Easy Learning Python 3

Python 3 for Beginner's Guide

Easy Learning
Python 3

YANG HU

Simple is the beginning of wisdom. This book briefly explain the concept and vividly cultivate programming interest, this book for beginner fast learning Python 3 programming.

http://en.verejava.com

ISBN: 9781092328128

CONTENTS

1. Python Installation
2. Python Basic Concepts
    2.1 Hello World
    2.2 Variable
    2.3 Basic Data Type
    2.4 Data Type Conversion

3. Python Operator
    3.1 Arithmetic Operator
    3.2 Function
    3.3 Class
4. Control Statement
    4.1 Relational Operator
    4.2 Assignment Operator
    4.3 Logical Operators
    4.4 If Conditional Statements
    4.5 While Loop
    4.6 While Loop Fruit Game
    4.7 For Loop
    4.8 For Loop Bubble Ball
    4.9 Continue and break
5. Python Data Structure
    5.1 List
    5.2 Two-Dimensional List
    5.3 Find Gog Game
    5.4 Tuple
    5.5 Dictionary
    5.6 Set
    5.7 Iterator
    5.8 Generator
6. String
7. Modules And Packages
8. Date And Time
9. File and Input and Output I/O
10. Exception Handling

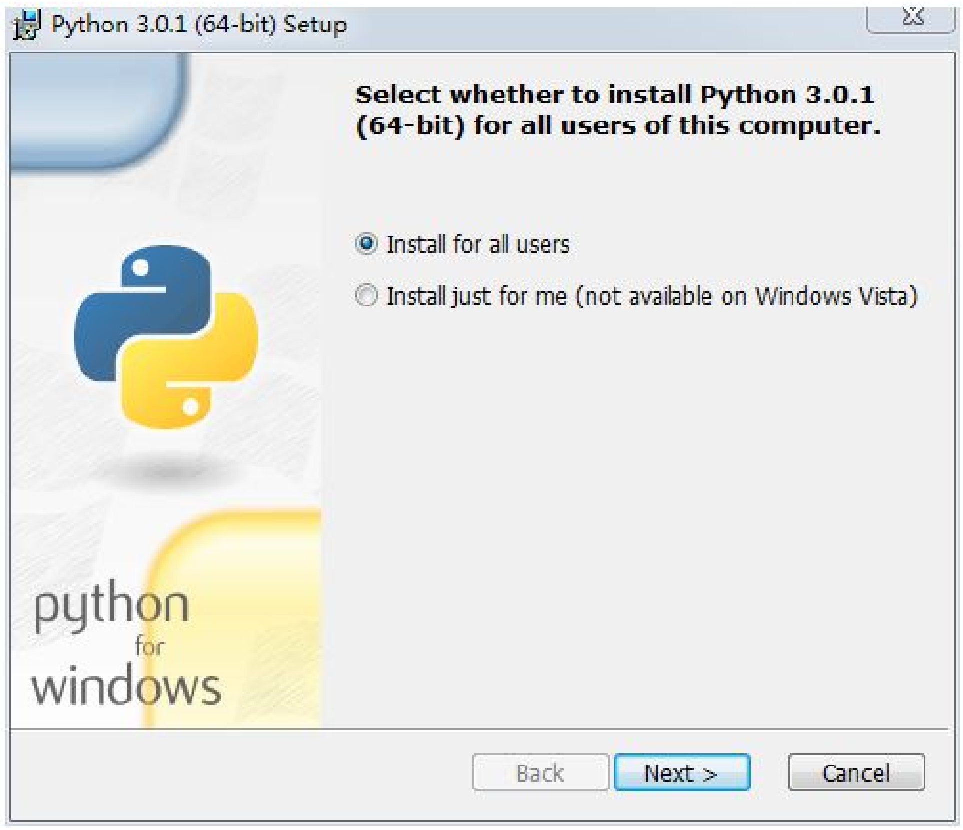
11. Regular Expression
12. Python 3 Object-Oriented Programming
    12.1 Create Class
    12.2 Encapsulation
    12.3 Constructor Method
    12.4 Inheritance
    12.5 Override and Polymorphism
    12.6 List and Class
    12.7 Dictionary and Class
13. Multithreading
14. Python PyMySQL MySQL

Python Installation

Download Python3.7.zip Development Tool

https://www.python.org/downloads/windows/
or
http://en.verejava.com/download.jsp?id=1

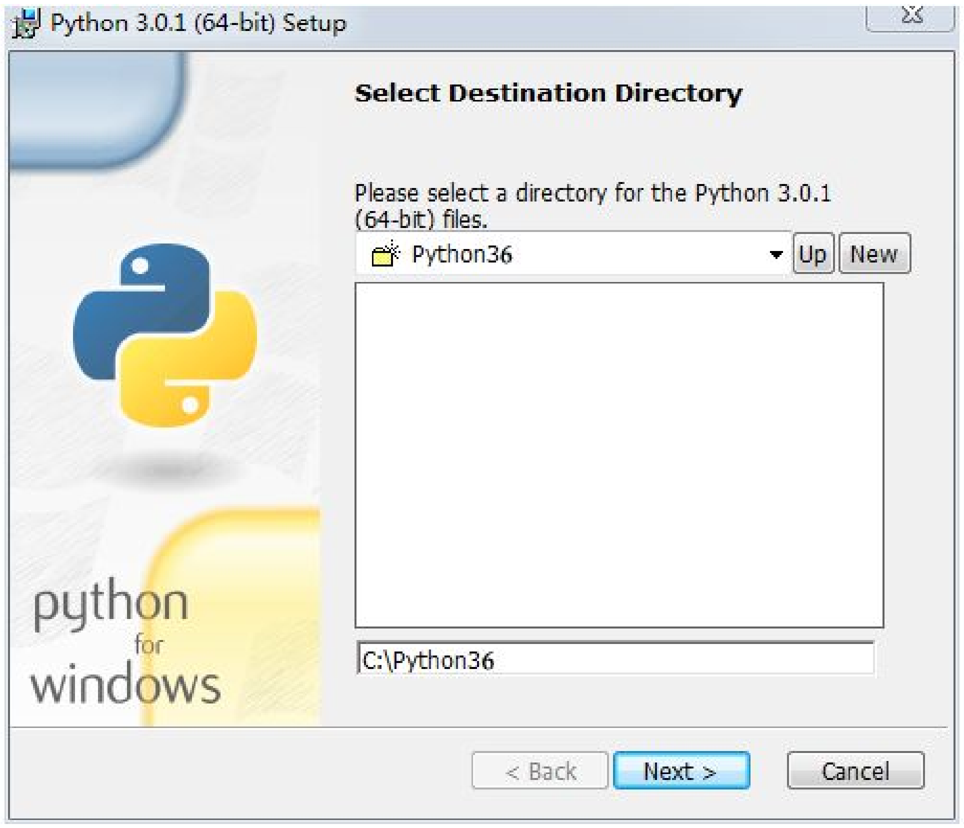
Unzip python-3.7.0-amd64.zip to python-3.7.0-amd64.exe


Python 3.0.1 (64-bit) Setup

Select whether to install Python 3.0.1 (64-bit) for all users of this computer.

[●] Install for all users
[ ] Install just for me (not available on Windows Vista)

Click Next


Python 3.0.1 (64-bit) Setup

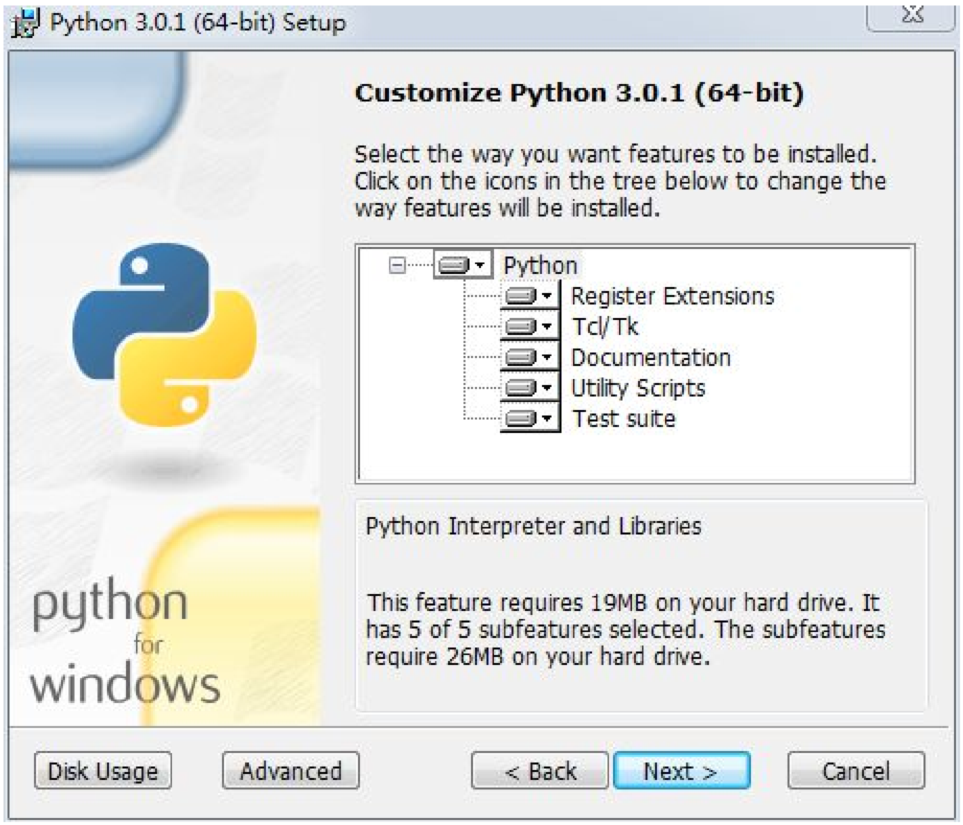
Select Destination Directory

Please select a directory for the Python 3.0.1 (64-bit) files.

Python36

C:\Python36

Installation Path: C:\Python36\ and then click Next



Customize Python 3.0.1 (64-bit)

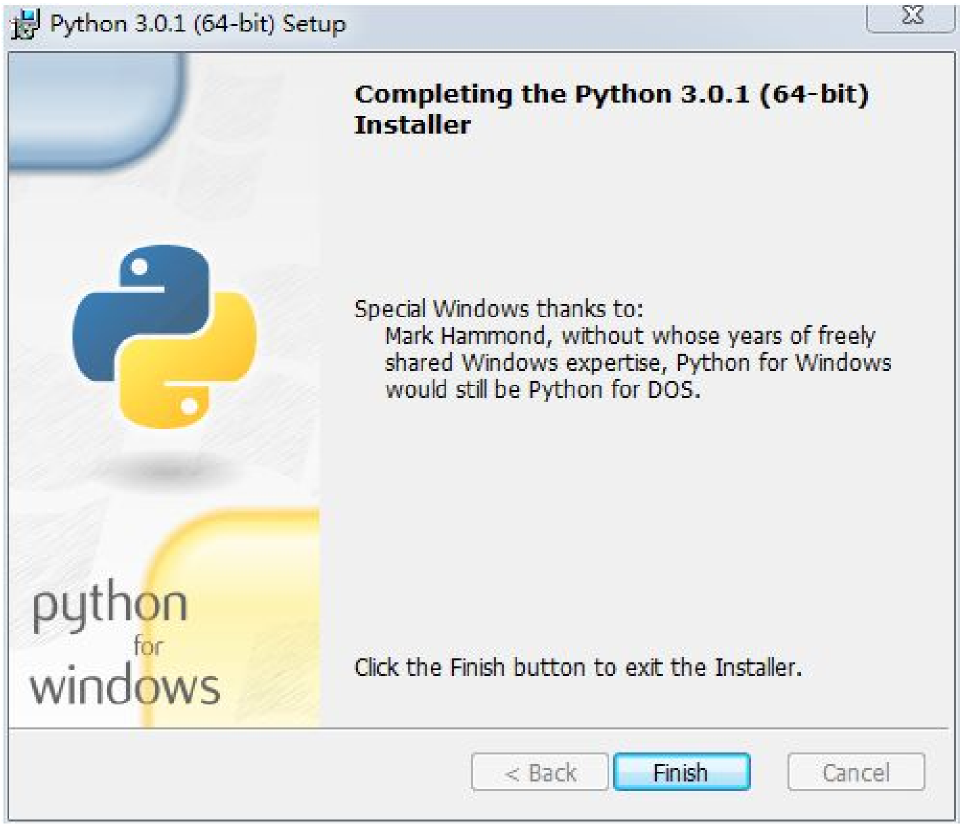
Select the way you want features to be installed. Click on the icons in the tree below to change the way features will be installed.

- Python
  - Register Extensions
  - Tcl/Tk
  - Documentation
  - Utility Scripts
  - Test suite

Python Interpreter and Libraries

This feature requires 19MB on your hard drive. It has 5 of 5 subfeatures selected. The subfeatures require 26MB on your hard drive.

Click Next


Python 3.0.1 (64-bit) Setup

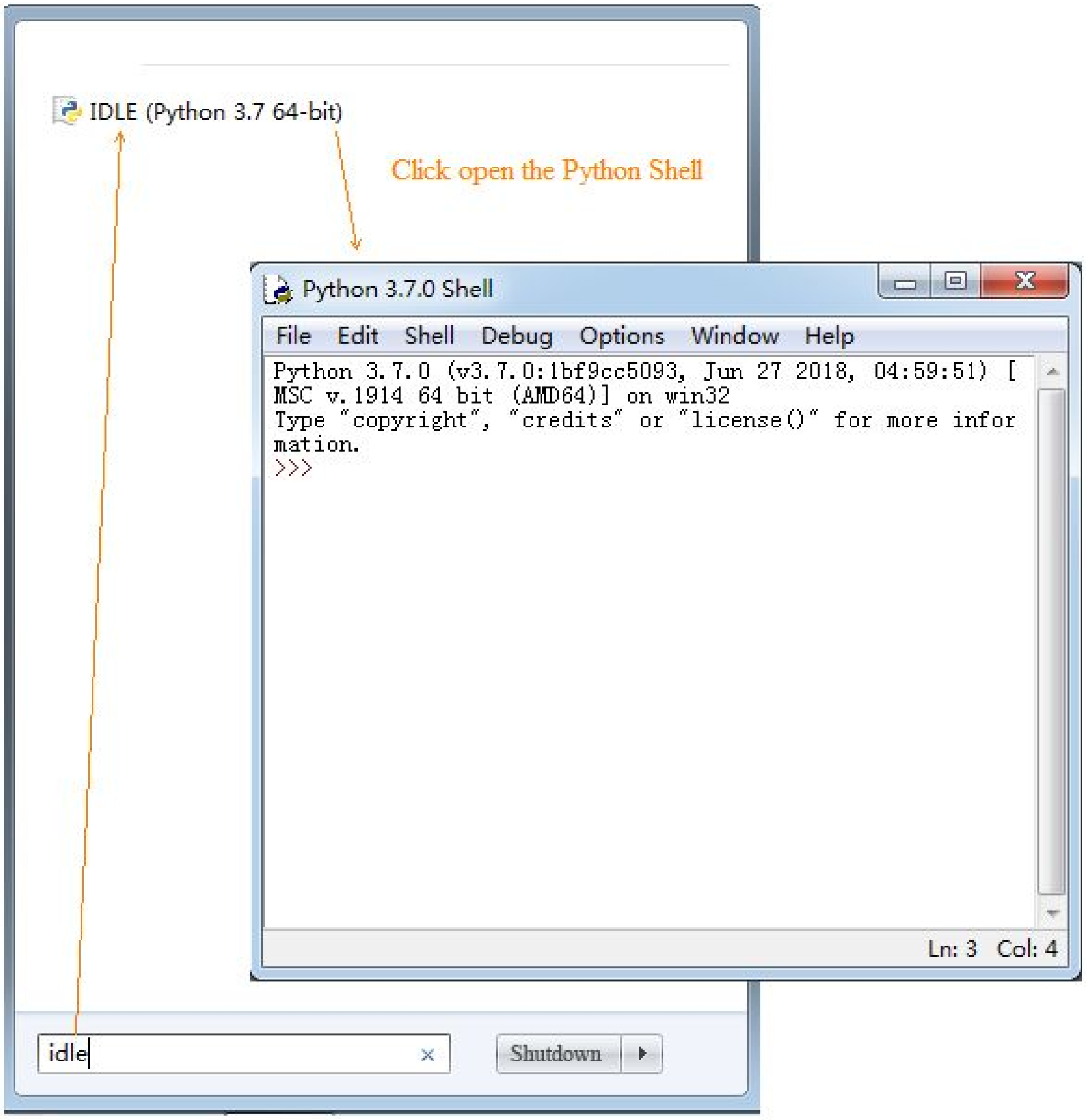
Completing the Python 3.0.1 (64-bit) Installer

Special Windows thanks to:
Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

Click the Finish button to exit the Installer.

