coggle
made for free at coggle.it

PYTHON

File I/O

one file operation

Step 1: open file

```
file_variable=
open("fileName[c:\zz\zz.txt]",
"r/w/x/a/b/t...",encoding="utf-8")
```

Notice:
1. create/read/write/append mode chỉ có một
2. nếu file không tồn tại, tự động tạo ms

- 'x': create a new file and open it for writing
- 'r': open for reading (default) — in r mode, will not create a new file if the file name is not exist.
- 'a' mở ghi, ghi nối đuôi
- 'w': mở ghi, xóa hết cái cũ
- 't': text mode (default)
- 'b': binary (nhị phân) mode
- 'U': universal newline mode (deprecated)
- ex: open("","a+") có thể đọc và nối đồng thời

'+' open a disk file for updating (reading and writing)

- close file.

```
file_variable.close()
```

cú pháp ghi vào tệp
- file1.***write("contents")*** ***Notice:***rewrite the whole file
- a mode file1.***write***("append contents") — appending to a file

Step 2: read file

file1.***read***(number]): đọc tất cả hoặc số được chỉ định

Notice: lần read() đầu tiên sẽ di chuyển con trỏ đến cuối tài liệu => lần đọc thứ hai () không thể đọc bất cứ thứ gì.

file1.***tell***(): the current position(字符数) of the cursor

file1.***seek***(offset[,whence]): di chuyển con trỏ đến vị trí xác định

whence(optional): vị trí bắt đầu của phần bù
- 0 từ đầu (mặc định),
- 1-từ vị trí hiện tại,
- 2-từ cuối tệp

offset: Bù lại

- **value +**: đến cuốitệp.
- **value -**: đến đầu tệp.

copying to files

```
f1 = open("1.txt","r")
f2 = open("2.txt","w")
f2.write(f1.read())
```

- hàm ghi file

```
def writefile(path):
  file=open(path,'w',encoding='utf-8')
  file.write('Hello world')
  file.close()
writefile('databaset.xt')
```

example

- hàm đọc file

```
def readfile():
  file=open('database.txt','r',encoding='utf-8')
  for line in file:
     print(line.strip())
  file.close()
readfile
```

Introduce

interpreter language (nn phiên dịch), chạy trực tiếp ko build file thực thi

cú pháp trong sáng, dễ đọc, hướng đối tượng, ...

Tài nguyên:

~Website
- Learn Python The Hardway: https://learnpythonthehardway.org/
- Learn Python Code academy: https://www.codecademy.com/learn/python
- Learn Python Treehouse: https://teamtreehouse.com/learn-to-code/python
- Learn Python code mentor: https://www.codementor.io/learn-python-online
- Visualize Python: http://www.pythontutor.com

~Ebooks:
- Fundamentals of Programming Python- Richard L. Halterman
- Learn Python the Hard Way, 3rd Edition
- Programming in Python 3, Mark Summerfield
- Python Algorithms, Magnus Lie Hetland
- Python and Tkinter Programming, John E. Grayson
- Professional IronPython - John Paul Mueller
- Python Tools for Visual Studio, Martino Sabia, Cathy Wang

Basic grammer

Kiểu DL & Khai báo biến

- **int**: số nguyên, ex: 5
- **float**: số thực, ex: 3.5
- **complex**: số phức

```
z= 2+3j
  khai bao: complex(2,3)
  xuất:
       print("thực=",z.real) => result: thực = 2
       print("ảo=",z.imag) => result: ảo = 3
```

Kiểu DL

- **str**: chuỗi

```
'...' / "..."
x= ['haha','cute',5]
chuỗi n` dòng: '''...''' / """..."""
```

- **bool**: luận lý

```
x1= True
x2= False
```

Khai báo biến

khi gán giá trị, tự động nội suy kiểu dl. 1 biến có thể n` kiểu dl. hàm **type()** check kiểu dl

```
- x= 5 // print(type(x)) => result: kiểu dl:
  int
- x= True // print(type(x)) => result: kiểu:
  bool
```

Xóa biến

biến đang :) # xóa đi thì ko use đc

```
x= 'obama' // print(x)
del x // print(x) => result:
error...
```

import thư viện sys

```
import sys
print("thông tin của int:")
print(sys.int_infor)
```

Check vùng lưu trữ

Commenting

```
# comment 1 dòng
```

```
"""..."""/'''...'''
comment n` dòng
```

Operator

hàm input(): nhập DL từ bàn phím giá trị là kiểu chuỗi

- **input()** cũng có thể nhập tiêu đề

```
x = (input('Nhap dữ liệu vào:'))
print('bạn đã nhập:',x)
print('thông tin kiểu dl:',
type(x))
```

- ép kiểu về **int** (ntstring)

```
x = int(input('Nhap dữ liệu vào:'))
print('bạn đã nhập:',x)
print('thông tin kiểu dl:',
type(x))
```

