lines of Python code:

```
1 print("Hello world")
2
```

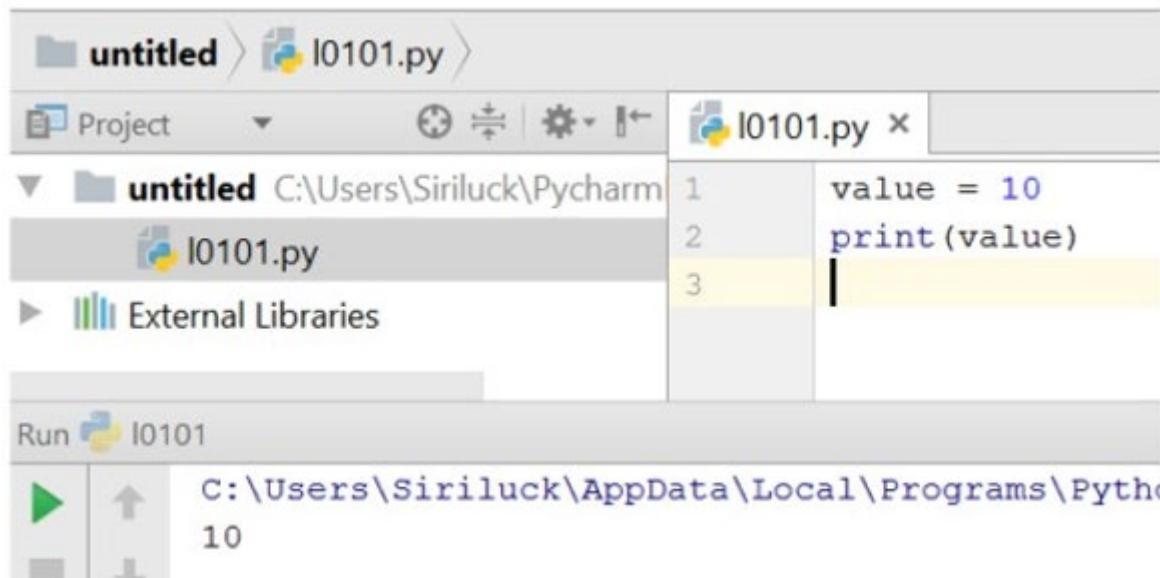
The first line is highlighted with a yellow background. The bottom part of the interface shows the 'Run' toolbar with a green play button icon and the text 'Run l0101'. To the right of the toolbar, the output window displays the text 'Hello world'.

print

33

```
print ("Hello word")
```

```
print (name_data)
```



The screenshot shows the PyCharm IDE interface. The project navigation bar at the top shows 'untitled' and 'l0101.py'. The left sidebar shows a project structure with 'untitled' containing 'l0101.py' and 'External Libraries'. The main editor window displays the following code:

```
1 value = 10
2 print(value)
3
```

The output window at the bottom shows the result of running the program: '10'.

```
print('{:.3f}'.format(value))
```

```
print('value is {:.2f} {}'.format(value))
```

print

34

print (a, b)

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top indicates the current file is `l0101.py`. The left sidebar displays the project structure under the `untitled` folder, showing the file `l0101.py` is selected. The main editor window contains the following Python code:

```
myInt = 10          #integer
myFloat = 5.5       #Floating point
myString = "Python" #String
print(myInt, myFloat, myString)
```

The code uses triple quotes for the string assignment. The run configuration at the bottom shows the command being used is `C:\Users\Siriluck\AppData\Local\Programs\Python\Python36-32\python.` The output pane shows the results of the print statement: `10 5.5 Python`.

Variables (operators)

35

+	addition
-	subtraction
*	multiplication
/	classic division
%	modulus
//	floor division
**	Exponentiation

Variables (operators)

36

$(50 - 5 * 6) / 4$

$50 - 5 * 6 / 4$

$17 / 3$

$17 // 3$

$17 \% 3$

$5 * 3 + 2$

Operator	Description	highest
$**$	Exponentiation	
$* / \% //$	Multiply, divide, modulo and floor division	
$+ -$	Addition and subtraction	



lowest

Variables (Strings)

37

“John”

“hello”

‘123’

“8/5”

“.....”	double quote
‘.....’	single quote

```
print ("Hello word")
```

```
print ("Hello TNI")
```

or

```
print ("Hello word\nHello TNI")
```

The screenshot shows the PyCharm IDE interface. The left sidebar displays a project named "untitled" with a single file "l0101.py". The code editor window shows the following content:

```
1 print ("Hello word")
2 print ("Hello TNI")
3
4 print ("Hello word\nHello TNI")
```

The run tool window at the bottom shows the output of the script:

```
C:\Users\Siriluck\AppData\Local\Programs\Python\Python36-32\pyt
Hello word
Hello TNI
Hello word
Hello TNI
```

```
word = "Py"  
print(word + 'thon')
```

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top indicates the current file is `l0101.py`. The left sidebar displays the project structure under the `untitled` folder, which contains the file `l0101.py`. The main editor window shows the following Python code:

```
word = "Py"  
print(word + 'thon')
```

The code is numbered from 1 to 6. The run configuration bar at the bottom shows the file name `l0101` and the Python path `C:\Users\Siriluck\AppData\Local\Programs\Python\Python`.