Click Finish Installation Successfully

Open Python shell input idle in window search box.


IDLE (Python 3.7 64-bit)
Click open the Python Shell
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Ln: 3 Col: 4
idle
Shutdown

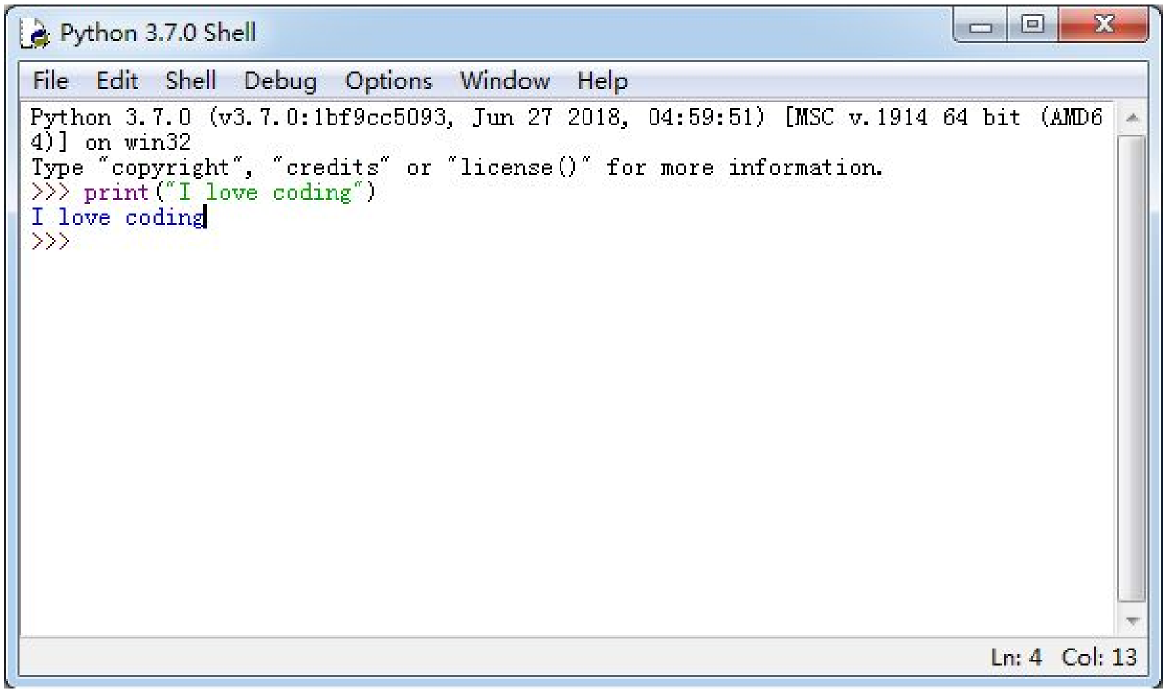
```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [
MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more infor
mation.
>>>
```

Create your first code :

```
>>> print("I love coding")
```

Press enter key output:

I love coding

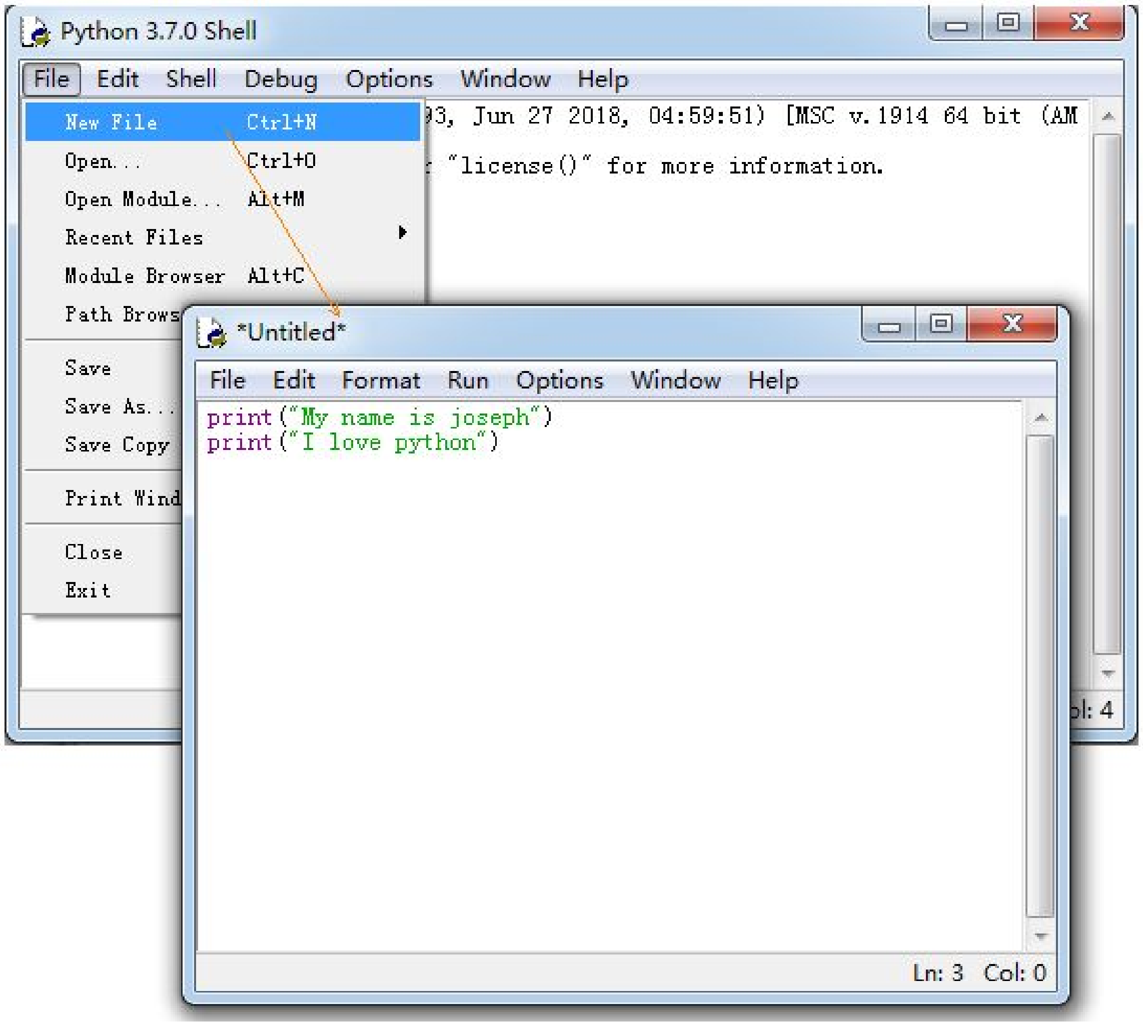


```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("I love coding")
I love coding
>>>
```

Create your first python file : hello.py

1. File -> New File input code:

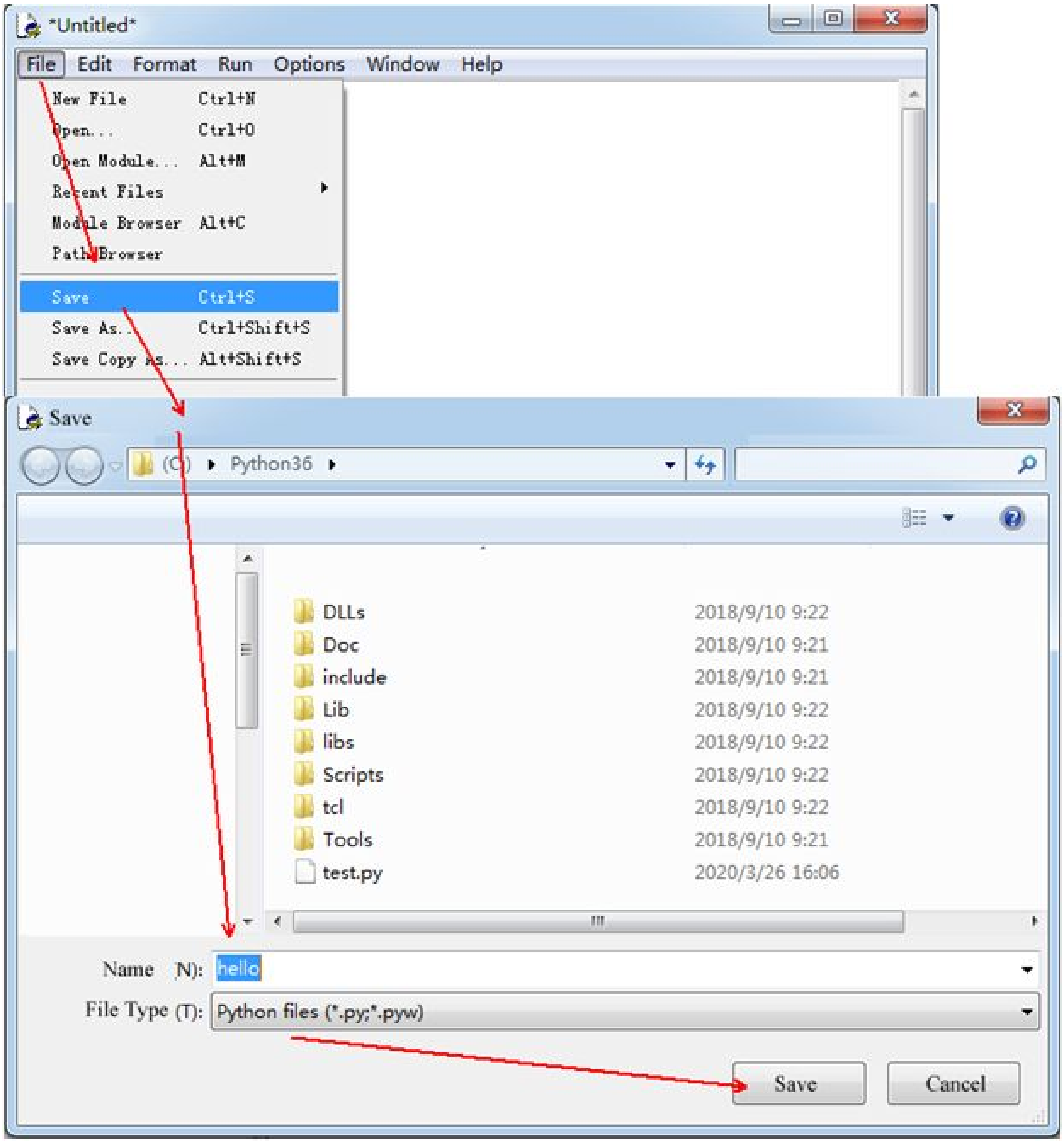
```
print("My name is joseph")
print("I love python")
```



```
93, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AM
 "license()" for more information.
```

```
print("My name is joseph")
print("I love python")
```

2. File -> Save: hello.py



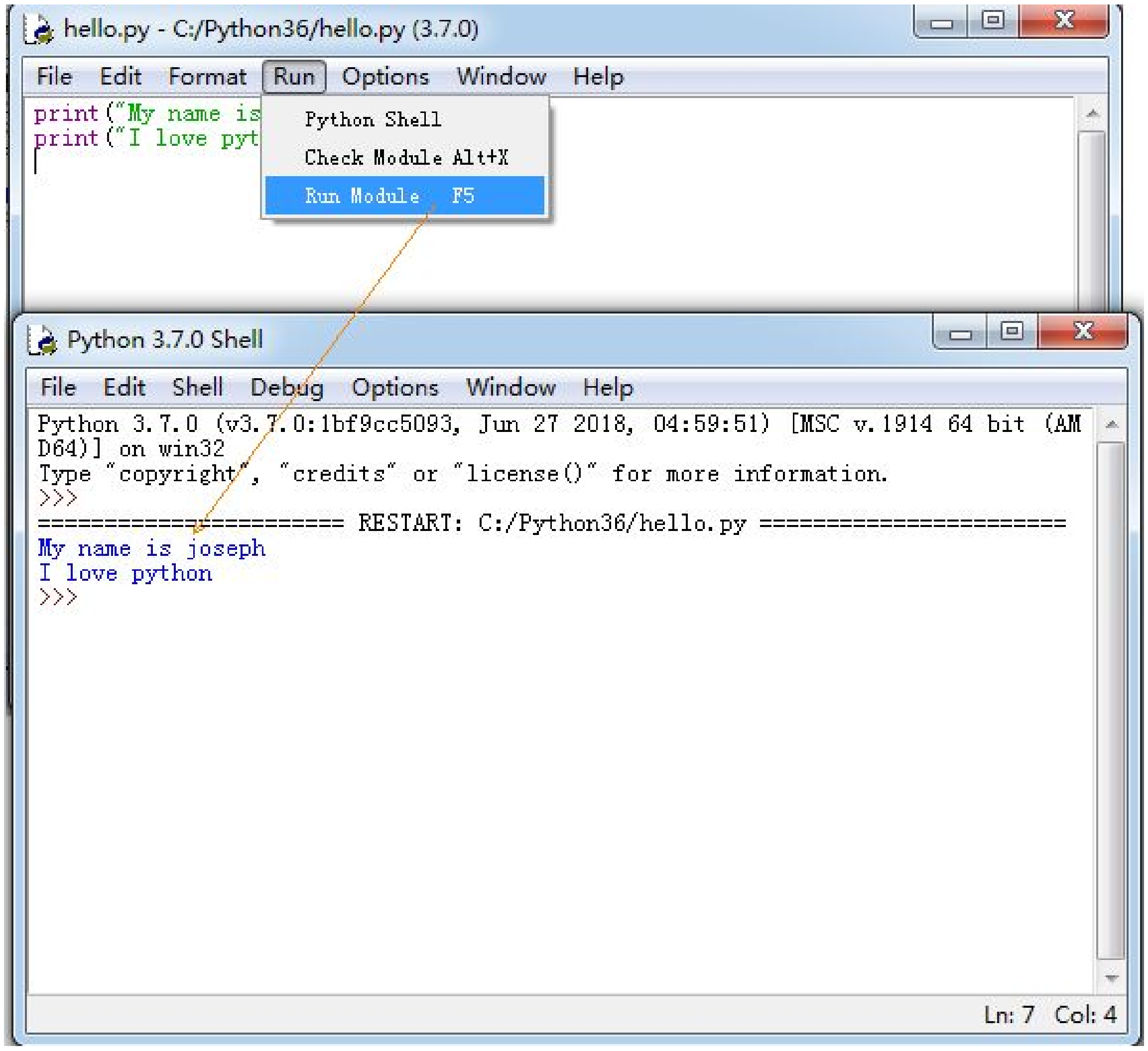
DLLs	2018/9/10 9:22
Doc	2018/9/10 9:21
include	2018/9/10 9:21
Lib	2018/9/10 9:22
libs	2018/9/10 9:22
Scripts	2018/9/10 9:22
tcl	2018/9/10 9:22
Tools	2018/9/10 9:21
test.py	2020/3/26 16:06

Name (N): hello

File Type (T): Python files (*.py;*.pyw)