- ép kiểu về **float**: tương tự **int**
- ép kiểu về **boolean**

```
-viết hàm strtobool
def strtobool(s):
    return s.lower() in ('yes',
'true','is','1')

-ép kiểu
x= strtobool(input('Mời bạn nhập dữ liệu'))
print('giá trị bool:',x)
```

độ ưu tiên

số học

```
+,-,*,/

- 9/2 => result:
4.5 chia
```

```
//, %, **

- 9//2 => result:4 chia
lấy nguyên
- 9%2 => 1 chia lấy dư
- 2**4 => 8 lũy thừa
```

gán

```
=, +=, -=, *=, /=, //=, %=, **=

- x= 5 gán
- x*= 2 nhân gán => x= x*2
```

so sánh

Logic

```
==, !=, <, >, <=, >=, is, is not

- so sánh kép: 2 == x < 4 / 3 > x
<= 2
- x= 5 || y =5 || print(5 is 5)
=> result: True, True khi biến 2
bên toán tử cùng trả tới 1 đối
tượng hoặc cùng giá trị, ngc lại
False
- x= 5 || y =5 || print(5 is not
5) => result: False
```

and, or, not
- x=2016 || print(x%4 == 0 **and** x%100 !=0) =>result: True
- x = 7 || if(not x>=5): print('hello') else: print('con gà') => con gà

in, not in: check :)

```
'goad' in 'this is good example' => True
else
'good' not in 'this is good example' =>
False
```

INPUT

- dùng ( ) chỉ định rõ
- độ ưu tiên

```
1.   **
2.   *, /, %, //
3.   +, -
4.   <<, >, <, >=
5.   <>, ==, !=
6.   =, %=, /=, //=, -=, +=, *=,
**=
7.   is, is not
8.   not, or, and
```

OUTPUT

xuất lập

```
print('x'*20)
```

hàm **format()** xuất DL

```
print('{0}*x^2+{1}*x+{0}'.format(5,2))
```

xuất căn phải

```
print('{0:>2} {1:>11}'. format(1, 'hello
world'))
print('{0:>2} {1:>11}'. format(2, 'nam'))
print('{0:>2} {1:>11}'. format(3, 'dũng'))
print('{0:>2} {1:>11}'. format(4, 'kiên'))
```

ERROR

- **Syntax Errors:** lỗi cú pháp
- **Run-time Exceptions:** lỗi thực thi
- **Logic Errors:** lỗi nghiệp vụ

bắt lỗi

```
try:
    statements
except:
    statements
```

REVISION

- **import math**: import thư viện toán
- **hàm round(,):** làm tròn

practive
- tính dt, cvi hình tròn
- tính giờ, phút, giây

List Function

- **Insert()**: chèn vào vtri

```
list_name.insert(index,item)
```

- **Append()**: chèn cuối list

```
list_name.append()
```

- Cả **insert** và **append** đều có thể truyền mảng con, tức là trong list lại chứa list khác

```
lst=[1,2]
lst.append([10,20])
=> kq: [1,2,[10,20]]
```

- **Remove**: xóa ptu tồn tại và đầu tiên

```
list_name.remove(item)
```

- **Del**: xóa all,1 or n ptu

```
del list_name[]
del lst[2]   del lst[2:3]
del lst[4], lst[1]
```

- **Reverse**: đảo list, làm thay đổi list ban đầu

```
list_name.reverse()
```

- **Reversed**: đảo list nhg ko làm thay đổi list

```
lst2=reversed(lst)
```

- **Sort**: sắp xếp tăng

```
list_name.sort()
```

- **Storted**: sắp xếp, ko làm thay đổi list ban đầu

```
lst2=sorted(lst)
```

- 5x **giảm dần**:

```
lst.sort(reverse=True)
```

- **Slicing:** trích lọc list

```
list[begin: end:step]
```

- **Max/min:**

```
max/min (list_name)
```

⚠***In python, List cấu trúc mảng có các ptử index, nên List – Mảng*** => cùng số l/c chung

List

Revision

- Khởi tạo list, insert, count, sum ngtố, sort,del,remove
- Xóa tất cả ptu k trong list
- Check list đối xứng
- Khởi tạo ma trận, xuất dòng, xuất cột, max
- nhập 1 list có n số random ko trùng
- nhập dãy số tăng, nhập sai yêu cầu nhập lại
- hàm tính matrix hoán vị

- **Khai báo:**
List chứa all kiểu dl, nhg nên khai báo cùng kiểu dl trong 1 list

```
lst=[1,2,3,4,5]
lst1=['obama','hanoi']
lst2=[2]*10
```

print(lst) => kq: 1,2,3,4,5
print(lst[2])=> kq:3

- **Khởi tạo List ngẫu nhiên**

```
n= int(input("Số ptu list: "))
lst=[0]*n
for i in range(n):
    lst[i]= randrange(-100,100)
print(lst)
```