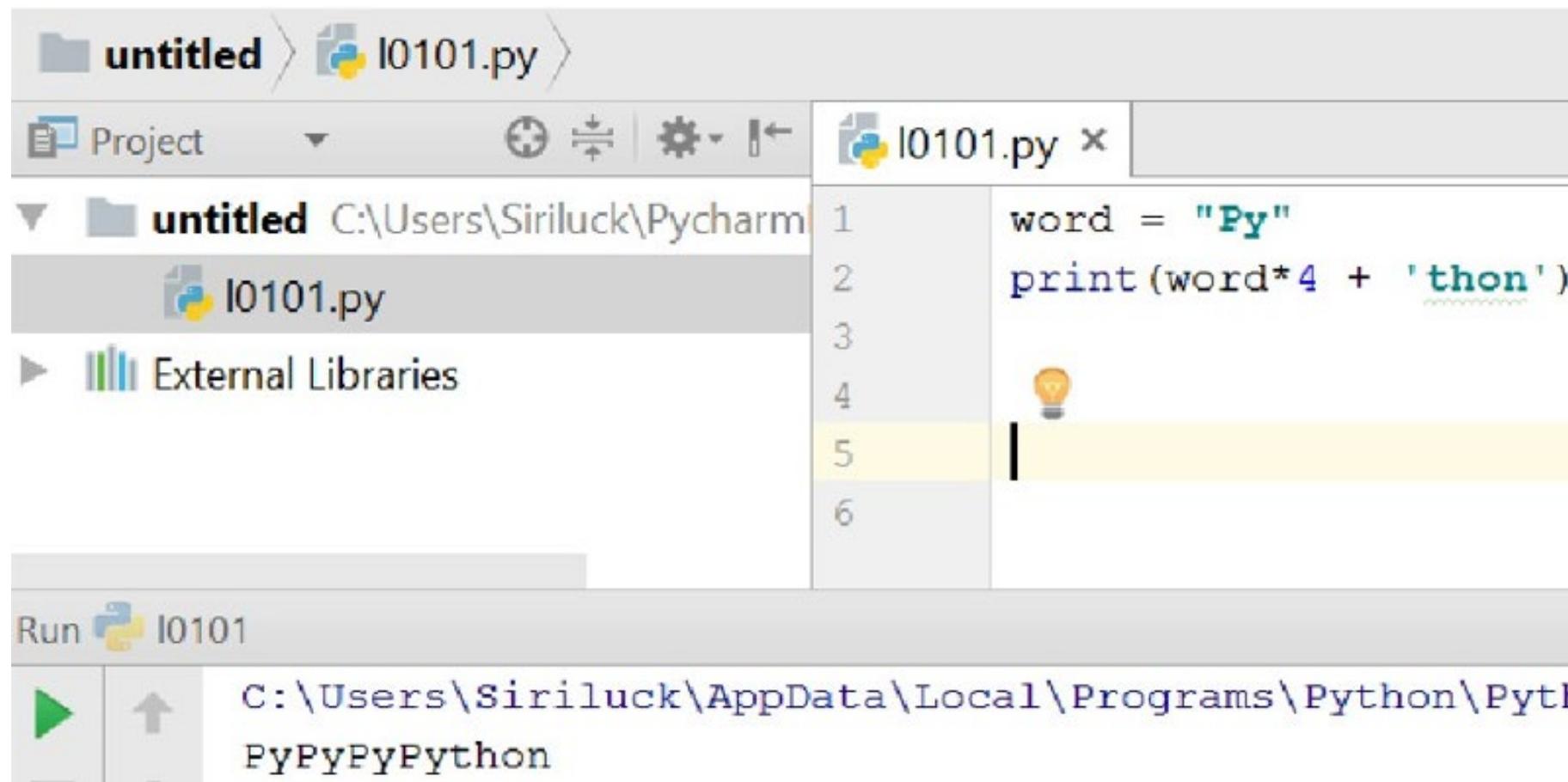
```
word = "Py"  
print(word*4)
```

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top shows the current file is `l0101.py`. The left sidebar displays the project structure under an `untitled` folder, with `l0101.py` selected. The main editor window contains the following Python code:

```
1 word = "Py"  
2 print(word*4)  
3  
4  
5  
6
```

A yellow lightbulb icon is positioned next to the fifth line of code. The bottom run toolbar shows the command `Run l0101`, and the terminal window below it displays the output: `PyPyPyPy`.

`word = "Py"`
`(word*4 + 'thon')`



The screenshot shows the PyCharm IDE interface. The project navigation bar at the top indicates the current file is `l0101.py`. The left sidebar shows a single file named `l0101.py` under the `untitled` folder. The main editor window displays the following Python code:

```
1 word = "Py"
2 print(word*4 + 'thon')
```

The code consists of two lines: line 1 defines a variable `word` with the value `"Py"`, and line 2 prints the result of concatenating `word` four times with the string `'thon'`. A yellow lightbulb icon is positioned above line 4, indicating a potential code analysis or suggestion. The bottom navigation bar shows the run configuration `Run l0101` and the Python interpreter path `C:\Users\siriluck\AppData\Local\Programs\Python\Python`.

```
print("Enter the value : ")
```

```
n=input()
```

or

```
n=input("Enter the value : ")
```

The screenshot shows the PyCharm IDE interface. On the left is the project tree with a folder named 'untitled' containing a file 'l0101.py'. The code editor window shows the following Python code:

```
1 print("Enter the value : ")
2 n=input()
3
```

The third line is highlighted with a yellow background. Below the code editor is the run terminal window, which displays the command 'C:\Users\Siriluck\AppData\Local\Programs\Python\Python36-' followed by the prompt 'Enter the value : '.

The screenshot shows the PyCharm IDE interface with the code editor window. The code shown is:

```
1 n=input("Enter the value : ")
2
```

The second line is highlighted with a yellow background, and the cursor is positioned at the end of the line.

The screenshot shows the PyCharm Run dialog titled 'Run l0101'. It displays the output of the program, which includes the command 'C:\Users\Siriluck\' followed by the prompt 'Enter the value : ' and the user input '12'.

```
print("Enter the value : ")
```

```
n=input()
```

or

```
n=input("Enter the value : ")
```

```
if n = 2
```

```
A = n * 2      // 22      print (A)
```

```
A = int(n) * 2 // 4      print(str(A))
```

Question

44

- เขียนโปรแกรมคำนวณ พ.ศ. เกิดจากการรับอายุ
(ตัวเลข)

```
How old are you : 31  
So, you're born in 2535
```

- โปรแกรมคำนวณพื้นที่สี่เหลี่ยม จากการรับค่าความกว้าง

```
Enter the length of side : 12  
Area of square : 144
```



สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)
Technology Promotion Association (Thailand-Japan)



สถาบันเทคโนโลยีไทย-ญี่ปุ่น
Thai-Nichi Institute of Technology
泰日工業大学

conditionals and loops

Comparison operator

46

operator

Description

Example

=

Checks if the value of two operands is equal or not, if yes then condition becomes true.

$(a == b)$ is not true.

!=

Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.

$(a != b)$ is true.

◇

Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.

$(a <> b)$ is true. This is similar to != operator.

>

Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.

$(a > b)$ is not true.

Comparison operator

47

operator	Description	Example
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) is true.

Comparison operator

48

operator

Description

Example

=

Simple assignment operator, Assigns values from right side operands to left side operand

`c = a + b` will assigne value of `a + b` into `c`

`+=`

Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand

`c += a` is equivalent to `c = c + a`

`-=`

Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand

`c -= a` is equivalent to `c = c - a`

`*=`

Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand

`c *= a` is equivalent to `c = c * a`

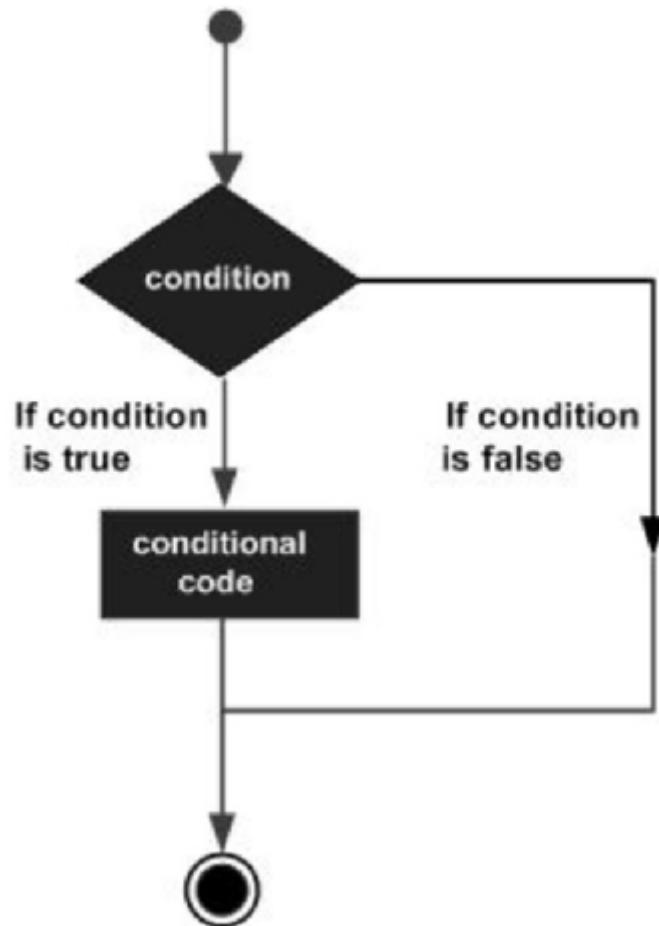
Comparison operator

49

operator	Description	Example
<code>/=</code>	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	<code>c /= a</code> is equivalent to <code>c = c / a</code>
<code>%=</code>	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	<code>c %= a</code> is equivalent to <code>c = c % a</code>
<code>**=</code>	Exponent AND assignment operator, Performs exponential (power) calculation on operators and assign value to the left operand	<code>c **= a</code> is equivalent to <code>c = c ** a</code>
<code>//=</code>	Floor Division and assigns a value, Performs floor division on operators and assign value to the left operand	<code>c //= a</code> is equivalent to <code>c = c // a</code>