Click Save Button : the hello.py the created in c:/Python36

3. Run hello.py


hello.py - C:/Python36/hello.py (3.7.0)
File Edit Format Run Options Window Help
Python Shell
Check Module Alt+X
Run Module F5
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Ln: 7 Col: 4

```
print("My name is
print("I love pyt
```

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
====================== RESTART: C:/Python36/hello.py ======================
My name is joseph
I love python
>>>
```

Hello World

# : Single line comment will not be executed by the program
print(): function prints the message to the screen and wrap a new line
print(end=""): prints the message to the screen and not wrap a new line
\n: wrap a new line
\t: a tab space

1.Create a python file : hello_world.py run in Python Shell

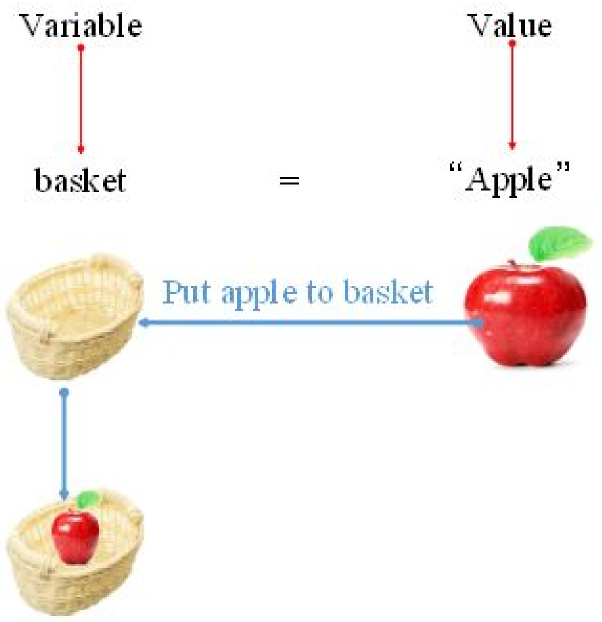
```python
print("Hello World");
print("A thousand miles ", end="") # end="" Print does not wrap
print("begins with a single step ")
print("\n Life is in time") # \n Wrap one line
print("\t Today is life, now is the power") # \t a tab distance
```

Result:

```
Hello World
A thousand miles begins with a single step

Life is in time
        Today is life, now is the power
```

Key programming specification
1. Do not put a semicolon at the end of the line; do not put two commands on the same line.
2. Indentation is consistent 4 spaces or tabs, preferably 4 spaces.
Because tabs may be different length in different editor.

Variable

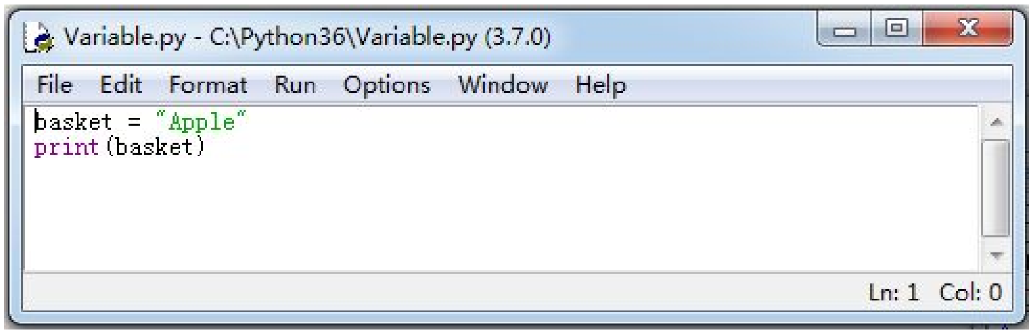


String: are surrounded by single quotation marks ' ', or double quotation marks " "

'Apple' is the same as "Apple".

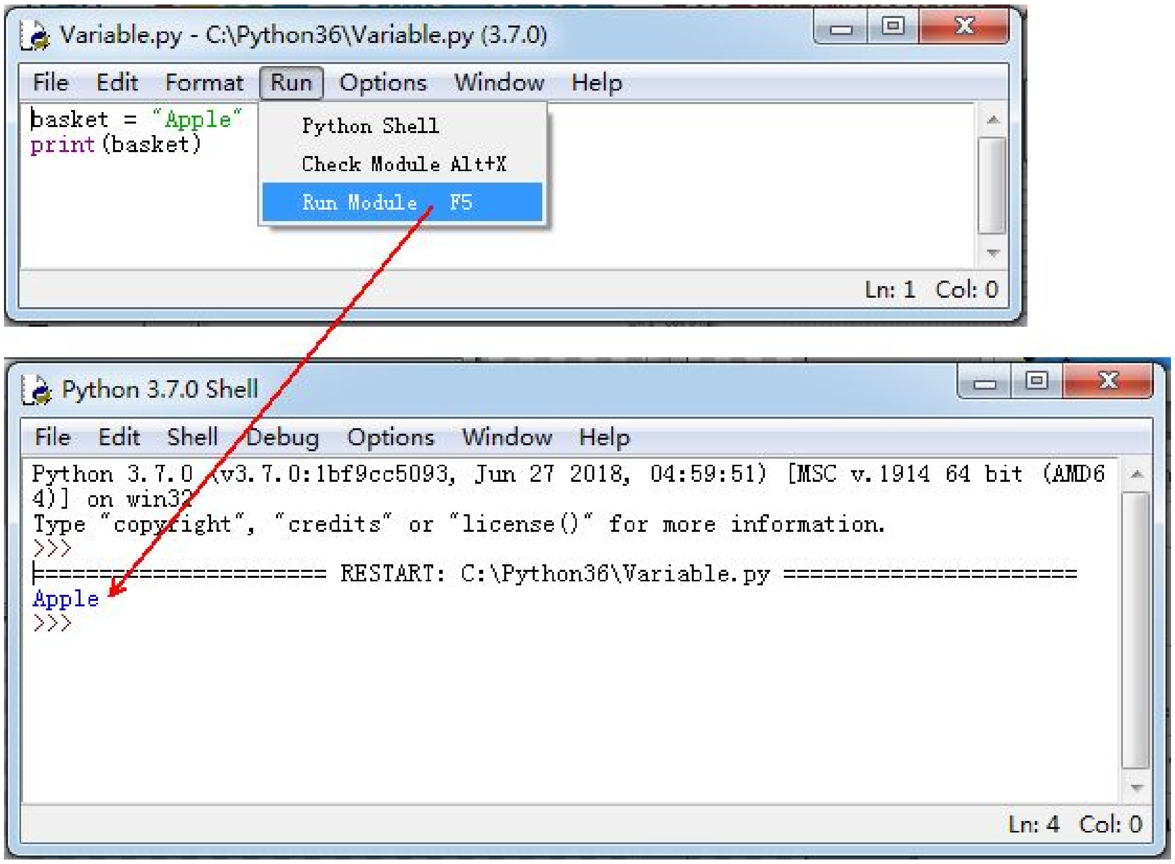
1.Create a python file : Variable.py run in Python Shell

```
basket = "Apple"
print(basket)
```


Variable.py - C:\Python36\Variable.py (3.7.0)
File Edit Format Run Options Window Help
Ln: 1 Col: 0

```
basket = "Apple"
print(basket)
```

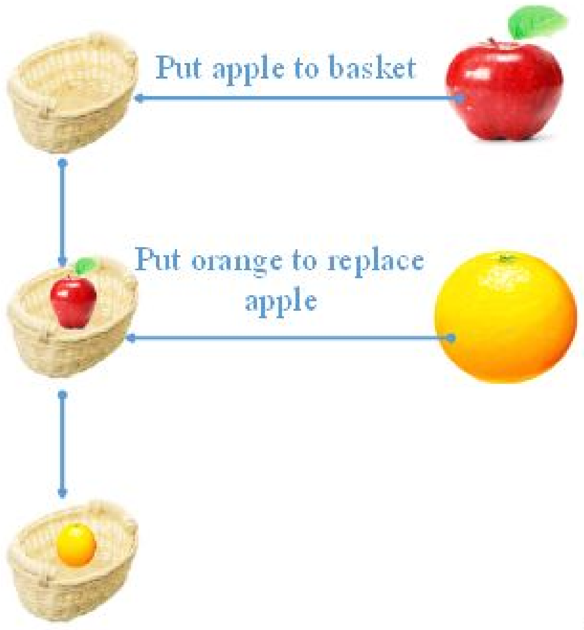
Run Result:


Variable.py - C:\Python36\Variable.py (3.7.0)
File Edit Format Run Options Window Help
Python Shell
Check Module Alt+X
Run Module F5
Ln: 1 Col: 0
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Ln: 4 Col: 0

```
basket = "Apple"
print(basket)
```

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===================== RESTART: C:\Python36\Variable.py =====================
Apple
>>>
```

2. Replace variable basket from "Apple" to "Orange"



```
basket = "Apple"
print(basket)

basket = "Orange"
print(basket)
```

Run Result:

```
Apple
Orange
```

Basic Data Type

1.Create a python file : datatype.py run in Python Shell

String: are surrounded by single quotation marks ' ', or double quotation marks " "
'Apple' is the same as "Apple".
Integer: Number data types store numeric values.
Float: floating point number.
Boolean: represent one of two values: True or False.

```
# integer variable
age = 20

# floating variable
money = 8000.0

# string variable
word = "waste time called imaginary"

# boolean variable
married=True

print (age)
print (money)
print (word)
print (married)
```

Result:

```
20
8000.0
waste time called imaginary
True
```

Data Type Conversion

1.Create a python file : conversion.py run in Python Shell

float(): function converts the specified value into a floating point number.
int(): function converts the specified value into an integer number.
+ : If two strings are added, it means that they are concatenated

```
c1=1
c2=10.3
# integer to float
c2=float(c1)
print(c2)

d1=1
d2=10.3
# float to integer requires conversion and may lose precision.
d1=int(d2)
print(d1)

#  + : concatenate two strings
str="true agility is a very valuable thing"
str2=str+", keep going"
print(str2)
```

Result:

```
1.0
10
true agility is a very valuable thing, keep going
```

Arithmetic Operator

Arithmetic operator:
Add +, minus -, multiply *, divide /, divisible //, take modulo %

1.Create a python file : arithmetic.py run in Python Shell

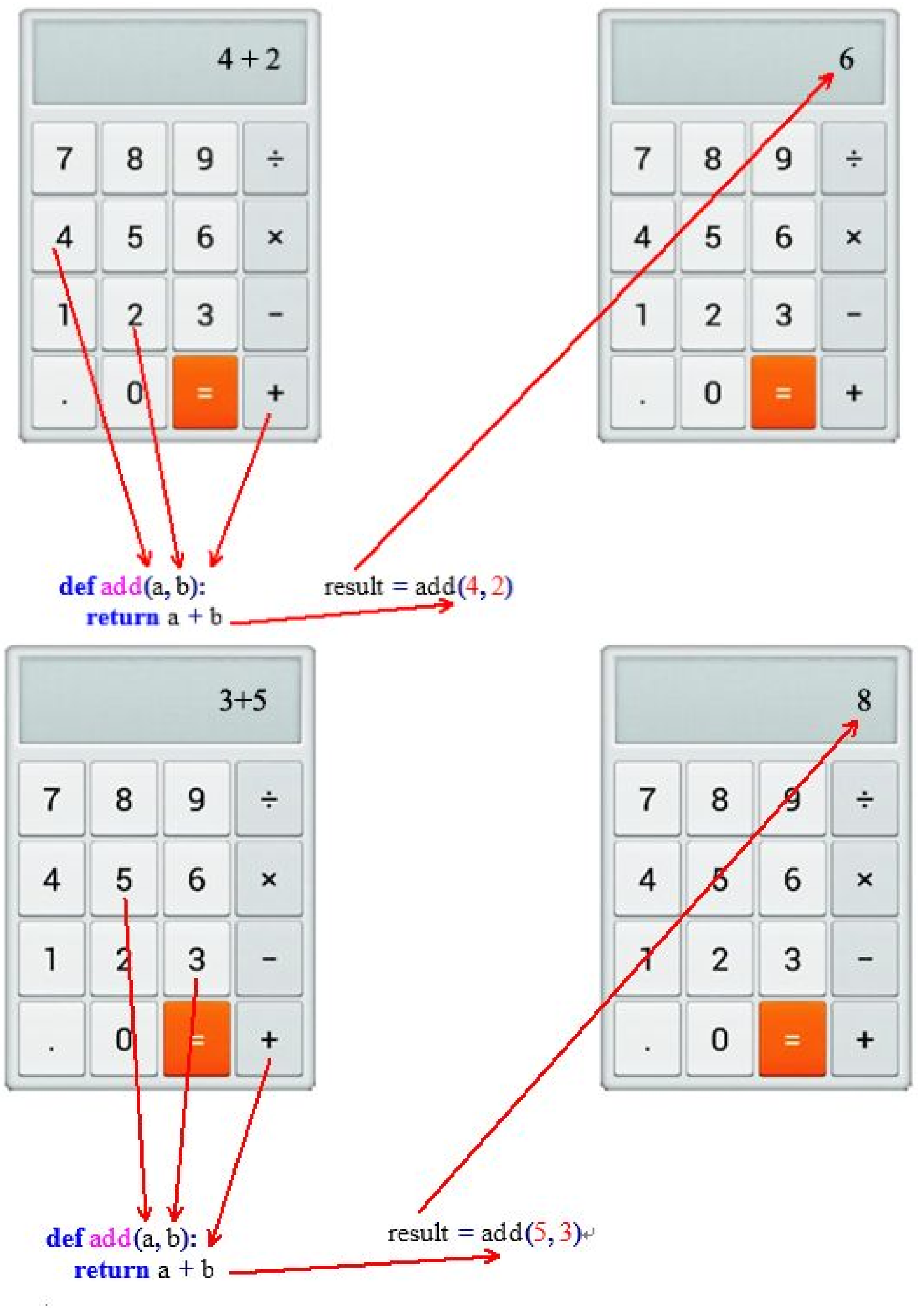
```
a = 1
b = 2
c = 3

print(a + b)
print(a - b)
print(a * b)
print(b / a)      # return the result to a decimal
print(b // a)     # return the result to an integer
print(c % b)   # returns the remainder of dividing
```

Result:

```
3
-1
2
```

2.0
2
1

Function



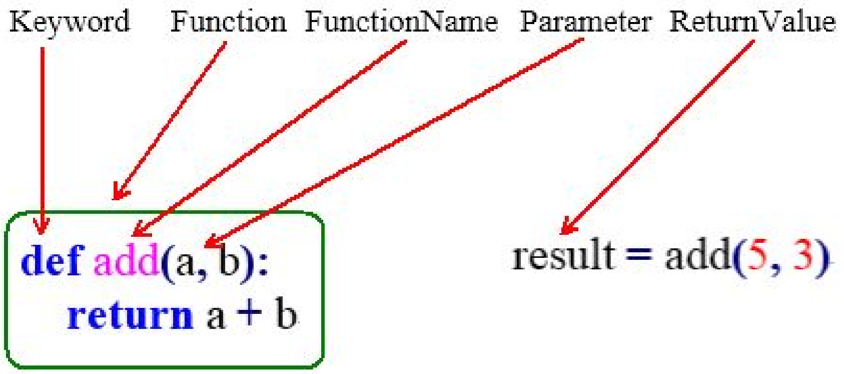
```
def add(a, b):
    return a + b
```

```
result = add(4, 2)
```

```
def add(a, b):
    return a + b
```

```
result = add(5, 3)
```

Function: is a block of code which only runs when it is called.You can pass data, known as parameters, into a function. can return data as a result.