Flowchart : if

50



Flowchart : if

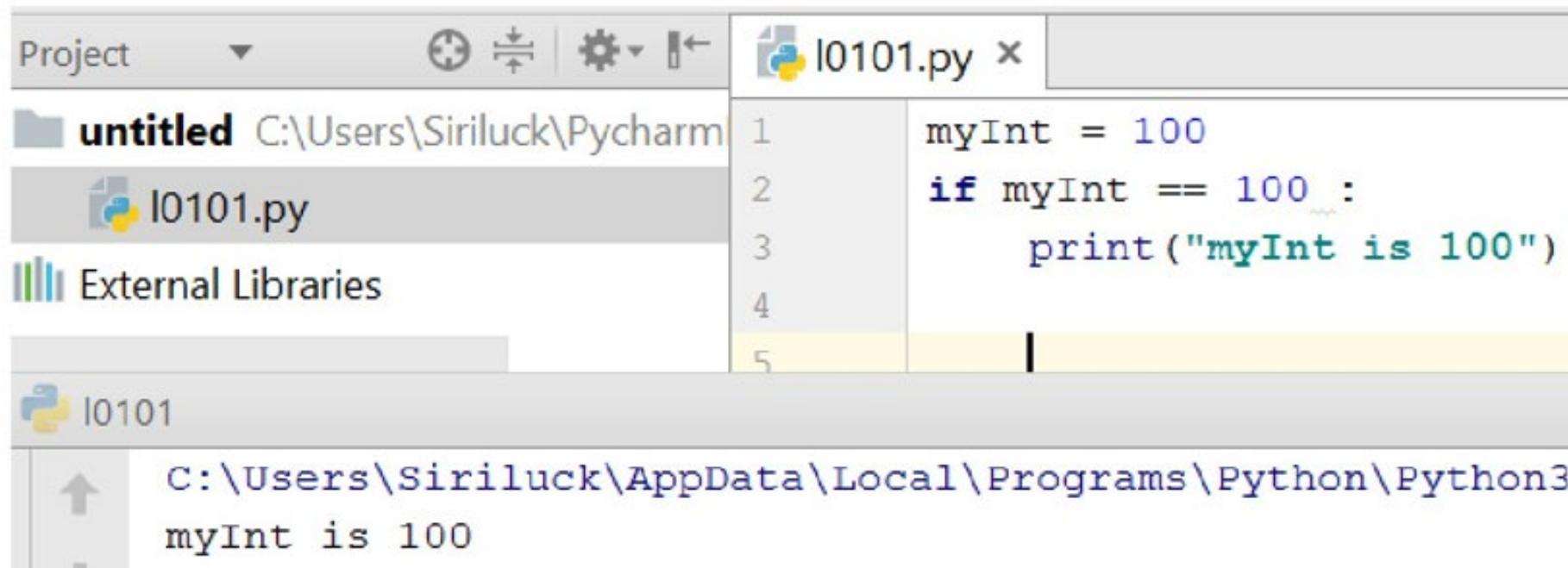
51

```
if expression:  
    #statement goes here  
    #.....  
    #.....
```

Flowchart : if

52

```
myInt = 100  
if myInt == 100 :  
    print("myInt is 100")
```



The screenshot shows the PyCharm IDE interface. The left sidebar displays a project structure with an 'untitled' folder containing 'l0101.py'. The main editor window shows the following Python code:

```
myInt = 100  
if myInt == 100 :  
    print("myInt is 100")
```

The code is run, and the output is shown in the bottom terminal window:

```
C:\Users\Siriluck\AppData\Local\Programs\Python\Python3  
myInt is 100
```

Flowchart : if

53

```
myInt = 100
if myInt > 10 :
    myInt = 0
    print("RESET")
```

The screenshot shows the PyCharm IDE interface. On the left is the Project tool window with an 'untitled' folder containing 'l0101.py'. The main editor window displays the following Python code:

```
myInt = 100
if myInt > 10 :
    myInt = 0
    print("RESET")
```

The 'Run' toolbar at the bottom has a green play button and the text 'Run l0101'. Below the toolbar, the status bar shows the path 'C:\Users\siriluck\AppData\Local\Programs\Python\'. The output window shows the word 'RESET'.

Flowchart : if

54

```
myInt_1 = 100
myInt_2 = 15
if myInt_1 > 10 :
    myInt_1 = 0
    print("Reset myInt_1")

if myInt_2 > 20 :
    myInt_2 = 10
    print("Reset myInt_2")
```

```
untitled > l0101.py >
Project C:\Users\Siriluck\Pycharm
untitled C:\Users\Siriluck\Pycharm
l0101.py
External Libraries
```

```
myInt_1 = 100
myInt_2 = 15
if myInt_1 > 10 :
    myInt_1 = 0
    print("Reset myInt_1")
if myInt_2 > 20 :
    myInt_2 = 10
    print("Reset myInt_2")
```