1.Create a python file : Function.py run in Python Shell

```
def add(a, b):
    return a + b

result = add(4, 2)
print(result)

result = add(5, 3)
print(result)
```

Run Result:

```
6
8
```

2.Change python file : Function.py create 3 more functions about - , *, /

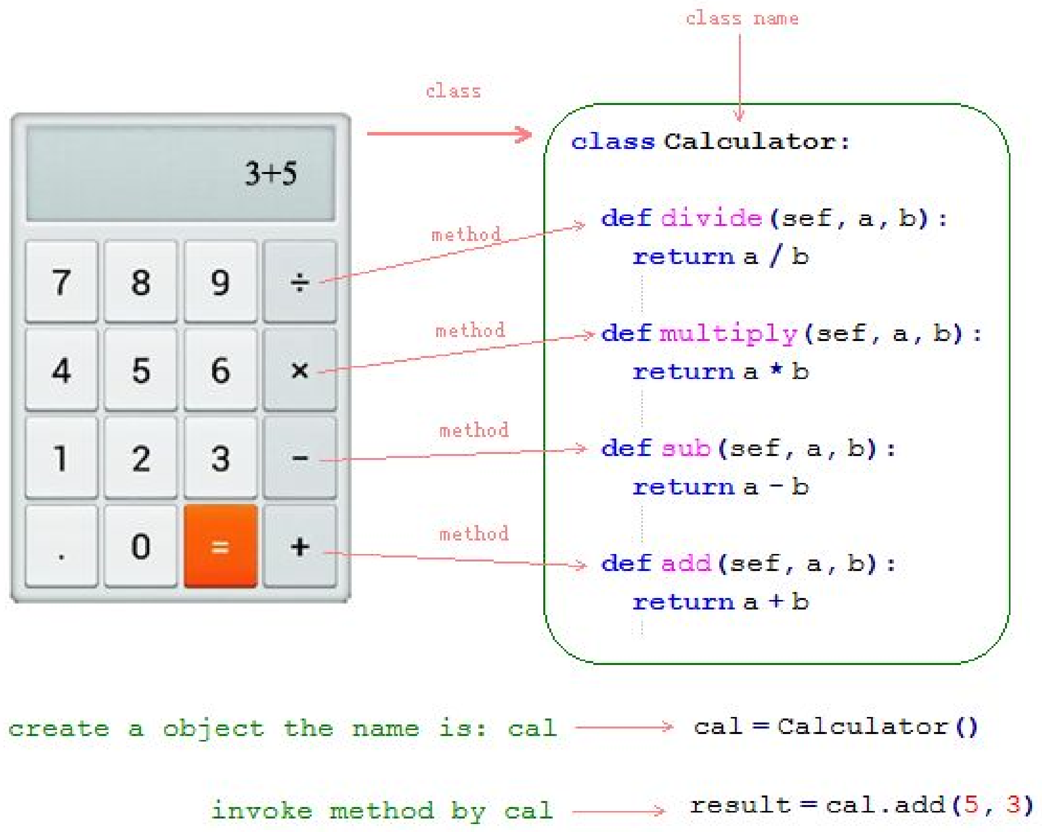
```
def add(a, b):
    return a + b

def sub(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    return a / b

#################test##################

result = add(4, 2)
print(result)

result = sub(4, 2)
print(result)

result = multiply(4, 2)
print(result)

result = divide(4, 2)
print(result)
```

Run Result:

```
6
2
8
2.0
```

Class

Python is an object oriented programming language. Almost everything in Python is an object, with its properties and methods. **Class** is like an object constructor for creating objects.



```
class Calculator:

    def divide(sef, a, b):
      return a / b

    def multiply(sef, a, b):
      return a * b

    def sub(sef, a, b):
      return a - b

    def add(sef, a, b):
      return a + b
```

```
cal = Calculator()

result = cal.add(5, 3)
```

1.Create a file : Calculator.py

self: the reference of current object.

```
class Calculator:

    def divide(self, a, b):
        return a / b

    def multiply(self, a, b):
        return a * b

    def sub(self, a, b):
        return a - b

    def add(self, a, b):
        return a + b

#################test##################

cal = Calculator() # create a object the name is: cal

result = cal.add(4, 2) # invoke method by cal
print(result)

result = cal.sub(4, 2)
print(result)

result = cal.multiply(4, 2)
print(result)

result = cal.divide(4, 2)
print(result)
```

Run Result:

6
2
8
2.0

Relational Operator

1.Create a python file : Relational.py run in Python Shell

Relational operator: only two value: True or False

```
print(100>200)
print(100>=100)
print(100<200)
print(100<=200)
print(100==100)
print(100!=200)
```

Result:

```
False
True
True
True
True
True
```

Assignment operator

1.Create a python file : assignment.py run in Python Shell

```
var=10
var +=1
print(var)

var=10
var -=1
print(var)

var=10
var *=1
print(var)

var=10
var /=1
print(var)
```

Result:

```
11
9
10
10.0
```

Logical Operators

Logical Operators: and, or, not

1. and: return True if both sides of the operation are True, otherwise False
2. or : return False when both sides of the operation are False, otherwise True
3. not: if True return False, otherwise False return True

1.Create a python file : Logic.py run in Python Shell

```
print(True and False)    # return False
print(False and True)    # return False
print(False and False)    # return False
print(True and True)     # return True

print("--------------")

print(True or False)    # return True
print(False or True)    # return True
print(True or True)    # return True
print(False or False)   # return False

print("--------------")

print(not True)    # return False
print(not False)    # return True
```

Result:

```
False
False
False
True
---------------
```

```
True
True
True
False
---------------
False
True
```

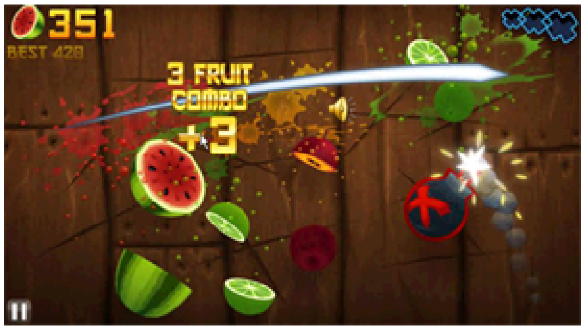
2.Create a python file : Logic2.py run in Python Shell

```
print(1>2 and 3>4)   # return False
print(2>1 and 3>4)   # return False

print("--------------")

print(2>1 or 3>4)   # return True
print(2>1 or 3>4)   # return True
print(1>2 or 3>4)   # return True
```

Result:

```
False
False
--------------
True
True
False
```

If Conditional Statements



Simulation Games:

if num equal 1: watermelon
else if num equal 2: banana
else: thunder

1.Create a python file : If.py run in Python Shell

```
num=1

if num==1:
    print("You cut the watermelon")
elif num==2:
    print("You cut the banana")
else:
    print("You cut to the thunder")
```

Result:
You cut the watermelon

If change num = 2 Run Result:
You cut the banana

If change num = -1 Run Result:
You cut to the thunder

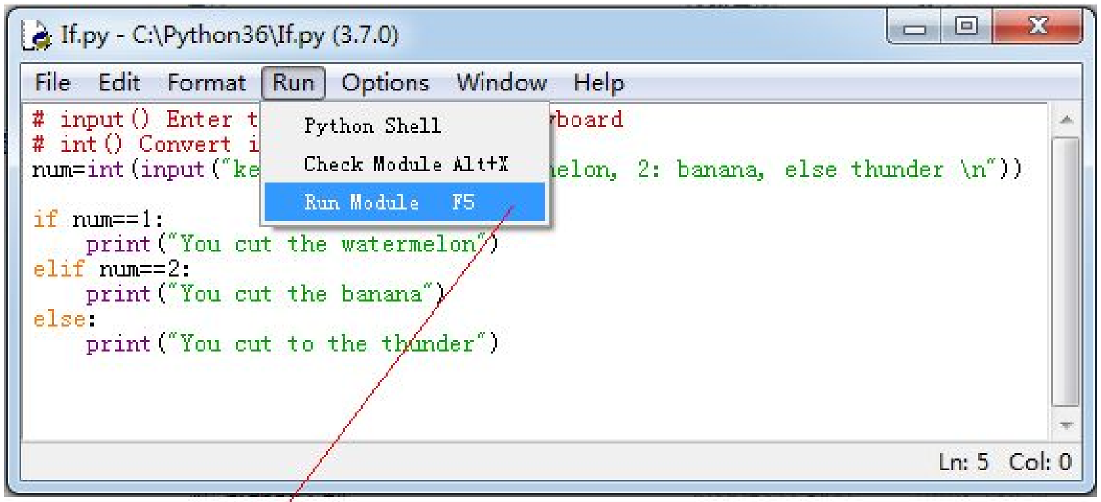
2.Enter the value of num by keyboard

num=int(input("keyboard input 1: watermelon"))
input by keyboard and then store to num

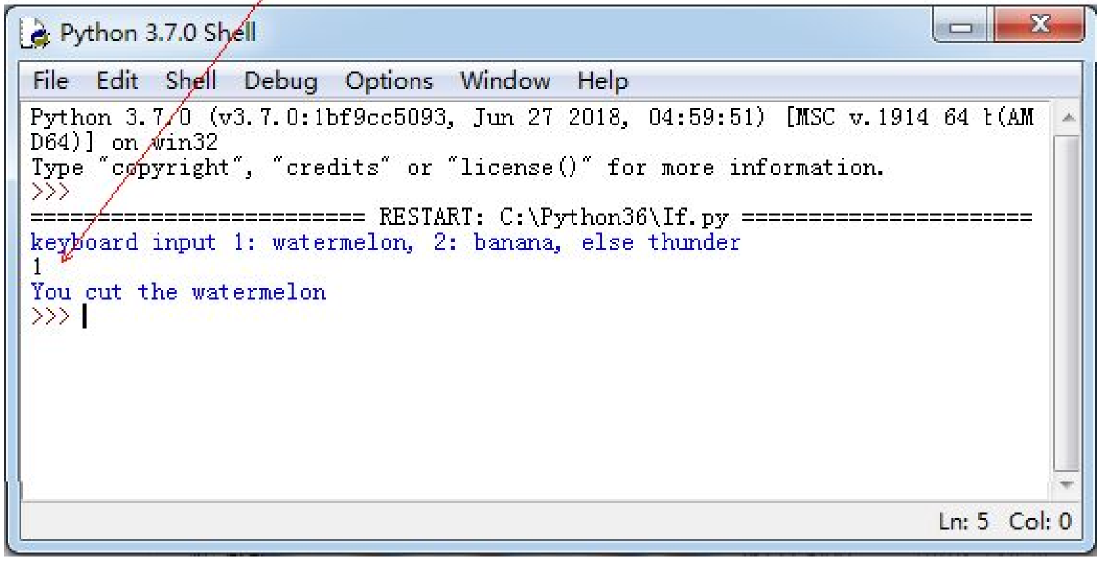
input(): Enter the value of num by keyboard
int(): Convert string to integer

```
num=int(input("keyboard input 1: watermelon, 2: banana, else
thunder \n"))

if num==1:
   print("You cut the watermelon")
elif num==2:
   print("You cut the banana")
else:
   print("You cut to the thunder")
```

If Enter the value of 1 by keyboard

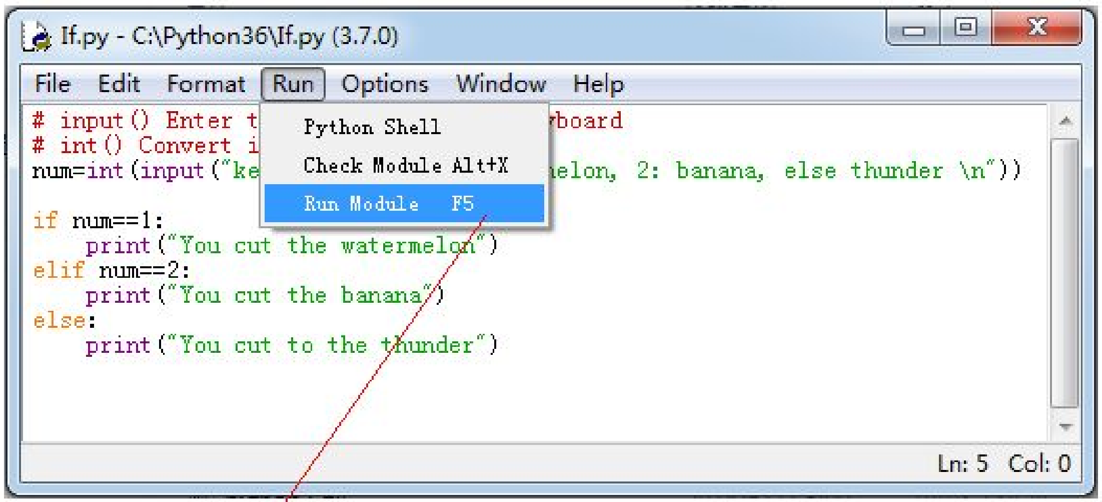

If.py - C:\Python36\If.py (3.7.0)
File Edit Format Run Options Window Help
Python Shell
Check Module Alt+X
Run Module F5
Ln: 5 Col: 0

```
# input() Enter t                      rboard
# int() Convert i
num=int(input("ke                      ielon, 2: banana, else thunder \n"))

if num==1:
    print("You cut the watermelon")
elif num==2:
    print("You cut the banana")
else:
    print("You cut to the thunder")
```

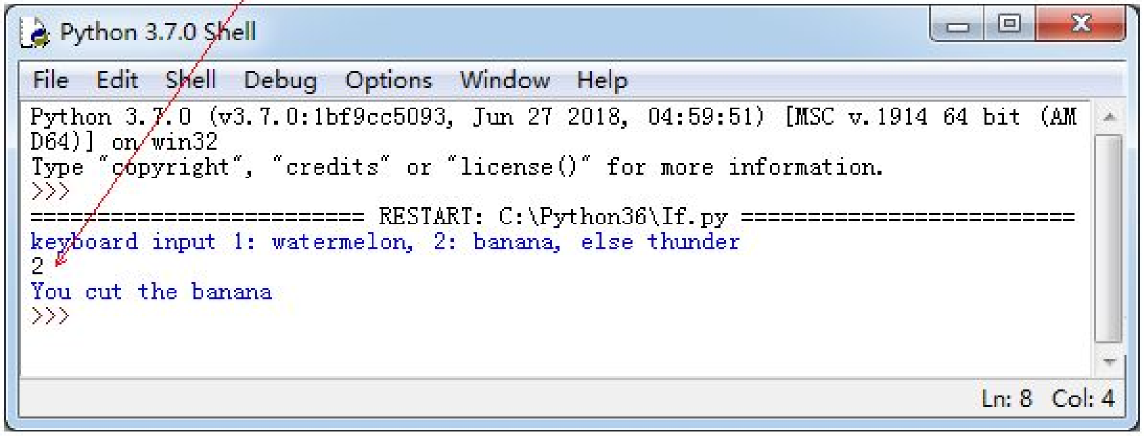


```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit(AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======================== RESTART: C:\Python36\If.py ======================
keyboard input 1: watermelon, 2: banana, else thunder
1
You cut the watermelon
>>> |
```

If Enter the value of 2 by keyboard


If.py - C:\Python36\If.py (3.7.0)
File Edit Format Run Options Window Help
Python Shell
Check Module Alt+X
Run Module F5
Ln: 5 Col: 0

```
# input() Enter t                  board
# int() Convert i
num=int(input("ke                 elon, 2: banana, else thunder \n"))

if num==1:
    print("You cut the watermelon")
elif num==2:
    print("You cut the banana")
else:
    print("You cut to the thunder")
```



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======================== RESTART: C:\Python36\If.py ========================
keyboard input 1: watermelon, 2: banana, else thunder
2
You cut the banana
>>>
```

3.Create a python file : Tax.py run in Python Shell

Payroll tax example:

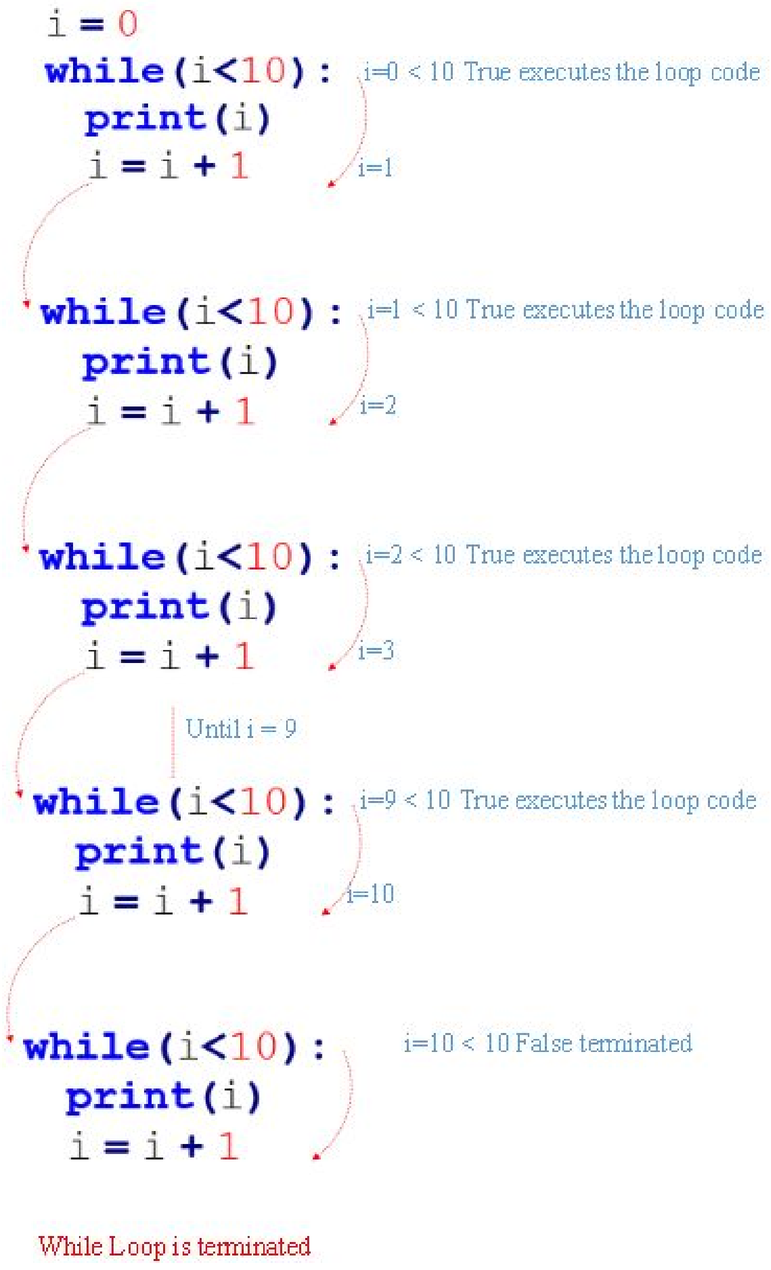
Tax amount = salary* tax rate
level:
500 -- 2000 $: 10% tax rate
2000--5000 $:   tax rate 15%
5000-- 20000 $: tax rate 20%
More than 20000$:   tax rate 30%

```
salary=10000;
tax=0.0;

if salary>=500 and salary<2000:
    tax=salary*0.1
elif salary>=2000 and salary<5000:
    tax=salary*0.15
elif salary>=5000 and salary<20000:
    tax=salary*0.2
else:
    tax=salary*0.3

print("tax amount="+str(tax))   #str() converts float to string
```

Result:

```
tax amount = 2000.0
```

While Loop



```
i = 0
while(i<10):
  print(i)
  i = i + 1

while(i<10):
  print(i)
  i = i + 1

while(i<10):
  print(i)
  i = i + 1

while(i<10):
  print(i)
  i = i + 1

while(i<10):
  print(i)
  i = i + 1
```

i=0 < 10 True executes the loop code
i=1
i=1 < 10 True executes the loop code
i=2
i=2 < 10 True executes the loop code
i=3
Until i = 9
i=9 < 10 True executes the loop code
i=10
i=10 < 10 False terminated
While Loop is terminated

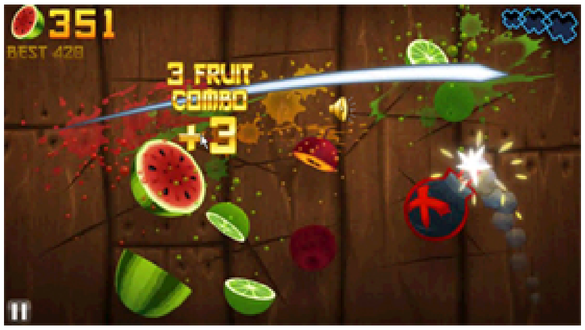
1.Create a python file : WhileLoop.py run in Python Shell

```
i = 0
while(i<10): # if (i<10) True executes the loop, otherwise is
terminated
    print(i)
    i = i + 1
```

Result:

```
0
1
2
3
4
5
6
7
8
9
```

While Loop Fruit Game

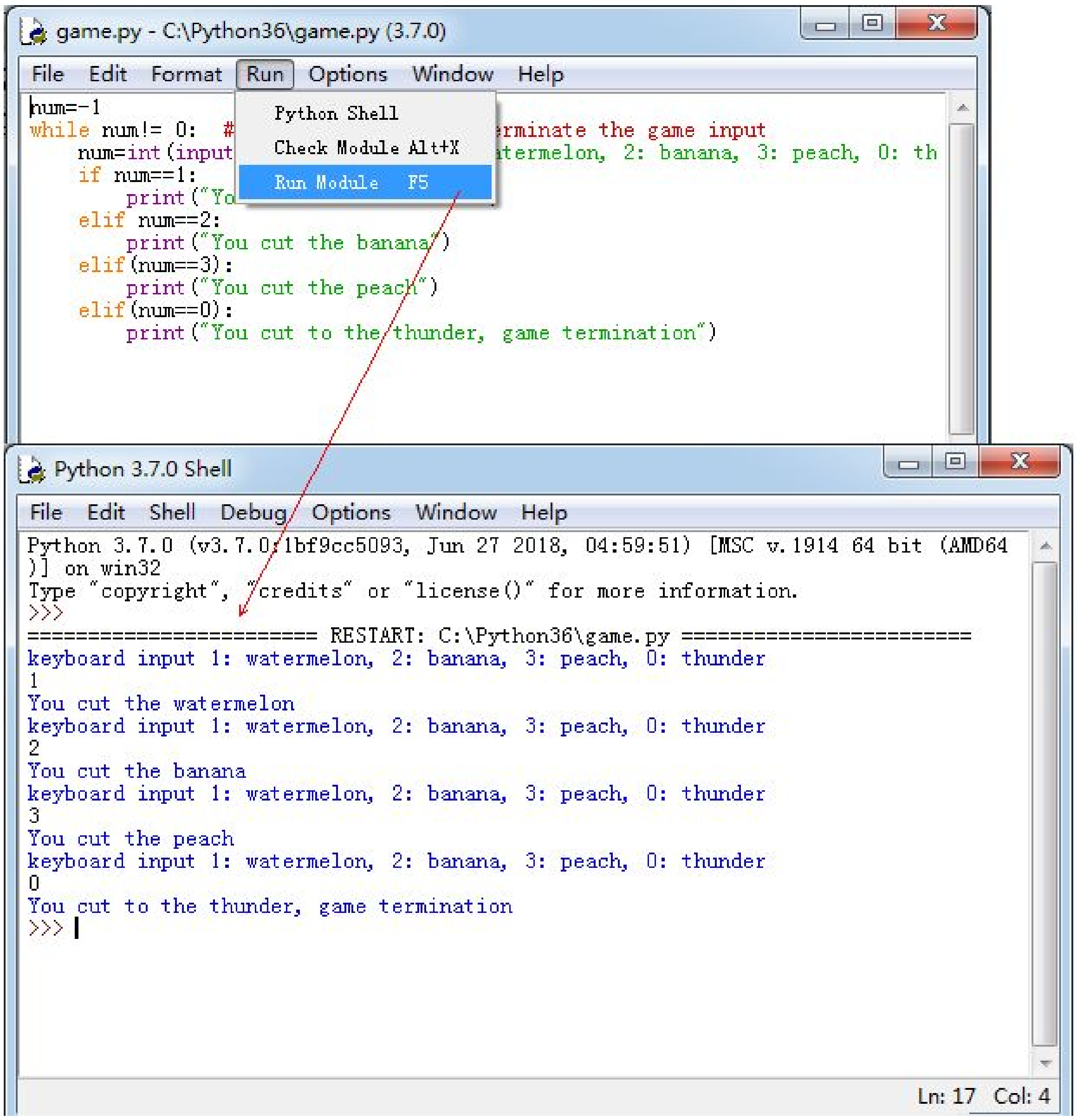


Simulation Games:

```
While num!= 0:
    if num equal 1: watermelon
    else if num equal 2: banana
    else if num equal 3: peach
    else if num equal 0: thunder
```

1.Create a python file : Game.py run in Python Shell

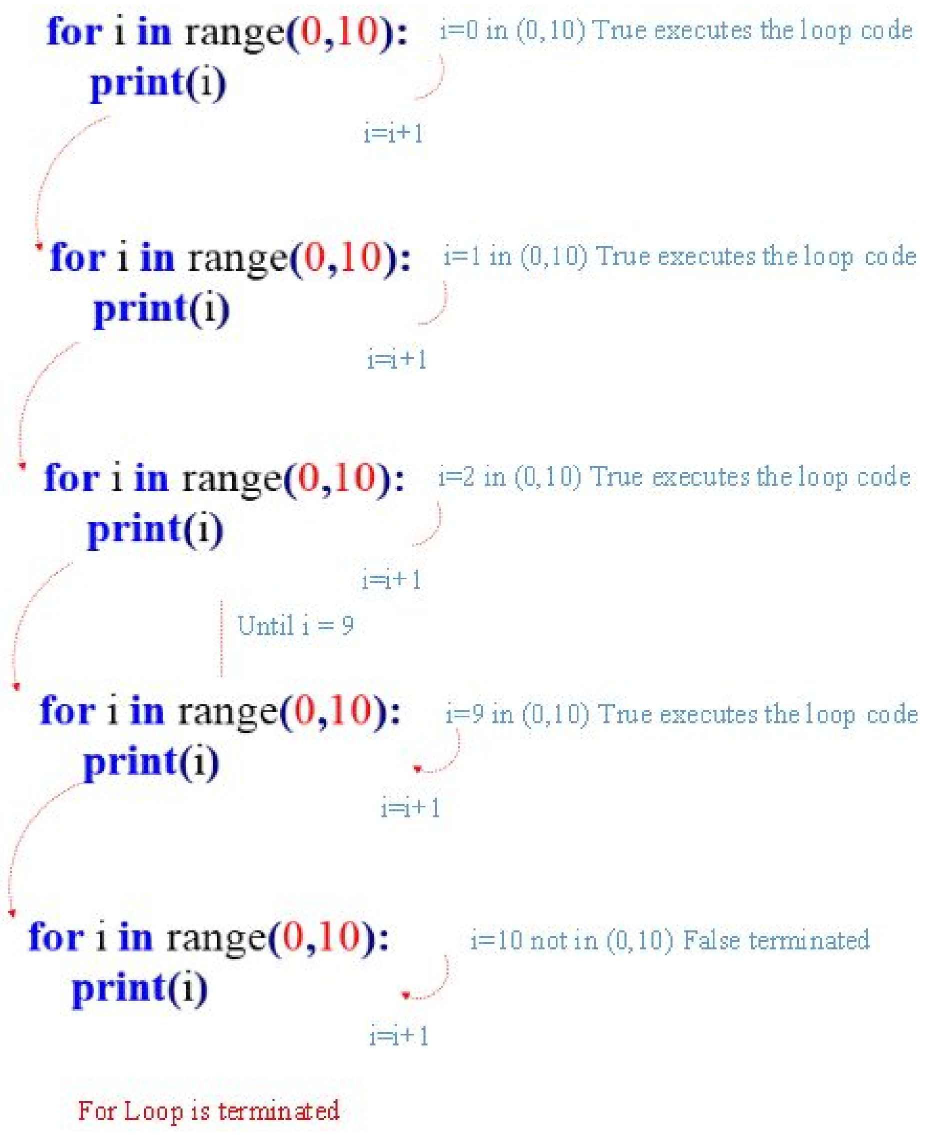
```
num=-1
while num!=0:  # If you enter -1 to terminate the game input
    num=int(input("keyboard input 1: watermelon, 2: banana, 3: 
peach, 0: thunder \n"))
    if num==1:
        print("You cut the watermelon")
    elif num==2:
        print("You cut the banana")
    elif(num==3):
        print("You cut the peach")
    elif(num==0):
        print("You cut to the thunder, game termination")
```


game.py - C:\Python36\game.py (3.7.0)
File Edit Format Run Options Window Help
Python Shell
Check Module Alt+X
Run Module F5
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Ln: 17 Col: 4

```
num=-1
while num!= 0:  #                       rminate the game input
    num=int(input                       atermelon, 2: banana, 3: peach, 0: th
    if num==1:
        print("Yo
    elif num==2:
        print("You cut the banana")
    elif(num==3):
        print("You cut the peach")
    elif(num==0):
        print("You cut to the thunder, game termination")
```

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64
)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======================= RESTART: C:\Python36\game.py =======================
keyboard input 1: watermelon, 2: banana, 3: peach, 0: thunder
1
You cut the watermelon
keyboard input 1: watermelon, 2: banana, 3: peach, 0: thunder
2
You cut the banana
keyboard input 1: watermelon, 2: banana, 3: peach, 0: thunder
3
You cut the peach
keyboard input 1: watermelon, 2: banana, 3: peach, 0: thunder
0
You cut to the thunder, game termination
>>>
```

For Loop



```
for i in range(0,10):
    print(i)
```

i=0 in (0,10) True executes the loop code

i=i+1

```
for i in range(0,10):
    print(i)
```

i=1 in (0,10) True executes the loop code

i=i+1

```
for i in range(0,10):
    print(i)
```

i=2 in (0,10) True executes the loop code

i=i+1

Until i = 9

```
for i in range(0,10):
    print(i)
```

i=9 in (0,10) True executes the loop code

i=i+1

```
for i in range(0,10):
    print(i)
```

i=10 not in (0,10) False terminated

i=i+1

For Loop is terminated

1.Create a python file : ForLoop.py run in Python Shell

```
for i in range(0,10):
    print(i)
```

Result:

```
0
1
2
3
4
5
6
7
8
9
```

For Loop Bubble Ball

Bubble ball game:
the game starts with 64 balls,
requiring 8 balls per line. * is ball

1.Create a python file : ForLoopBall.py run in Python Shell

```
# range(1,65) == [1 -- 65)
for i in range(1,65):
    print("* ", end="") #  end="" : not wrap a new line
    if (i%8)==0: # 8%8==0 , 16%8==0 , 24 %8==0, 32%8==0 ,
48%8==0 , 64%8==0
        print("")   # wrap a new line
```

Result:

```
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
```

Continue and break

1.Create a python file : ContinueBreak.py run in Python Shell

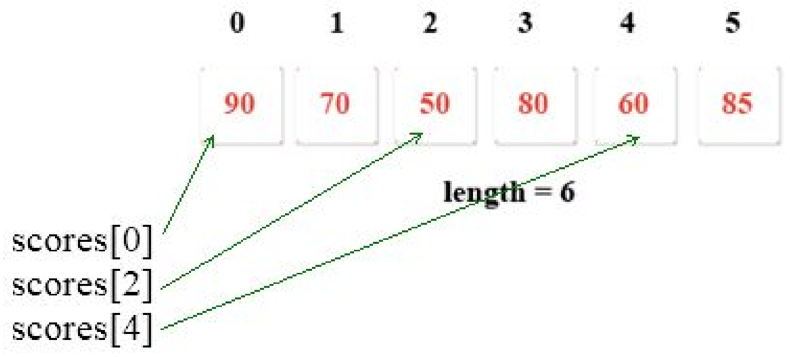
```
for i in range(1,65):
    if i == 13:
        continue    # current loop is not executed, continue to
execute the loop

    if i == 20:
        break   #terminates and exit loop, the subsequent loop will
not execute again

    print("*" + str(i)+" ",end="")

    if (i % 8) == 0:
        print("")   # wrap a new line
```

Result:

```
*1 *2 *3 *4 *5 *6 *7 *8
*9 *10 *11 *12 *14 *15 *16
*17 *18 *19
```

List

List: is a collection which is ordered and changeable. Allows duplicate members.