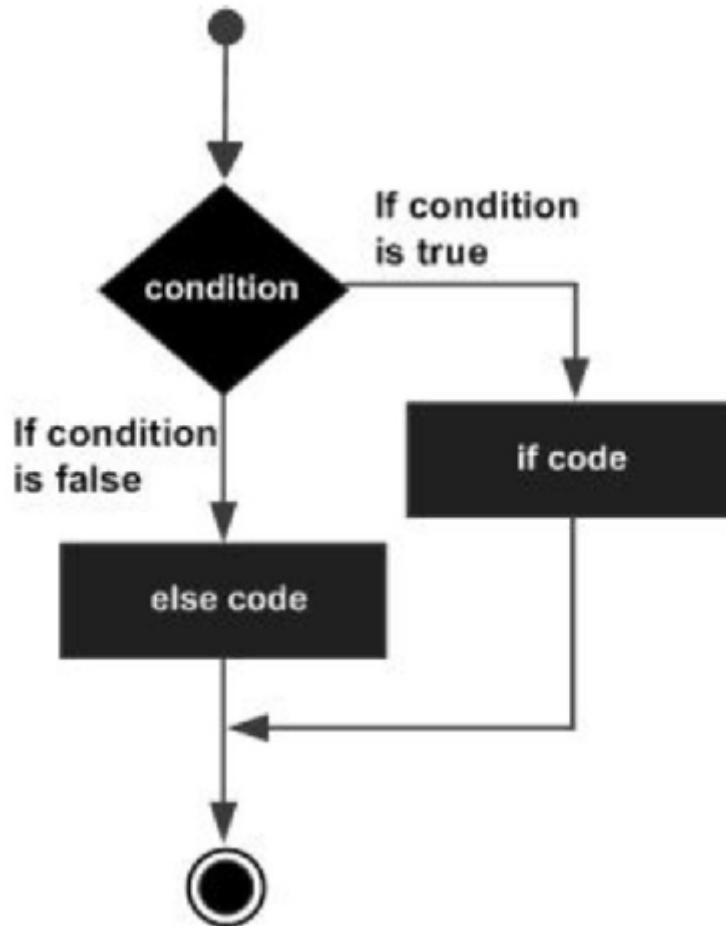
Run l0101

C:\Users\Siriluck\AppData\Local\Programs\Python\Python36-32

Reset myInt_1

Flowchart : else if

55



Flowchart : else if

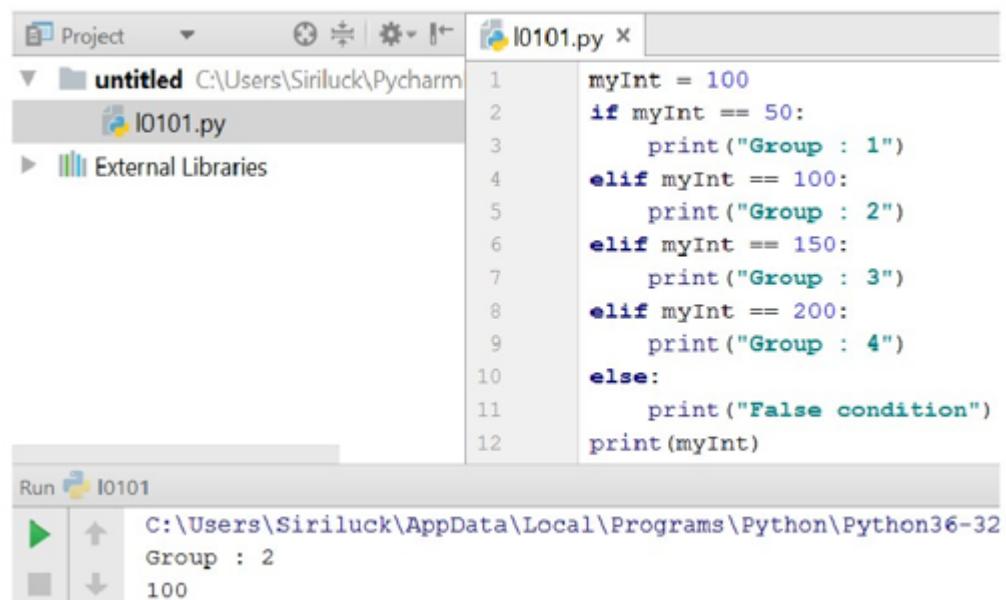
56

```
if expression1:  
    statement(s)  
elif expression2:  
    statement(s)  
elif expression3:  
    statement(s)  
else:  
    statement(s)
```

Flowchart : else if

57

```
myInt = 100
if myInt == 50:
    print("Group : 1")
elif myInt == 100:
    print("Group : 2")
elif myInt == 150:
    print("Group : 3")
elif myInt == 200:
    print("Group : 4")
else:
    print("False condition")
print(myInt)
```

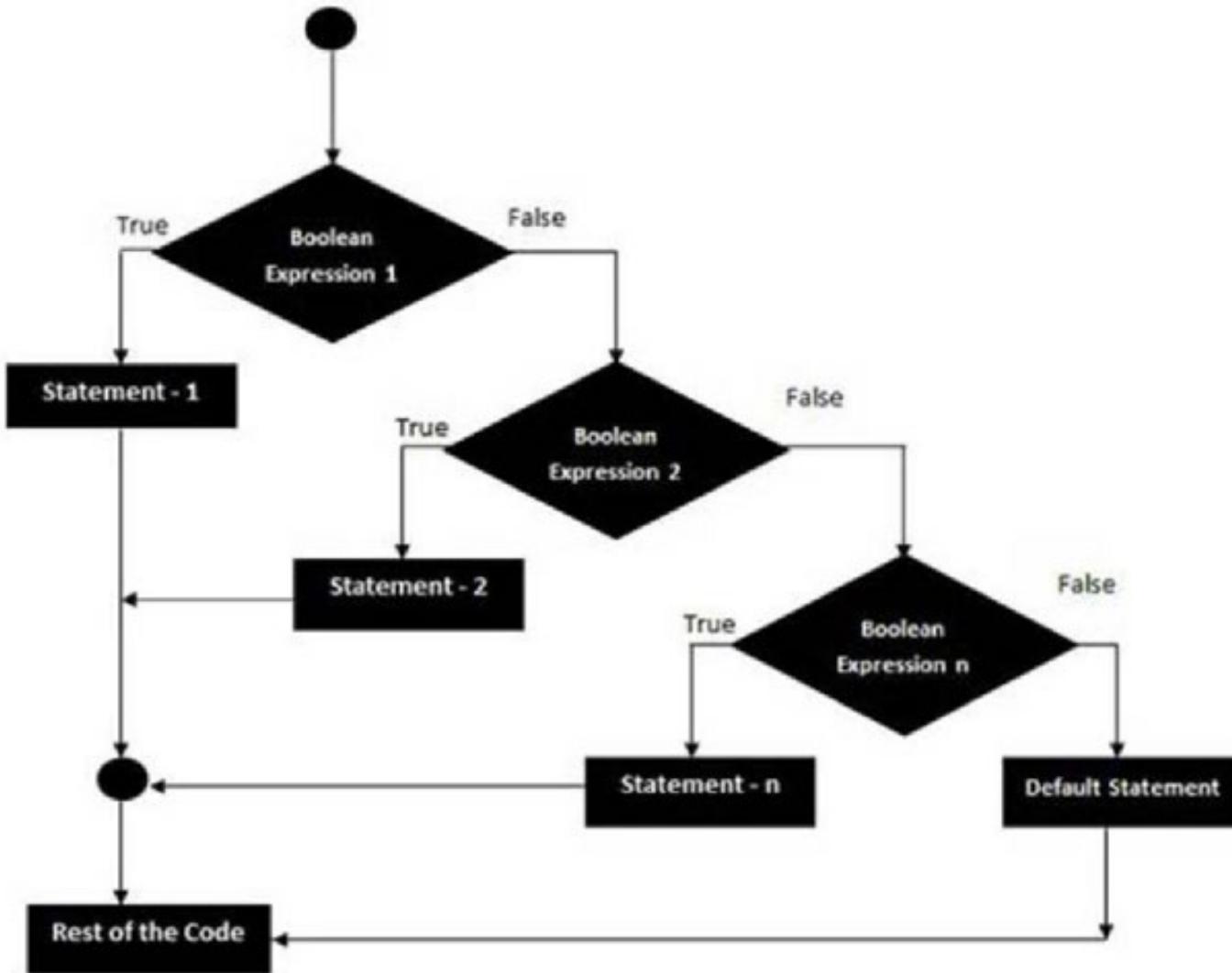


```
I0101.py x
myInt = 100
if myInt == 50:
    print("Group : 1")
elif myInt == 100:
    print("Group : 2")
elif myInt == 150:
    print("Group : 3")
elif myInt == 200:
    print("Group : 4")
else:
    print("False condition")
print(myInt)

Run I0101
C:\Users\Siriluck\AppData\Local\Programs\Python\Python36-32
Group : 2
100
```

Flowchart : else if

58



Flowchart : else if

59

```
if expression1:  
    statement(s)  
    if expression2:  
        statement(s)  
        elif expression3:  
            statement(s)  
        else  
            statement(s)  
        elif expression4:  
            statement(s)  
    else:  
        statement(s)
```

Flowchart : else if

60

```
myInt = 100

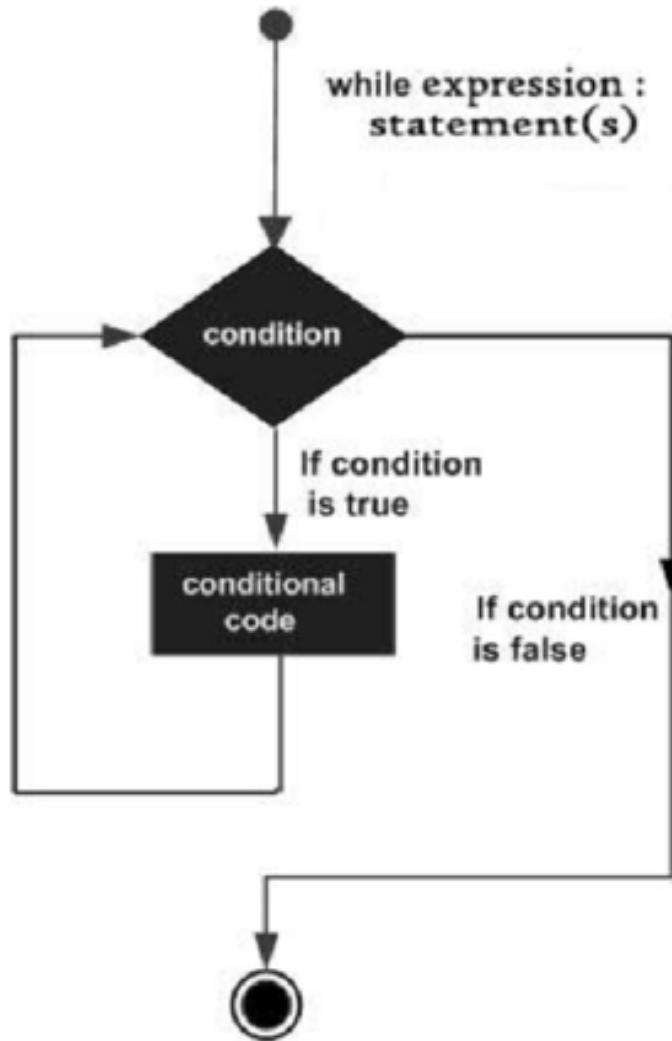
if myInt < 200:
    print("myInt is less than 200")
    if myInt == 150:
        print("Which is 150")
    elif myInt == 100:
        print("Which is 100")
    else:
        print("False")
else:
    print("myInt is more than 200")
```

The screenshot shows a Python development environment with the following details:

- Code Editor:** A window titled "I0101.py" containing the provided Python script. The code uses indentation for readability.
- Output Window:** A "Run" window titled "I0101" showing the execution results:
 - Line 1: myInt = 100
 - Line 2: (empty)
 - Line 3: if myInt < 200:
 - Line 4: print("myInt is less than 200")
 - Line 5: if myInt == 150:
 - Line 6: print("Which is 150")
 - Line 7: elif myInt == 100:
 - Line 8: print("Which is 100")
 - Line 9: else:
 - Line 10: print("False")
 - Line 11: (empty)
 - Line 12: print("myInt is more than 200")
- File Explorer:** Shows a folder named "untitled" containing "I0101.py". It also lists "External Libraries".

Flowchart : while loop

61



Flowchart : while loop

62

*while expression:
statement(s)*

Flowchart : while loop

63

```
count = 0
while(count<10):
    print("The count is",count)
    count+=1;
```

The screenshot shows a Python code editor window with a file named 'l0101.py'. The code is a simple while loop that prints the value of 'count' from 0 to 9. The output window below the editor shows the printed results.

```
l0101.py x
untitled C:\Users\Siriluck\PycharmProjects\untitled\src
l0101.py
External Libraries
```

```
1 count = 0
2 while(count<10):
3     print("The count is",count)
4     count+=1;
5
```

Run	l0101
▶	C:\Users\Siriluck\AppData\Local\Programs\Python\Python37\python.exe l0101.py
↑	The count is 0
↓	The count is 1
	The count is 2
	The count is 3
	The count is 4
	The count is 5
	The count is 6
	The count is 7
	The count is 8
	The count is 9

range

64

range(stop)

range([start], stop[, step])

```
1 >>> # One parameter
2 >>> for i in range(5):
3 ...     print(i)
4 ...
5 0
6 1
7 2
8 3
9 4
```

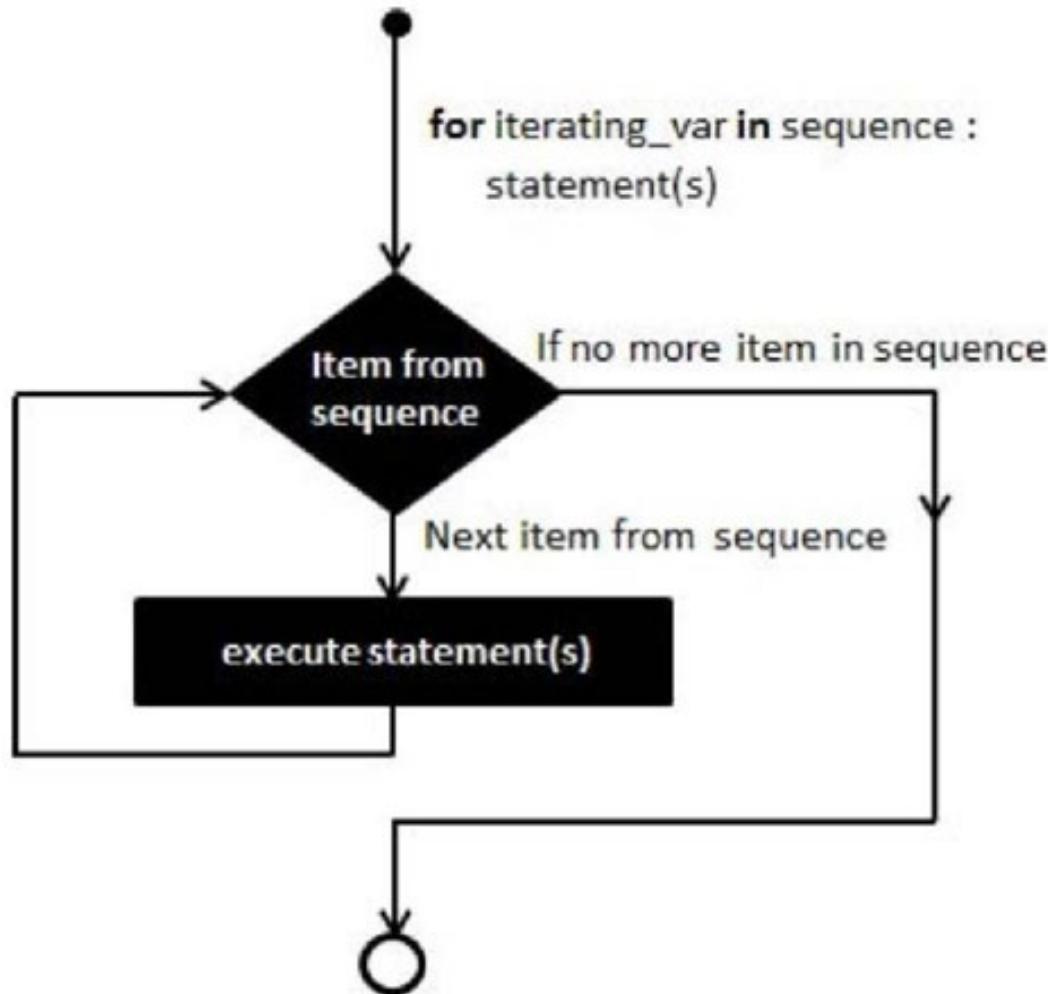
```
10 >>> # Two parameters
11 >>> for i in range(3, 6):
12 ...     print(i)
13 ...
14 3
15 4
16 5
```

```
17 >>> # Three parameters
18 >>> for i in range(4, 10, 2):
19 ...     print(i)
20 ...
21 4
22 6
23 8
```

```
24 >>> # Going backwards
25 >>> for i in range(0, -10, -2):
26 ...     print(i)
27 ...
28 0
29 -2
30 -4
31 -6
32 -8
```

Flowchart : for loop

65



Flowchart : for loop

66

```
for iterating_var in sequence:  
    statements(s)
```

Flowchart : for loop

67

```
for n in range(5):  
    print("n is",n)
```

The screenshot shows a Python code editor interface. On the left, there's a file tree with an 'untitled' folder containing 'l0101.py'. The main window displays the code:

```
l0101.py x  
1 for n in range(5):  
2     print("n is",n)  
3  
4
```

Below the code, there's a 'Run' button followed by the output of the program:

```
C:\Users\Siriluck\AppData\Local\Pr  
n is 0  
n is 1  
n is 2  
n is 3  
n is 4
```

The interface includes standard Python editor icons for file operations, code navigation, and run control.