1. Create a python file : List.py run in Python Shell

```
# the first index is 0, the second index is 1, and so on.
scores=[90,70,50,80,60,85] # initial a list

print(scores[0])
print(scores[2])
print(scores[4])
```

Result:

```
90
50
60
```

2. Print all list scores

len(): function returns the number of items

```
scores=[90,70,50,80,60,85]

# Print all list scores
length=len(scores)    #len() Get the count of list
for i in range(0,length):
    print(scores[i], ",", end="")
```

Result:

```
90 ,70 ,50 ,80 ,60 ,85 ,
```

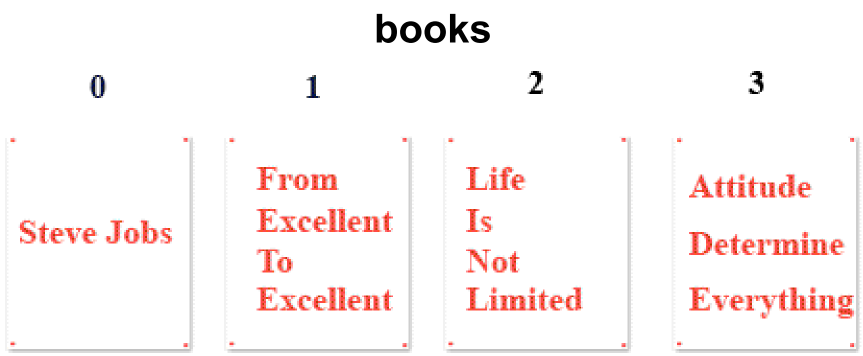
3. Print all scores by anther way

```
scores=[90,70,50,80,60,85]

# print all list scores
for score in scores:
    print(score, ",", end="")
```

Result:

```
90 ,70 ,50 ,80 ,60 ,85 ,
```

4. Add, delete, update List of books

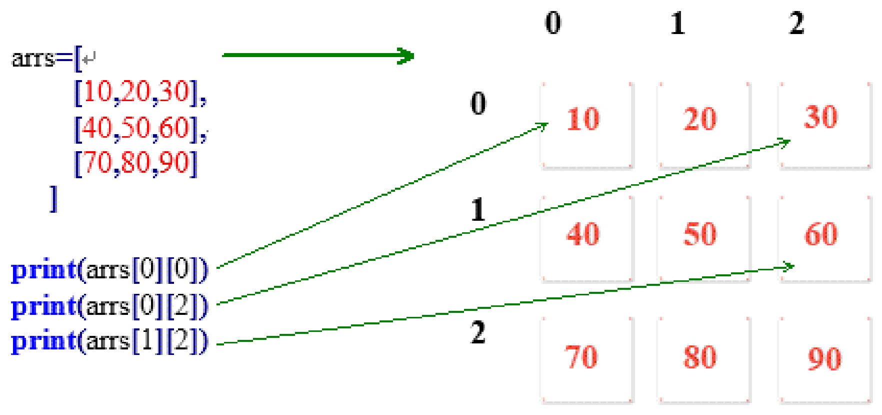


```
#### Create a bookshelf List that stores books ####
books=[]

#### Append data to the list books ####
books.append("Steve Jobs")
books.append("From excellent to excellent")
books.append("Life is not limited")
books.append("Attitude determines everything")

for book in books:
    print(book)

#### delete book at index = 2 ####
del books[2]

for book in books:
    print(book)

####The book at index=2 is updated to: Self-motivated ####
books[2]="Self-motivated"

for book in books:
    print(book)
```

Result:

```
Steve Jobs
From excellent to excellent
Life is not limited
Attitude determines everything


Steve Jobs
From excellent to excellent
Attitude determines everything


Steve Jobs
From excellent to excellent
Self-motivated
```

Two-Dimensional List



```
# define a two-dimensional list
arrs=[
        [10,20,30],
        [40,50,60],
        [70,80,90]
    ]

print(arrs[0][0])
print(arrs[0][2])
print(arrs[1][2])
```

Result:

```
10
30
60
```

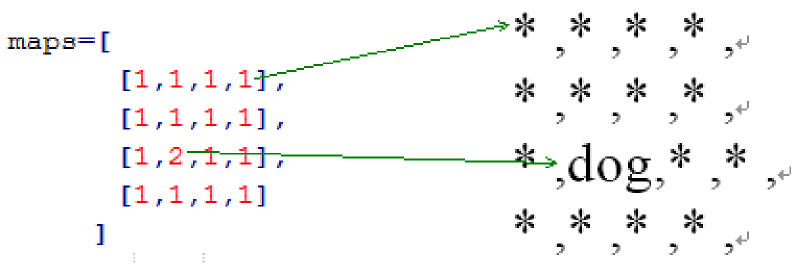
2. Print all value of Two-dimensional list

```
# define a two-dimensional list
arrs=[
        [10,20,30],
        [40,50,60],
        [70,80,90]
     ]

rowIndex=len(arrs)  # count of rows
colIndex=len(arrs[0])   # count of columns

# i is index of row and j is index of column
for i in range(0,rowIndex):
    for j in range(0,colIndex):
        print(arrs[i][j],end=" , ")
    print("")
```

Result:

```
10 , 20 , 30 ,
40 , 50 , 60 ,
70 , 80 , 90 ,
```

Find Dog Game



In two-dimensional array maps

1: asterisk *

2: dog,

please enter the row number and column number to find the dog

```
maps=[
            [1,1,1,1],
            [1,1,1,1],
            [1,2,1,1],
            [1,1,1,1]
       ]

row=int(input("Please enter the row number: \n"))
col=int(input("Please enter column number: \n"))

value= maps[row][col]

rowLength=len(maps)   # the count of rows
colLength=len(maps[0])   # the count of columns

for i in range(0, rowLength):
    for j in range(0, colLength):
        if value== maps [i][j] and value==2:
            print("dog",end=",")
        else:
```

```
        print("* ",end=",")
    print("")
```

Result:

```
Please enter the row number:
1
Please enter column number:
1
* ,* ,* ,* ,
* ,* ,* ,* ,
* ,* ,* ,* ,
* ,* ,* ,* ,
```

Run Again Result:

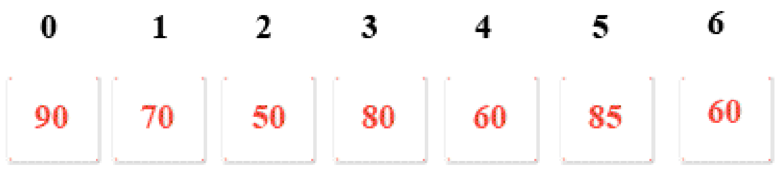
```
Please enter the row number:
2
Please enter column number:
1
* ,* ,* ,* ,
* ,* ,* ,* ,
* ,dog,* ,* ,
* ,* ,* ,* ,
```

Tuple

Python's tuple are similar to list

Tuple elements cannot be modified
Tuples use parentheses ()

scores



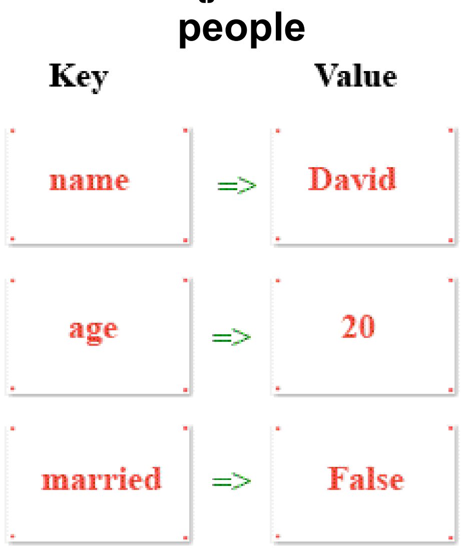
```
scores=(90,70,50,80,60,85,60)
books=("Wonderful Journey", "The Meaning of Life")

# Get tuple element by index
print(scores[1])

# Number of tuple elements
scoresLen=len(scores)
print("Number of tuple elements : "+str(scoresLen))

# Maximum tuple element
scoresMax=max(scores)
print("Maximum tuple element : "+str(scoresMax))

# minimum element
scoresMin=min(scores)
print("minimum element : "+str(scoresMin))
```

Result:

```
70
Number of tuple elements : 7
Maximum tuple element : 90
minimum element : 50
```

Dictionary

1. Dictionary: is key-value use {}



```
people={"name":"David", "age":20,"married":False}

# Print the value by key
print(people["name"])
print(people["age"])
print(people["married"])

# Modify the elements in the dictionary
people["age"]=25
people["married"]=True
print(people)

# Delete element
del people["age"]
print(people)
```

Result:

```
David
20
False
{'name': 'David', 'age': 25, 'married': True}
```

```
{'name': 'David', 'married': True}
```

2. Dictionary's functions

```
book={
            "title": "Daddy behind daughter",
            "price": 18,
            "author": "Wright",
            "publishDate": "2004-01-01"
            }

# return the value of key, without returning the default value
print(book.get("title"))
print(book.get("ISBN","No value"))

print("--------------------------")

#(key,value) traverse tuple
for key,value in book.items():
    print(key, " : ", value)

print("--------------------------")

# Delete according to the key, the return value is the deleted value
returnObj=book.pop("publishDate")
print(returnObj)
print(book)
```

Result:

```
Daddy behind daughter
No value
--------------------------
title  :  Daddy behind daughter
price  :  18
author  :  Wright
publishDate  :  2004-01-01
```

```
--------------------------
2004-01-01
{'title': 'Daddy behind daughter', 'price': 18, 'author': 'Wright'}
```

Set

Set : unordered non-repeating elements

film
0
1
2
The Matrix
Beautiful Mind
Forrest Gump

```
film={"The Matrix", "Beautiful Mind", "Forrest Gump"}

# Add a element to set
film.add("firefighte")
print(film)

# Modify element
film.update({"Love Communication"})
print(film)

# Delete element
film.remove("Forrest Gump")
print(film)

# Whether the element exists
exist="marriage on the rock" in film
print(exist)

# Empty set
film.clear()
print(film)
```

Result:

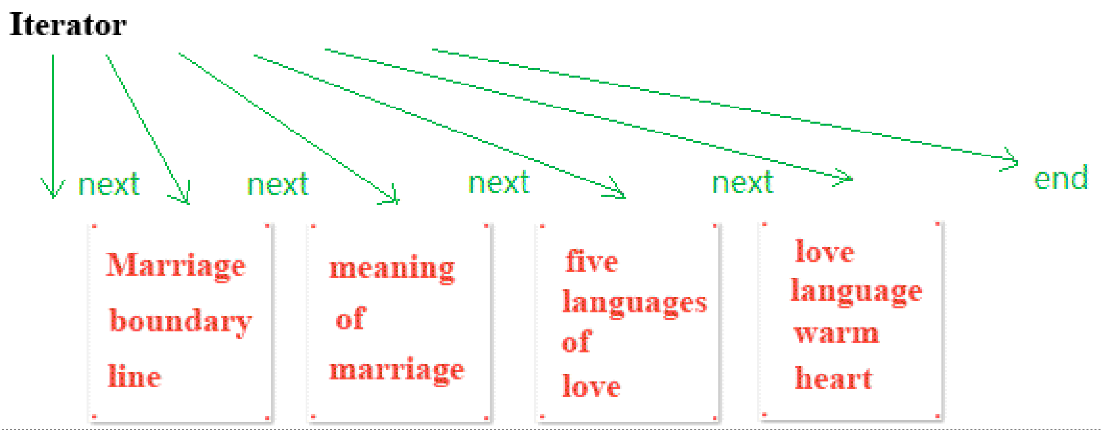
```
{'Beautiful Mind', 'The Matrix', 'Forrest Gump', 'firefighte'}
```

```
{'Beautiful Mind', 'Forrest Gump', 'Love Communication', 'The Matrix',
'firefighte'}
{'Beautiful Mind', 'Love Communication', 'The Matrix', 'firefighte'}
False
set()
```

Iterator

Iterator: is an object that contains a countable number of values.

film



```
books=[
          "Marriage boundary line",
          "meaning of marriage",
          "five languages of love",
          "love language warm heart"
          ]

bookIter = iter(books) # Create iterator object
for book in bookIter:
    print (book, end=" , ")

print("\n---------------------------------------------------")

# Dictionary
books={"title":"five languages of love","author":"Gary Chapman"}

# get books key value iterator object, then output
bookIter = books.keys()
for book in bookIter:
    print (book, " : ",books.get(book))
```

Result:

meaning of marriage , five languages of love , love language warm heart ,
----
title : five languages of love
author : Gary Chapman

Generator

1. The generator is decorated with yield : the result is actually a list

```
def gen():
    x = 0
    y = 2

print(gen())
```

Result:

```
None
```

No return value from the result

2. Modified with the yield generator

```
def gen():
    x = 0
    y = 2

    yield x  # append x value to the generator list
    yield y  # append y value to the generator list

for item in gen():
    print(item)
```

Result:

```
0
2
```

From the results, there is a return value.

All that is appended to the generator yield list.

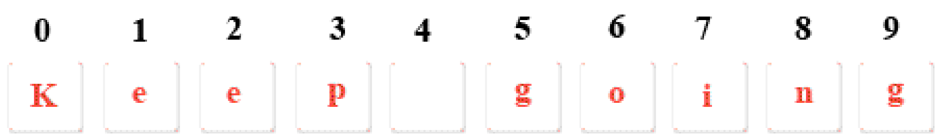
3. Yield generator saves various data

```
def gen():
    x = 0
    y = 2
    married = True
    book = "Heavenly rewards"
    cars=["Benz", "BMW"]
    propery={"user":"Robert","age":90}

    yield x
    yield y
    yield married
    yield book
    yield cars
    yield propery

for item in gen():
    print(item)
```

Result:

```
0
2
True
Heavenly rewards
['Benz', 'BMW']
{'user': 'Robert', 'age': 90}
```

String



1. Basic string operations

```
str="Whatever is worth doing is worth doing well "
str2="Keep going"

# Get a single character
print(str[1])

# Get a substring from 0 to 8
print(str[0:8])

# + string connection
print(str+str2)

# * Repeat output characters
print(str2*2)

# in and not in Whether it exists
print("Keep" in str2)
print("hello" in str2)

# Format string
print ("Name: %s Age: %d" % ("Grace", 38))
```

Result:

```
h
Whatever
Whatever is worth doing is worth doing well Keep going
```

```
Keep goingKeep going
True
False
Name: Grace Age: 38
```

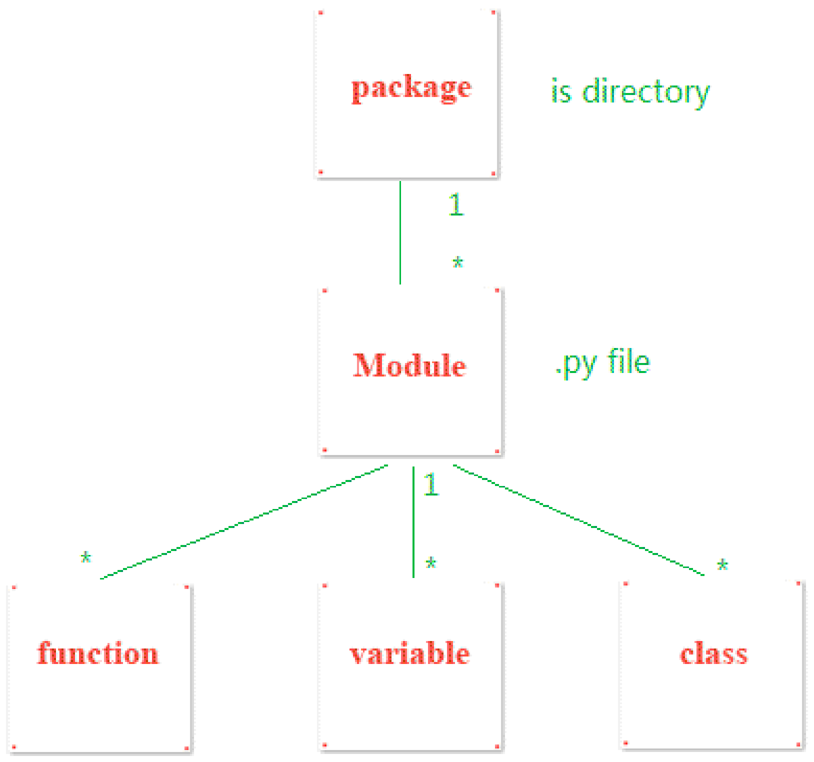
2. String function

```
print("------------------------------------------------")

# string function
# The first character of the string is converted to uppercase
print("nice day".capitalize())

# return the number of occurrences in the string
print("good morning".count("o"))

# Whether the string starts with something
print("c:/pic.jpg".startswith("c:/"))

# Whether the string ends with something
print("c:/pic.jpg".endswith(".jpg"))

# Replace the string in the space for the tab key
print("good job".expandtabs())

# find substring, Return index
print("good afternong".find("afternong"))

# Whether it is a number or letter
print("12345".isalnum())
print("abcd".isalnum())
print("12345a".isalnum())

# Whether they are all letters
print("abcd".isalpha())
print("abcd2".isalpha())

# Whether it is a number
print("1234".isdigit())
print("1234a".isdigit())

# Whether it is all lowercase letters
```

```
print("wonderfull".islower())
print("Wonderfull".islower())

# Whether it is all uppercase letters
print("GREAT".isupper())
print("Great".isupper())


# Whether it is blank
print(" ".isspace())
print(" Great ".isspace())

# Merge into a new string
print(",".join("5678"))
print("".join("awsong"))

# return the string to the left and use the symbol to fill the length.
print("12345".ljust(10,"*"))

# string converted to lowercase
print("HOW ARE YOU".lower())

# string converted to uppercase
print("fine".upper())

# Trop off the space to the left of the string.
print(" thank you ".lstrip())
print("* thank you *".lstrip("*"))

# Replace string
print("thank you".replace("you","too"))

# Split string and then return list
print("father,mother,son".split(","))
```

Modules And Packages

Module: contain custom functions or standard modules or variables .py file.
Modules can be imported by other programs.



1. Custom module

Create a calculate.py file

```
def add(a , b):
    return a + b

def sub(a , b):
    return a - b

def multipy(a , b):
    return a * b

def div(a , b):
    return a / b
```

Call calculate.py's function in another file, test.py

```
# import custom module calculate.py file
import calculate

# call module function
print(calculate.add(10 , 20))

print(calculate.sub(30 , 15))

print(calculate.multipy(10 , 5))

print(calculate.div(100 , 4))
```

Result:

```
30
```

15
50
25.0

2. Import the function from calculate.py

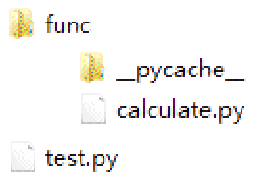
```
# import the add and sub functions from calculate.py
from calculate import add,sub

# call function direct
print(add(10 , 20))

print(sub(30 , 15))
```

Result:

```
30
15
```

3. If the calculate.py and test.py are not in the same directory, that directory is called a package.

the func directory is the package



```
func
    __pycache__
    calculate.py
test.py
```

The function file in the import code directory must have a __init__.py file.

```
from func.calculate import add,sub

# call module function
print(add(10 , 20))

print(sub(30 , 15))
```

Result:

```
30
15
```

Date And Time

Python formats the date and time with the time and calendar modules.

```
import time  # import time module
import calendar   # import calendar module

# Get timestamp : current date - the value of the 1970
print ("current timestamp is:", time.time())

# Get local time
localtime = time.localtime(time.time())
print ("year :", localtime.tm_year)
print ("month :", localtime.tm_mon)
print ("day :", localtime.tm_mday)
print ("hour :", localtime.tm_hour)
print ("minute :", localtime.tm_min)
print ("seconds :", localtime.tm_sec)

#Format time date
print (time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))
print (time.strftime("%Y/%m/%d %H:%M:%S", time.localtime()))
```

Result:

```
current timestamp is: 1554091008.1314328
year : 2019
month : 4
day : 1
hour : 9
minute : 56
seconds : 48
2019-04-01 09:56:48
```

2019/04/01 09:56:48

File and Input and Output I/O

1. Write content to a file

"w": means that the previous content can be overwritten.
"a": means that the content is appended to the end of the file.

```
# open a file
f = open("E:/python/base/study.txt", "w")

# write() Write the content to the file
f.write("a confidence, an effort, a success. \n" )
f.write("It’s not easy to insist on doing simple things" )

# Close open file
f.close()
```

2. Read content from a file

Create any new file: E:/python/base/success.txt input below content:

```
Keep going
If you want to dig a well, you have to dig up the water.
Genius is the ability to work endlessly and diligently.
```

Read E:/python/base/success.txt

```
# open a file
f = open("E:/python/base/study.txt", "w")

# read() read all data from the file
content = f.read()
print (content)

f.close()# Close open file
```

Result:
Keep going
If you want to dig a well, you have to dig up the water.
Genius is the ability to work endlessly and diligently.

3. Read part of the file

```
# open a file
# "r" Open the file as read-only
f = open("E:/python/base/success.txt", "r")

# Read part of the file
content = f.read(4)
print(content)

content = f.read(10)
print(content)

# Close file
f.close()
```

Result:

```
Keep
going
If
```

4. Read a line of a file

```
# open a file
# "r" Open the file as read-only
f = open("E:/python/base/success.txt", "r")

# Read a line of a file
content = f.readline()
print(content)

# Close file
f.close()
```

Result:
Keep going

5. Loop iteration to read files line by line

```
f = open("E:/python/base/success.txt", "r")

# Loop iteration to read files line by line
for line in f:
    print(line, end='')

f.close()
```

Result:

```
Keep going
If you want to dig a well, you have to dig up the water.
Genius is the ability to work endlessly and diligently.
```

6. File and directory copy, delete, rename

```
# import shutil and os standard built-in modules
import shutil,os

file="E:/python/base/file.txt"
if os.path.exists(file): # the file exists
    os.unlink(file)      # elete the file

# Copy file Rename
shutil.copy("E:/python/base/study.txt","E:/python/base/study_copy.txt")

# Copy the entire directory Rename
shutil.copytree("E:/python/base/func","E:/python/base/func_new")

# moving files and folders
shutil.move("E:/python/base/study_copy.txt","E:/python/base/study_move.txt")

f.close()
```

7. Location Positioning

Create file E:/python/base/file.txt input content below:

```
11111aaaaaAAAAA33333BBBBB22222bbbbb
```

Create file locationfile.py

```
# open a file
# "rb" Opens a file in binary format.
f = open("E:/python/base/file.txt", "rb")

# 0 means to move 5 characters from the first character
f.seek(5,0)
content=f.read(5)
print(content)
print(f.tell())  # return the current position

# 1 means to move 5 characters backward from the current
position
f.seek(5,1)
content=f.read(5)
print(content)
print(f.tell())

f.close()
```

Result:

```
b'aaaaa'
10
b'33333'
20
```

Exception Handling

1. Exception: Error detected during Python program runtime

```
def div(a , b):
   # if there is a exception jump to  except block, finally always be executed
   try:
      return a / b
   except (ZeroDivisionError) as Argument:
      print("exception : ",Argument)
   finally:
      print("Always execute, free resources")

print(div(10,2))

div(10,0)
```

Result:

```
Always execute, free resources
5.0
exception :  division by zero
Always execute, free resources
```

Other standard exceptions:

EOFError:	user input file end mark EOF
FloatingPointError:	floating point calculation error
ImportError:	when the import module fails
IndexError:	index is out of range of sequence
MemoryError:	Memory overflow (can be released by deleting objects)
NameError:	tries to access a variable that does not exist
OSError:	OS generated exception

OverflowError:	numeric operation exceeds maximum limit
RuntimeError:	general runtime error
TypeError:	Invalid operation between different types
ValueError:	passed invalid parameters
ZeroDivisionError:	divide by zero

Regular Expression

1. Regular expression: Checks if a string matches a pattern.

```
import re

# Replace spaces and tabs with commas ,
partern = re.compile(r"[ \t]+")
print(partern.sub(",","David    Isacc       Sally    Tim James"))

# students' scores are printed out
partern = re.compile(r"[,;:&]")
print(partern.split("100,69;70,90:85&50"))

# The verification code must be a number and is a 4 digit number
partern = re.compile(r"\\d{4}")
print(partern.match("8766"))
```

Result:

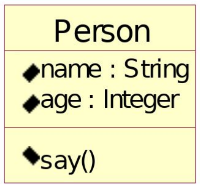
```
David,Isacc,Sally,Tim,James
['100', '69', '70', '90', '85', '50']
None
```

Regular expression common rule match:

^ :	The beginning of the matching string
$:	matches the end of the string.
. :	Match any character.
[...] :	is used to represent a set of characters [abc]
[^...] :	Characters not in []: [^abc] Matches characters other than a, b, c.
re* :	matches zero or more expressions.
re+:	matches one or more expressions.
re? :	matches 0 or 1 expression
re{ n} :	matches n previous expressions.
re{ n,} :	exactly matches n previous expressions.
re{ n,m}:	matches n to m times before the expression
a| b :	matches a or b
\w :	matches the alphanumeric underline
\W :	matches non-alphanumeric underscores
\s :	matches any white space character.
\S :	matches any non-null character
\d :	matches any number, equivalent to [0-9].
\D :	matches any non-number
\A :	matches the string to start
\Z :	Matches the end of the string. If there is a newline, it only matches the end string before the newline.
\z :	matches the end of the string
\b :	matches a word boundary, which is the position between a word and a space.
\B :	matches non-word boundaries.

Create Class

Class: is a template definition of the method s and variable s in a particular kind of object .

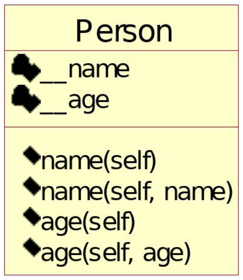


```
# defining a Person class
class Person:
    # Define the basic propertie of the class
    name = ""
    age = 0

    # defining method , the method must contain the parameter self
    def say(self):
        print("My name is:",self.name,",this year ",self.age," years
old")

print("---------------------")

# Instantiate the Person object, return a object reference p of
Person
p = Person()

# Use p access properties and methods
p.name = "David"
p.age = 22
p.say()

print("---------------------")

p2 = Person()
p2.name = "Mathew"
p2.age = 23
p2.say()
```

Result:

```
-----------------------
My name is: David ,this year  22  years old
-----------------------
My name is: Mathew ,this year  23  years old
```

Encapsulation



```
class Person:
    # Define private properties, outside the class can not be directly
access
    __name = ""
    __age = 0

    # Define public methods for accessing private properties
    @property
    def name(self):
        return self.__name
    @name.setter
    def name(self,name):
        self.__name = name

    @property
    def age(self):
        return self.__age
    @age.setter
    def age(self,age):
        self.__age = age

    def say(self):
        print("My name is:", self.__name, ",this year", self.__age,
"years old")
################ test ################
p = Person() # Create a Person Object named p
p.name = "David"
p.age = 22
```

```
p.say() # Invoke method: say by p
print(p.name , " " , p.age)
```

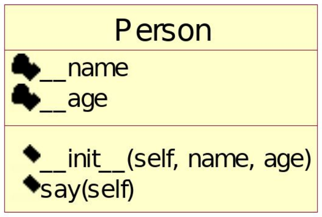
Result:

```
My name is: David ,this year 22 years old
David   22
```

Constructor Method

Constructor method name must be __init__

When the object is created, the constructor is automatically called



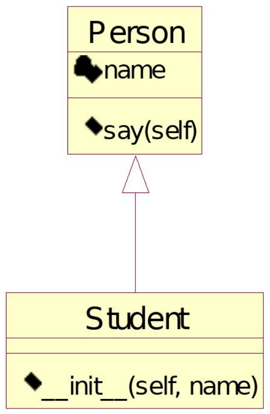
```
class Person:
    __name = ""
    __age = 0

    # Constructor method
    def __init__(self,name,age):
        self.__name=name
        self.__age=age

    def say(self):
        print("My name is :",self.__name,",this year:",self.__age,"
years old")


# Create Person Object by constructor with parameters
p=Person("David",22)
p.say();

#  Anonymous object
Person("James",23).say()
```

Result:

```
My name is : David ,this year: 22  years old
My name is : James ,this year: 23  years old
```

Inheritance

Inheritance: allows us to define a class that inherits all the methods and properties from another class.



Topic:

Students inherit the characteristics of people

Step:

1. class: Student, Person
2. Relationship: Student(Person)
3. Attribute: the person's characteristics (name)
4. Method: People's actions (say)

```
# Person is parent class
class Person:
  name = ""

  def say(self):
    print(self.name,"Speaking")

# Student is subclass of Person
class Student(Person):

  def __init__(self,name):
    self.name = name
    print("Subclass Student Instantiation")
```

```
# Instanceization Subclass Student
s=Student("David")
s.say()

'''
  1. The subclass cannot access the private property of the parent
class
  2. Python supports inherit multiple parent classes
'''
```

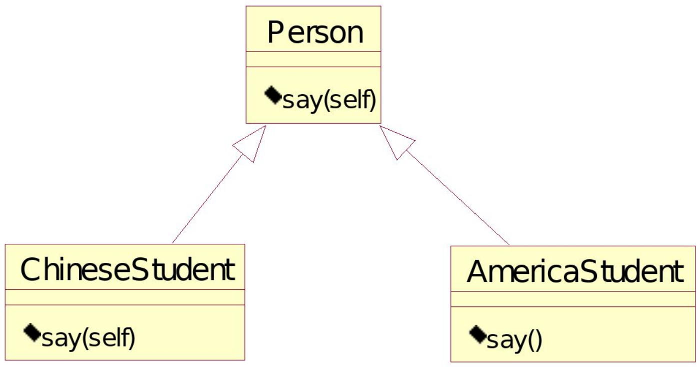
Result:

```
Subclass Student Instantiation
David Speaking
```

From the results, the subclass Student inherit the property name and method say() of the Person

Override and Polymorphism

Override: the subclass overwrites the method of the parent class, the instance of the subclass will call the method of subclass, not call the method of the parent class.