Flowchart : Nested loop

68

```
for iterating_var in sequence:  
    for iterating_var in sequence:  
        statements(s)  
    statements(s)
```

```
while expression:  
    while expression:  
        statement(s)  
    statement(s)
```

Flowchart : Nested loop

69

```
n = 0
i = 0

while n<5:
    while i<n:
        print("When n is ",n,"i is ",i)
        i += 1
    n += 1
```

The screenshot shows the PyCharm IDE interface. The top bar displays the file name 'I0101.py'. The code editor window contains the following Python script:

```
1 n = 0
2 i = 0
3
4 while n<5:
5     while i<n:
6         print("When n is ",n,"i is ",i)
7         i += 1
8     n += 1
```

Below the code editor is the 'Run' tool window, which shows the output of the program:

```
C:\Users\Siriluck\AppData\Local\Programs\Python\Python3
▶ When n is  1 i is  0
▶ When n is  2 i is  1
▶ When n is  3 i is  2
▶ When n is  4 i is  3
```

Flowchart : Nested loop

70

```
n = 0
i = 0

for n in range(0,4,2):
    for i in range(2):
        print("When n is ",n,"i is ",i)
```

The screenshot shows a Python code editor interface with the following details:

- File Explorer:** Shows a folder named "untitled" containing a file named "I0101.py".
- Code Editor:** Displays the following Python code:

```
1  n = 0
2  i = 0
3
4  for n in range(0,4,2):
5      for i in range(2):
6          print("When n is ",n,"i is ",i)
7
```
- Run Tab:** Shows the output of running the code:

```
Run I0101
C:\Users\Siriluck\AppData\Local\Programs\Python\Python3
When n is  0 i is  0
When n is  0 i is  1
When n is  2 i is  0
When n is  2 i is  1
```

Question

71

- เขียนโปรแกรมสำหรับคำนวณเงินของพนักงานจากการระยะเวลาในการทำงาน โดยมีเงื่อนไขดังต่อไปนี้
- จำนวนชั่วโมงน้อยกว่าหรือเท่ากับ 8 ชั่วโมง ชั่วโมงละ 50 บาท
 - จำนวนชั่วโมงมากกว่า 8 ชั่วโมง จำนวนที่มากกว่า 8 ชั่วโมง คิดชั่วโมงละ 80 บาท
 - หากทำงานมากกว่า 12 ชั่วโมง จะได้รับเงินทั้งหมด 800 บาท

Enter working time : 4

Total wage is 200

Enter working time : 11

Total wage is 640

Enter working time : 100

Total wage is 800

Question

72

- เขียนโปรแกรมสำหรับรับตัวเลขไปเรื่อย ๆ จนกว่าผู้ใช้จะป้อนเลข 0 เข้ามาในระบบ พร้อมทั้งหาผลรวมของตัวเลขที่ป้อนเข้ามาทั้งหมด : โดยใช้ **loop while**

Enter number : 3.2

Enter number : 53

Enter number : 1

Enter number : 3

Enter number : 0

Sum : 60.2



สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)
Technology Promotion Association (Thailand-Japan)



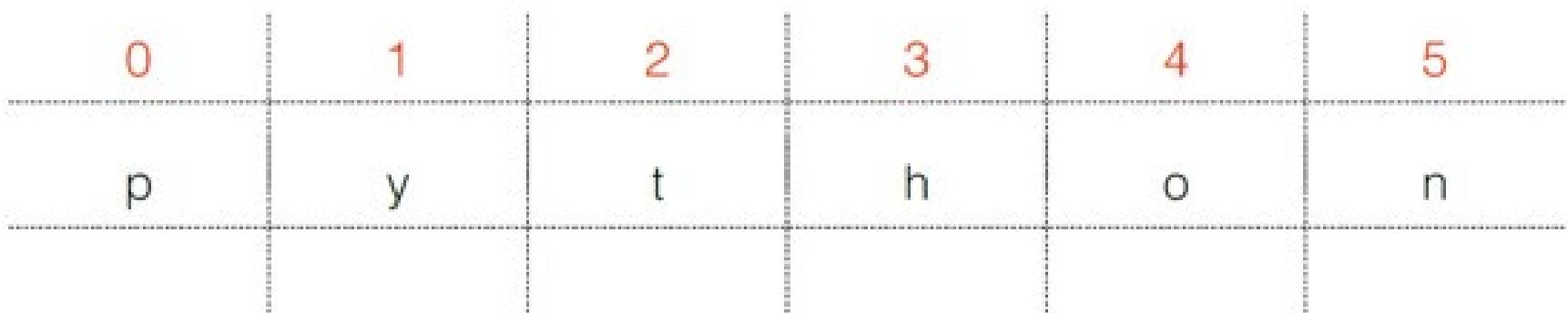
สถาบันเทคโนโลยีไทย-ญี่ปุ่น
Thai-Nichi Institute of Technology
泰日工業大学

String และ List

Access values

74

```
var = "python"
```



String

75

Operator	Description	Example
+	Concatenation - Adds values on either side of the operator	a + b will give HelloPython
	Repetition - Creates new strings, concatenating multiple copies of the same string	a*2 will give -HelloHello
[]	Slice - Gives the character from the given index	a[1] will give e
[:]	Range Slice - Gives the characters from the given range	a[1:4] will give ell
in	Membership - Returns true if a character exists in the given string	H in a will give 1
not in	Membership - Returns true if a character does not exist in the given string	M not in a will give 1
%	Format - Performs String formatting	See at next section

String : + , *

76

word = "Py"

print(word + 'thon')

The screenshot shows the PyCharm IDE interface. On the left is the Project tool window with a single file 'I0101.py' in the 'untitled' folder. The code editor window contains the following Python code:

```
1 word = "Py"
2 print(word + 'thon')
```

The Run tool window at the bottom shows the output of running the script:

```
C:\Users\Siriluck\AppData\Local\Programs\Python\Python
PyPyPyPy
Process finished with exit code 0
```

word = "Py"

print(word*4)

The screenshot shows the PyCharm IDE interface. On the left is the Project tool window with a single file 'I0101.py' in the 'untitled' folder. The code editor window contains the following Python code:

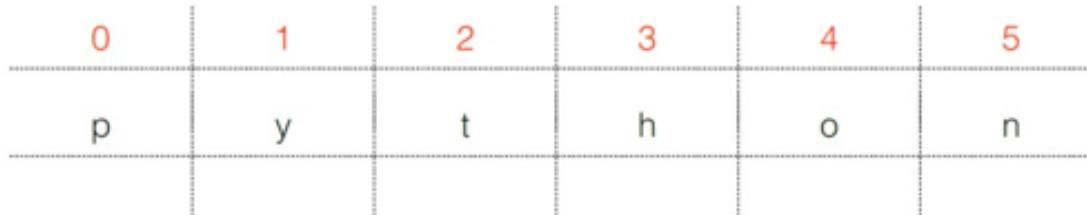
```
1 word = "Py"
2 print(word*4)
```

The Run tool window at the bottom shows the output of running the script:

```
C:\Users\Siriluck\AppData\Local\Programs\Python\Python
PyPyPyPy
Process finished with exit code 0
```

String : [] , [:]

77



word = "Python"

print(word[3])

word = "Python"

print(word[2:4])

```
1 word = "Python"  
2 print(word[3])  
3  
4  
5
```

Run l0101

C:\Users\Siriluck\AppData\Local\Pr
h

```
1 word = "Python"  
2 print(word[2:4])  
3  
4  
5
```

Run l0101

C:\Users\Siriluck\AppData\Local\Pr
th

String : %

78

Format Symbol	Conversion
%c	character
%s	string conversion via str() prior to formatting
%d , %i	signed decimal integer
%f	floating point real number

```
1 name = input("Enter your name : ")
2 w = int(input("Enter your weight : "))
3
4 print ("My name is %s and weight is %d kg!" % (name, w))
5
```

Run  I0101

```
C:\Users\Siriluck\AppData\Local\Programs\Python\Python36
Enter your name : Zara
Enter your weight : 30
My name is Zara and weight is 30 kg!
```

String : find()

79

Description

It determines if string *str* occurs in string, or in a substring of string if starting index *beg* and ending index *end* are given.