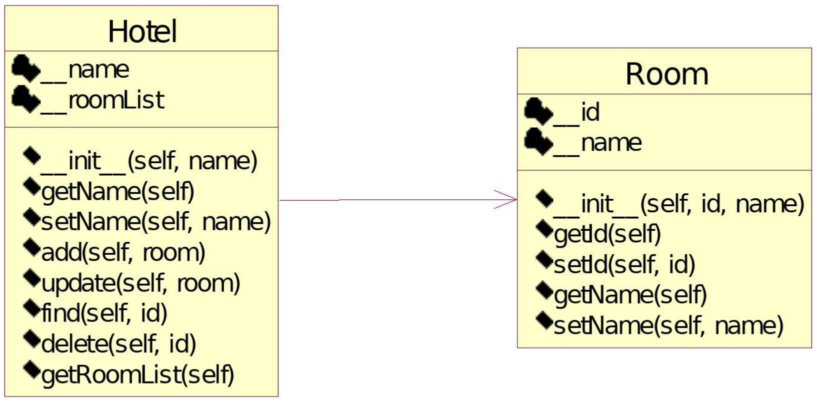


```
class Person:
    def say(self):
        print("Speaking")

# ChineseStudent Inherit Person
class ChineseStudent(Person):
    def say(self):
        print("Speaking Chinese")

#AmericaStudent Inheritance Person
class AmericaStudent(Person):
    def say(self):
        print("Speaking English")

#############  test ##############
s = ChineseStudent()
s.say();

s2 = AmericaStudent()
s2.say();
```

Result:

```
Speaking Chinese
Speaking English
```

List and Class



There are various types of rooms in the hotel
Step:

1. Class: Hotel, Room
2. Relationship: Hotel contains rooms (Hotel 1 to Multi Room)
3. Attribute: Hotel (name), Room (id, name)
4. Method: Hotel (add, update, delete, find)

```
class Room:
    __id = ""
    __name = ""

    def __init__(self , id , name):
        self.__id = id
        self.__name = name

    def getId(self):
        return self.__id

    def setId(self,id):
        self.__id = id

    def getName(self):
        return self.__name
```

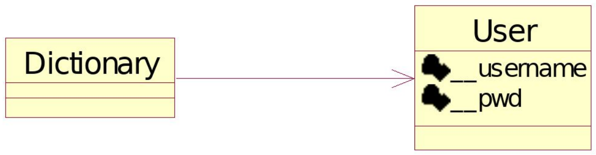
```
 def setName(self,name):
       self.__name = name


class Hotel:
   __name = "" # Hotel name
   __roomList = [] # store rooms in List

   def __init__(self , name):
      self.__name = name

   def getName(self):
      return self.__name

   def setName(self,name):
      self.__name = name

   # Add Room
   def add(self,room):
      self.__roomList.append(room)

   # Modify the room according to the room id
   def update(self,room):
      length=len(self.__roomList)
      for i in range(0,length):
         if self.__roomList[i].getId() == room.getId():
            self.__roomList[i] = room
            break;

   #Find a room based on the room id
   def find(self,id):
      length=len(self.__roomList)
      for i in range(0,length):
         if self.__roomList[i].getId() == id:
            return self.__roomList[i]
```

```
    #According to the room id, delete the room.
    def delete(self,id):
        length=len(self.__roomList)
        for i in range(0,length):
            if self.__roomList[i].getId() == id:
                del self.__roomList[i]
                break;




    # Get a list of rooms
    def getRoomList(self):
        return self.__roomList

#-----------------------------------------------
h = Hotel("Express Inn Hotel")

# Add room
h.add(Room("001","Advanced King Room"))
h.add(Room("002","Twin Room"))
h.add(Room("003","Business Room"))

#Show hotel information
print("Hotel Information ： ")
print(h.getName())

print("----------------------")
print("Hotel Room Information")
for room in h.getRoomList():
    print(room.getId()," : ",room.getName())

print("----------------------")
# Modify room id 001 for standard room
h.update(Room("001","standard room"))
print("modified hotel room information")
for room in h.getRoomList():
    print(room.getId()," : ",room.getName())

print("----------------------")
# Find the room id 002
room=h.find("002")
print("002 hotel room information")
print(room.getId()," : ",room.getName())

print("----------------------")
# Delete room with room id 003
h.delete("003")
print("Deleted Hotel Room Information")
for room in h.getRoomList():
    print(room.getId()," : ",room.getName())
```

Result:

```
Hotel Information ：
Express Inn Hotel
-----------------------
Hotel Room Information
001  :  Advanced King Room
002  :  Twin Room
003  :  Business Room
-----------------------
modified hotel room information
001  :  standard room
002  :  Twin Room
003  :  Business Room
-----------------------
002 hotel room information
002  :  Twin Room
-----------------------
Deleted Hotel Room Information
001  :  standard room
002  :  Twin Room
```

Dictionary and Class



```
userDictionary = {} # dictionary store users

class User:
    __username = ""
    __pwd = ""

    def __init__(self , username , pwd):
        self.__username = username
        self.__pwd = pwd

    def getUsername(self):
        return self.__username

    def setUsername(self,username):
        self.__username = username

    def getPwd(self):
        return self.__pwd

    def setPwd(self,pwd):
        self.__pwd = pwd

# Add User
userDictionary["david"] = User("David","111111")
userDictionary["james"] = User("James","222222")
userDictionary["john"] = User("John","333333")

print("print user information")
```

```python
for key,value in userDictionary.items():
    print(key, " : ", value.getUsername()," ",value.getPwd())





print("----------------------")

# Change david password is 444444
davidUser=userDictionary["david"]
davidUser.setPwd("444444")

print("After modified user information")
for key,value in userDictionary.items():
    print(key, " : ", value.getUsername()," ",value.getPwd())

print("----------------------")

# Delete user james
del userDictionary["james"]

print("After delete User Information")
for key,value in userDictionary.items():
    print(key, " : ", value.getUsername()," ",value.getPwd())
```

Result:

```
print user information
david  :  David   111111
james  :  James   222222
john  :  John   333333
----------------------
After modified user information
david  :  David   444444
james  :  James   222222
```

```
john  :  John   333333
-----------------------
After delete User Information
david  :  David   444444
john  :  John   333333
```

Multithreading

Thread: is a single sequential flow of control within a program.
the CPU allocates to each thread for a period of time to execute.

1. Create Thread

```
import threading
import time

# MyThread Inherit threading.Thread
class MyThread(threading.Thread):

    def __init__(self, name):
        threading.Thread.__init__(self)
        self.name = name # thread name

    def run(self):
        for i in range(0,5):
            print(threading.currentThread().getName() , i)
            time.sleep(1) # sleep 1 second
################ test ####################
try:
    # create two threads , running the car and the train at the same
time:
    thread1 = MyThread("Car thread")
    thread2 = MyThread("Train thread")

    # start thread
    thread1.start()
    thread2.start()
except:
   print ("Error: Unable to start thread")
```

Result:

```
Car thread 0
Train thread 0
Car thread 1
```

```
Train thread 1
Train thread 2
Car thread 2
Train thread 3
Car thread 3
Train thread 4
Car thread 4
```

From the results, two threads are alternately executed.

2. Thread synchronization

Synchronization: when one thread executing, anther thread need waiting

Multiple threads work together to modify a shared data, and may have an incorrect result. threading.Lock can implement thread synchronization.

```
import threading
import time
class MyThread(threading.Thread):
    def __init__(self, name):
        threading.Thread.__init__(self)
        self.name = name

    def run(self):
        threadLock.acquire()# Get the lock, other threads must wait

        for i in range(0,5):
            print(threading.currentThread().getName() , i)
            time.sleep(1)

        threadLock.release()# release lock, other threads can access
############### test ####################
threadLock = threading.Lock()# Create thread lock
try:
    thread1 = MyThread("Car thread")
    thread2 = MyThread("Train thread")

    # start thread
    thread1.start()
    thread2.start()
except:
   print ("Error: Unable to start thread")
```

Result:

```
Car thread 0
Car thread 1
Car thread 2
Car thread 3
Car thread 4
Train thread 0
Train thread 1
Train thread 2
Train thread 3
Train thread 4
```

Python PyMySQL MySQL

1. install MySQL

http://en.verejava.com/?id=2524976079781

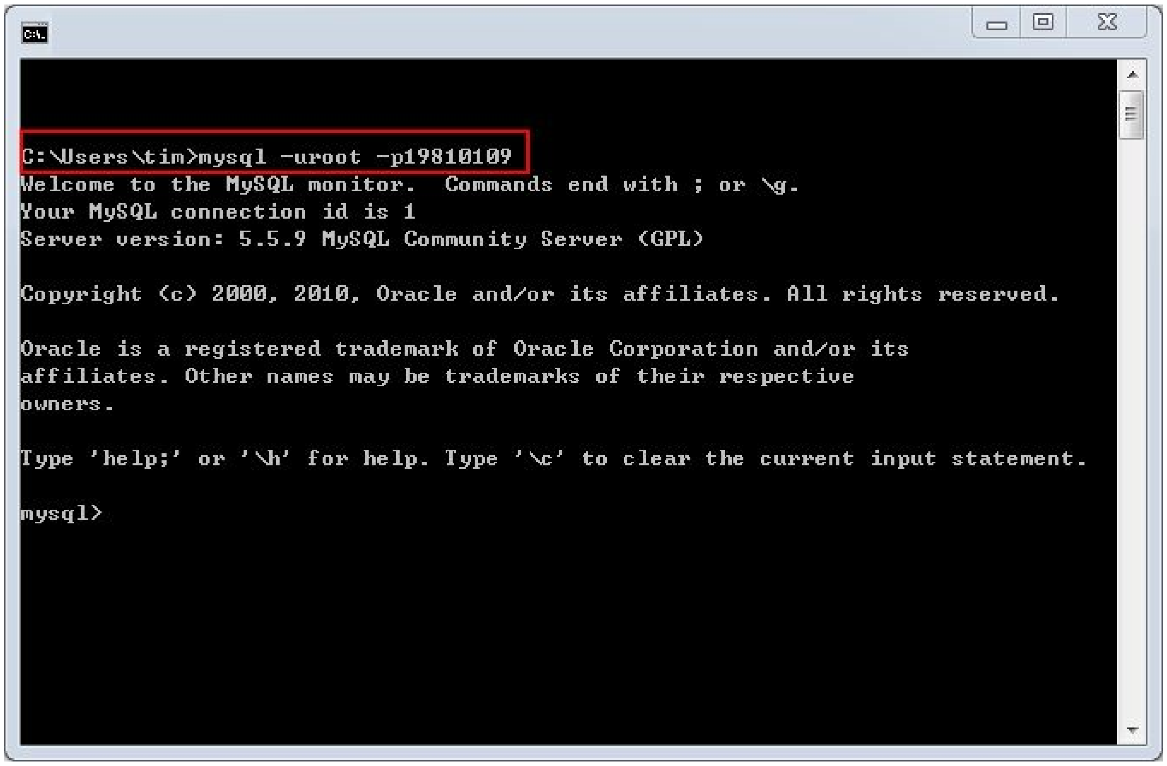
2. install PyMySQL

http://en.verejava.com/?id=19714057591735

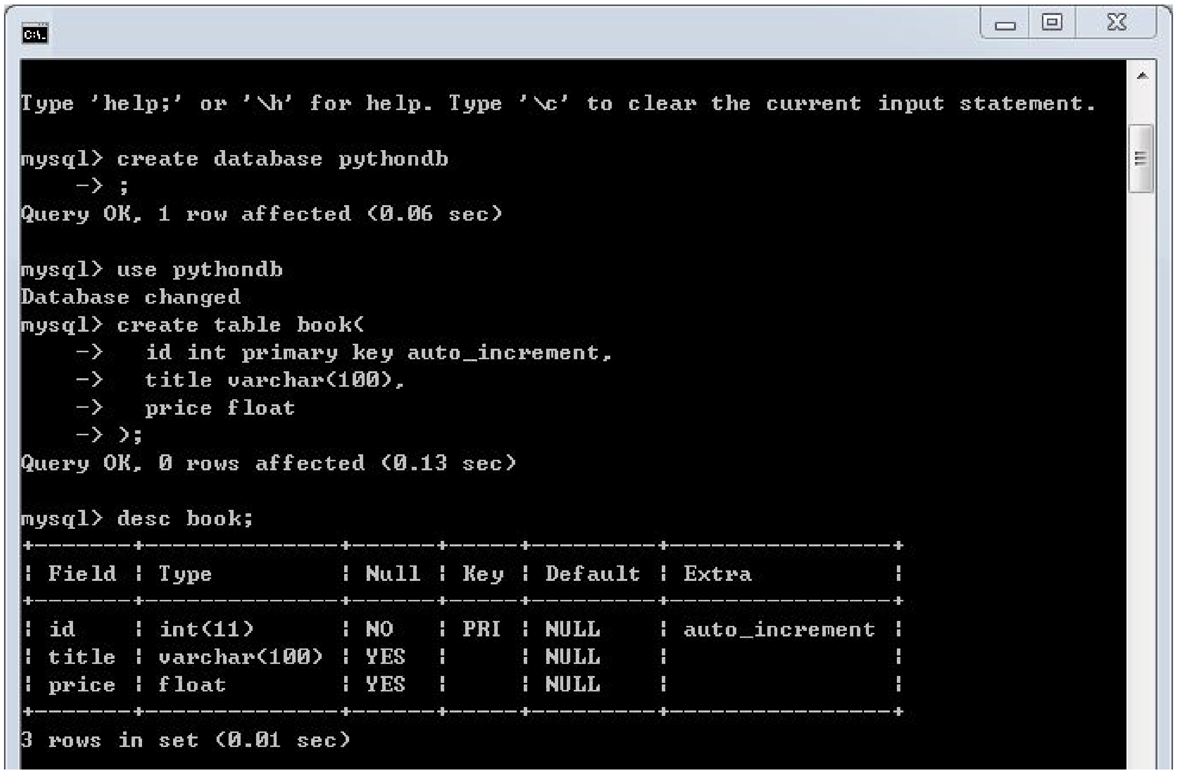
3. Python MySQL database add delete modify query

Open the command line Login to MySQL

```
C:\Users\tim>mysql -uroot -p19810109
```



```
C:\Users\tim>mysql -uroot -p19810109
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.5.9 MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql>create database pythondb;

mysql>use pythondb

create table book(
  id int primary key auto_increment,
  title varchar(100),
  price float
);
```



```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database pythondb
    -> ;
Query OK, 1 row affected (0.06 sec)

mysql> use pythondb
Database changed
mysql> create table book(
    ->   id int primary key auto_increment,
    ->   title varchar(100),
    ->   price float
    -> );
Query OK, 0 rows affected (0.13 sec)

mysql> desc book;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int(11)      | NO   | PRI | NULL    | auto_increment |
| title | varchar(100) | YES  |     | NULL    |                |
| price | float        | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

1. Insert data to book table

```
import pymysql  # import database module

#  Open database connection 4 parameters are:
#  database server IP, username, password, database name, character
set
db =
pymysql.connect("localhost","root","19810109","pythondb",charset="utf8")

cursor = db.cursor()# Get the database cursor

try:
   # Insert 2 rows
   sql = "insert into book(title,price)values('the power of positive
thinking',100)"
   cursor.execute(sql) # Execute sql statement
   sql = "insert into book(title,price)values('Thinking to get rich',200)"
   cursor.execute(sql)

   sql = "select * from book"
   cursor.execute(sql)

   results = cursor.fetchall()# Get data for all book tables
   for row in results:
      id = row[0]
      title = row[1]
      price = row[2]
      print ("id=",id,"title=",title," ","price=",price)

   # Submit transaction to database to execute
   db.commit()
except exception:
   db.rollback()# If the error occurs, the transaction is rolled back
   print("fail : ",exception)
finally:
   db.close()  # close the database connection
```

Result:

```
id= 11 title= the power of positive thinking   price= 100.0
id= 12 title= Thinking to get rich   price= 200.0
```

2. Modify book table data

```
import pymysql

db =
pymysql.connect("localhost","root","19810109","pythondb",charset="utf8")

cursor = db.cursor()

try:
    # Modify price based on id
    sql = "update book set price=80 where id=12"
    cursor.execute(sql)

    sql = "select * from book"
    cursor.execute(sql)

    results = cursor.fetchall()
    for row in results:
        id = row[0]
        title = row[1]
        price = row[2]
        print ("id=",id,"title=",title," ","price=",price)

    db.commit()
except exception:
    db.rollback()
    print("fail : ",exception)
finally:
    db.close()
```

Result:

```
id= 11 title= the power of positive thinking   price= 100.0
id= 12 title= Thinking to get rich   price= 80.0
```

3. Delete book table data

```
import pymysql

db =
pymysql.connect("localhost","root","19810109","pythondb",charset="utf8")

cursor = db.cursor()

try:
    # Delete based on id
    sql = "delete from book where id=12"
    cursor.execute(sql)

    sql = "select * from book"
    cursor.execute(sql)

    results = cursor.fetchall()
    for row in results:
        id = row[0]
        title = row[1]
        price = row[2]
        print ("id=",id,"title=",title," ","price=",price)

    db.commit()
except exception:
    db.rollback()
    print("fail : ",exception)
finally:
    db.close()
```

Result:

```
id= 11 title= the power of positive thinking   price= 100.0
```