Syntax

```
str.find(str, beg=0, end=len(string))
```

Parameters

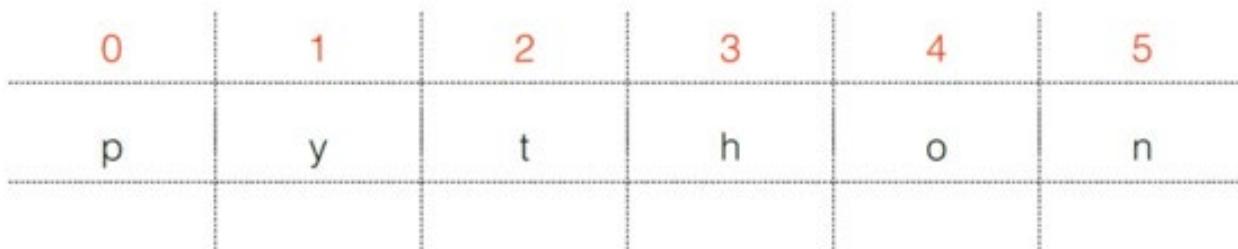
- **str** -- This specifies the string to be searched.
- **beg** -- This is the starting index, by default its 0.
- **end** -- This is the ending index, by default its equal to the length of the string.

Return Value

Index if found and -1 otherwise.

String : find()

80



```
1 word = "python"
2 str = "th"
3
4 print (word.find(str))
5 print (word.find(str,1))
6 print (word.find(str,1,3))
7 print (word.find(str,5))
```

Run l0101

	2
	2
	-1
	-1

List

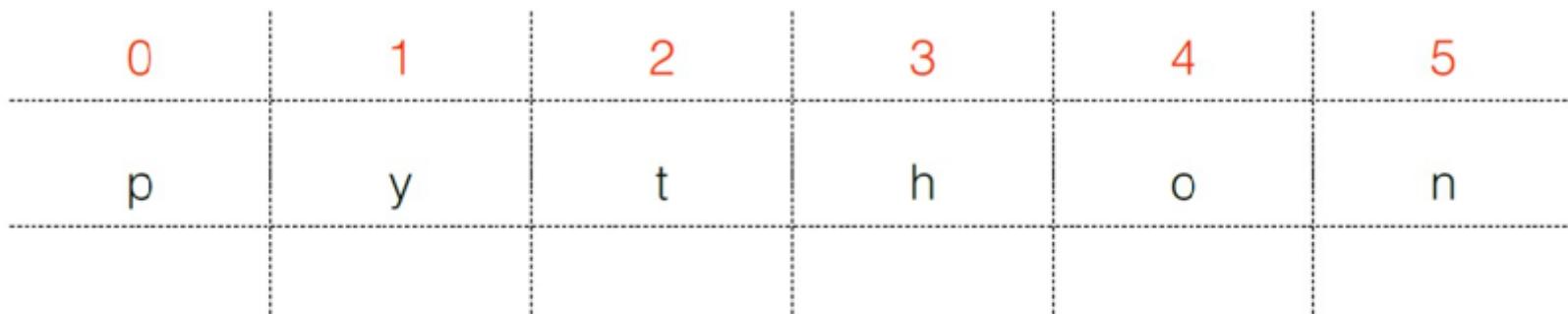
81

```
list1 = ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
```

String vs List

82

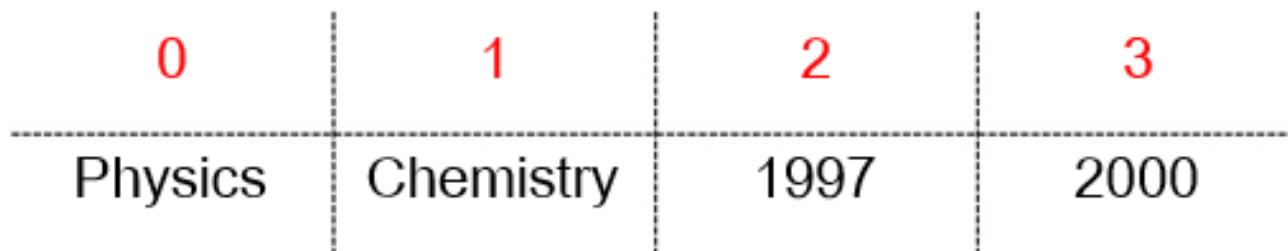
word = “python”



```
list1 = ['physics', 'chemistry', 1997, 2000]  
list2 = [1, 2, 3, 4, 5 ]  
list3 = ["a", "b", "c", "d"]
```

List [] , [:]

83



```
list_name = ['physics', 'chemistry', 1997, 2000]
```

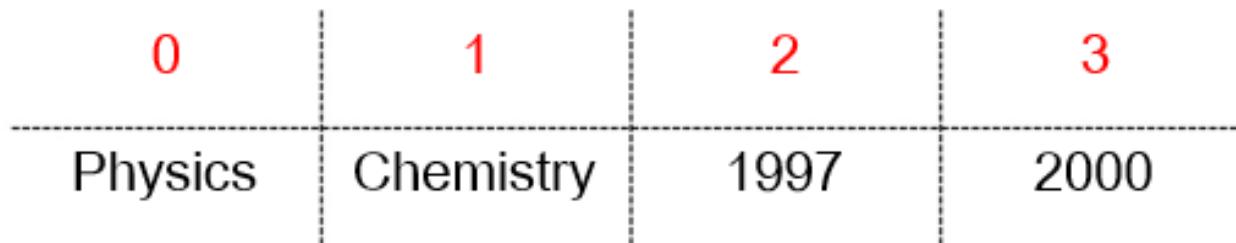
```
print(list_name[1])  
print(list_name[1:4])
```



```
C:\Users\Siriluck\AppData\  
chemistry  
['chemistry', 1997, 2000]
```

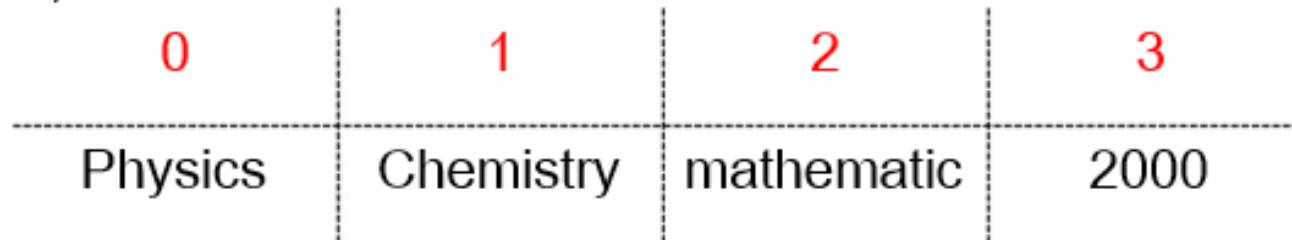
List : update value

84



```
list_name = ['physics', 'chemistry', 1997, 2000]
```

```
list_name[2] = "mathematic"  
print(list_name)
```



```
C:\Users\Siriluck\AppData\Local\Programs\Pyt  
['physics', 'chemistry', 'mathematic', 2000]
```

List : Nested list

85

1	2	3	4
5	6	7	8
9	10	11	12

```
matrix = [ [1,2,3,4], [5,6,7,8], [9,10,11,12] ]  
  
print("matrix[1][2] : ", matrix[1][2])
```



```
C:\Users\Siriluck\A:  
matrix[1][2] : 7
```

List : input nested list : append

86

```
matrix = []

for i in range(4):
    n = input("Enter name : ")
    matrix.append(n)

print(matrix)
```



```
C:\Users\Siriluck\AppData\Local\Prog
Enter name : nok
Enter name : jubjib
Enter name : sanook
Enter name : well
['nok', 'jubjib', 'sanook', 'well']
```

List : input nested list : append

87

```
matrix = []

for i in range(3):
    matrix.append([])
    for j in range(3):
        n = int(input("Enter number "))
        matrix[i].append(n)
print(matrix)
```



```
C:\Users\Siriluck\AppData\Local\P...
Enter number 1
Enter number 2
Enter number 3
Enter number 4
Enter number 5
Enter number 6
Enter number 7
Enter number 8
Enter number 9
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

List : basic operations

88

Python Expression	Results	Description
<code>len([1, 2, 3])</code>	3	Length
<code>[1, 2, 3] + [4, 5, 6]</code>	<code>[1, 2, 3, 4, 5, 6]</code>	Concatenation
<code>['Hi!'] * 4</code>	<code>['Hi!', 'Hi!', 'Hi!', 'Hi!']</code>	Repetition
<code>3 in [1, 2, 3]</code>	True	Membership
<code>max([1, 2, 3])</code>	3	Maximum value
<code>min([1, 2, 3])</code>	1	Minimum value

List : length

89

Python Expression	Results	Description
max([1, 2, 3])	3	Maximum value

```
list_name = [1, 4, 6, 2, 3, 8, 10]
```

```
print("Max : ",max(list_name))  
print("Min : ",min(list_name))
```



```
C:\Users\Siriluck\AppData\Local\  
    Max : 10  
    Min : 1
```

List : length

90

Python Expression	Results	Description
<code>len([1, 2, 3])</code>	3	Length

```
list_name = ['physics', 'chemistry', 'mathematic', 1997, 2000]
```

```
print(list_name)
print(len(list_name))
```



```
C:\Users\Siriluck\AppData\Local\Programs\Python\Pyt
['physics', 'chemistry', 'mathematic', 1997, 2000]
5
```

List : delete

91

0 1 2 3 4

Physics Chemistry mathematic 1997 2000

```
list_name = ['physics', 'chemistry', 'mathematic', 1997, 2000]
```

```
print(list_name)
del(list_name[2])
print(list_name)
```



```
C:\Users\Siriluck\AppData\Local\Programs\Python\Py
['physics', 'chemistry', 'mathematic', 1997, 2000]
['physics', 'chemistry', 1997, 2000]
```

0 1 2 3

Physics Chemistry 1997 2000



สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)
Technology Promotion Association (Thailand-Japan)



สถาบันเทคโนโลยีไทย-ญี่ปุ่น
Thai-Nichi Institute of Technology
泰日工業大学

Function

Function : Definition

93

```
def function_name( parameters ):  
    statement  
    ....  
    statement  
    return expression
```

Function

94

```
def happyBirthdayEmily():
    print("Happy Birthday to you!")
    print("Happy Birthday to you!")
    print("Happy Birthday, dear Emily.")
    print("Happy Birthday to you!")
```

A screenshot of a Python code editor. The code is identical to the one above, defining a function named `happyBirthdayEmily` that prints four lines of text. The code is numbered from 1 to 7. The editor has a light gray background with dark gray vertical and horizontal scroll bars. At the bottom, there is a toolbar with icons for file operations and a status bar displaying the path `C:\Users\Siriluck\AppData\Local\Programs\Pyt`. The status bar also shows a small icon and the number 10101.

```
1 def happyBirthdayEmily():
2     print("Happy Birthday to you!")
3     print("Happy Birthday to you!")
4     print("Happy Birthday, dear Emily.")
5     print("Happy Birthday to you!")
6
7
```

Function : Executable

95

```
1     def happyBirthdayEmily():
2         print("Happy Birthday to you!")
3         print("Happy Birthday to you!")
4         print("Happy Birthday, dear Emily.")
5         print("Happy Birthday to you!")
6
7     happyBirthdayEmily()
```

I0101

```
C:\Users\Siriluck\AppData\Local\Programs\Pyt
Happy Birthday to you!
Happy Birthday to you!
Happy Birthday, dear Emily.
Happy Birthday to you!

Process finished with exit code 0
```

Function

96

```
1 def happyBirthdayEmily():
2     print("Happy Birthday to you!")
3     print("Happy Birthday to you!")
4     print("Happy Birthday, dear Emily.")
5     print("Happy Birthday to you!")
6
7 def happyBirthdayJack():
8     print("Happy Birthday to you!")
9     print("Happy Birthday to you!")
10    print("Happy Birthday, dear Jack.")
11    print("Happy Birthday to you!")
12
13 happyBirthdayEmily()
14 happyBirthdayJack()
15
```

The screenshot shows a code editor window with Python code. The code defines two functions: `happyBirthdayEmily()` and `happyBirthdayJack()`. Both functions print four lines of text: "Happy Birthday to you!", "Happy Birthday to you!", "Happy Birthday, dear [Name].", and "Happy Birthday to you!". Below the code editor, there is a terminal window titled "10101" showing the execution of the code. The terminal output displays the same four lines of text for each function call.

```
C:\Users\Siriluck\AppData\Local\Programs\Python  
↑ Happy Birthday to you!  
↓ Happy Birthday to you!  
→ Happy Birthday, dear Emily.  
→ Happy Birthday to you!  
→ Happy Birthday to you!  
→ Happy Birthday to you!  
→ Happy Birthday, dear Jack.  
→ Happy Birthday to you!
```

Function

97

```
1 def happyBirthday(name):
2     print("Happy Birthday to you!")
3     print("Happy Birthday to you!")
4     print("Happy Birthday, dear", name, ".")
5     print("Happy Birthday to you!")
6
7 happyBirthday("Emily")
8 happyBirthday("Jack")
```

l0101

```
C:\Users\Siriluck\AppData\Local\Programs\Python 3.8\python.exe C:/Users/Siriluck/Desktop/happy_birthday.py
Happy Birthday to you!
Happy Birthday to you!
Happy Birthday, dear Emily .
Happy Birthday to you!
Happy Birthday to you!
Happy Birthday to you!
Happy Birthday, dear Jack .
Happy Birthday to you!
```

Function

98

```
1     def passVal(myVal):  
2         myVal = 2  
3             print("Inside function: ", myVal)  
4  
5         myVal = 1  
6         passVal(myVal)  
7         print("Outside Function ", myVal)  
8  
9
```

l0101

```
C:\Users\Siriluck\AppData\Local\Programs\  
Inside function: 2  
Outside Function 1
```

Function

99

```
def add(x,y) :  
    z = x+y  
    return z  
  
a = int(input("Enter number 1 : "))  
b = int(input("Enter number 2 : "))  
  
c = add(a,b)  
print ("Add : ",c)
```



```
C:\Users\Siriluck\Ap  
Enter number 1 : 12  
Enter number 2 : 3  
Add : 15
```

Function

100

```
def add_sub(x,y) :  
    z = x+y  
    e = x-y  
    return z,e
```

```
a = int(input("Enter number 1 : "))  
b = int(input("Enter number 2 : "))
```

```
c,d = add_sub(a,b)  
print ("Add : ",c)  
print ("Sub : ",d)
```



```
C:\Users\Siriluck\Appl  
Enter number 1 : 12  
Enter number 2 : 4  
Add : 16  
Sub : 8
```

Question

101

- ให้เขียนฟังก์ชันสำหรับ การบวก การลบ การคูณ และ การหาร ค่า 2 ค่า จากนั้นส่งผลลัพธ์กลับมาแสดงผล
(เขียนรวมในฟังก์ชันเดียว หรือเขียนแยกรายฟังก์ชันก็ได้)

Enter number 1 : 12

Enter number 2 : 3

Plus : 15

Minus : 9

Multiply : 36

Divide : 4.0