- **List đa chiều (ma trận)**

```
lst=[ [1,2]
      [6,8]
      [10,2]]
```

- **Khởi tạo List ma trận ngẫu nhiên**
- **Duyệt List ma trận**

- **Duyệt List:**

Cách 1: duyệt collection

```
for x in lst:
   print(x,end='\t')
```

Cách 2: duyệt index, duyệt xuôi or ngc

```
for i in range(len(lst)):
    print(lst[i],end='\t')
```

- **Gán gtrị tham chiếu:**

```
a= [2,3,4]      b=[2,3,4]
b=a
b[2]=1 => a[2] cũng =1
```

String

Basic

- **Chuỗi**: tập các kýtự nằm trong 3 dấu "" ... "" or '' ... '' => ex: str="" Khuất huy Kiên 471999"" ;dùng index để lấy kí tự=> str[1], KQ = 'K'

⚠***In python, List cấu trúc mảng có các ptử index, nên List – Mảng*** , => cùng số l/c chung

String Functions

✅**Syntax: object.method_name(parameter_list)**
=> ex: arr.upper()

- **upper(), lower():** đưa về in hoa, in thường
- **capitalize():** viết hoa chữ cái đầu
- **rjust()/ljust()/center():** căn lề phải, trái, căn giữa => nếu truyền 1 đối số thì chèn khoảng trắng, nếu có đối số 2 thì chèn đối số đó
- **strip():** chỉ xóa đc khoảng trắng/kí tự dư thừa 2 đầu,(tức là: khi gặp 1 kí tự khác nó sẽ dừng xóa)
- **startswith()/endswith():** check chuỗi có phải bđầu là ký tự?
- **find():** tìm kiếm chuỗi con, trả về vtri đầu tiên tìm thấy
- **rfind:** trả về vtri cuối cùng tìm thấy
- **count(): đếm số lần xuất hiện,**
  count("h",5,10): đếm n từ vtr 5->10
- **format:** định dạng chuỗi
  ex:**{ } { }.format(2,3)**
- **len():** đếm slg kí tự
- **substring:** trích lọc chuỗi,

```
x = "hello world"
print(x[2:4]) =>kq: " llo "
print(x[2:-2]) => kq: " llo worl "
```

- **split():** tách chuỗi ra list theo tham số truyền vào

```
print('Khuất Huy Kiên'.split(' '))
=> KQ: ['Khuất','Huy','Kiên']
```

- **splitlines():** tách dòng
- **join:** nối chuỗi, cách nhau 1 kí tự bất kì

```
print(' '.join(['Hello','World']))
```

=> kq: Hello World
- **Nối chuỗi bằng +**

```
a="he"  b="llo"  a+b =>kq: "hello"
```

Revision

- Check chuỗi đối xứng
- Tối ưu chuỗi
- Xử lý chuỗi: số chẵn, âm, ngtố,....
- Đếm in hoa/thường, chữ số, khoảng trắng, ngyuên âm,phụ âm
- xuất số âm trong chuỗi

Tuple/ Dic

- Là một cấu trúc mảng, ~ List.
- Đã khai báo rồi thì ko thay đổi đc giá trị
- Không use đc phương thức: append(), pop(), ...
- hỗ trợ truy xuất index, range, tìm kiếm,..

Tuples

```
my_tuple= ('x','y')
```

Dictionaries / Maps

Definition (a unique key for every value stored):

```
{ Key1 : Value1, Key2 : Value2, Key3 : Value3,
...}
```

```
Dictionary_name.values()
```

```
Dictionary_name.keys()
```

To get a specific value:

```
Dictionary_name.get(key_name)
```

To assign values:

```
Dictionary_name[Key_name] = New_value
```

You **cannot** join Dictionaries with '+'

Length of Dictionary:

```
len(Dictionary_name)
```

Delete value:

```
del(Dictionary_name[Key_name])
```

Functions

Functions often used

from math import ... cách import thư viện toán cụ thể

- sqrt, pow, log, log10, exp(ấn e^x)
- Degrees(đổi radians -> đ0), Radians(ngc lại)
- fabs(giá trị tuyệt đối)
- sin, cos, tan, cotan chỉ tính
- muốn tính: đưa góc về radians

- **round(x,y)**: ,x: số gốc; y: đvl làm tròn

Module time:
- **import time:** import thư viện time
- **time.time():** trả về chỉ tiết số giây
- **time.perf_corer():** trả về thời gian dưới dạng nano giây
- **time.Process_time():** trả về giá trị (tính bằng giây phân số) của tổng thời gian CPU của hệ thống và người dùng của tiến trình hiện tại.
- **time.sleep():** tam dừng chay (seconds)
- **time.localtime()**:(tm_year, tm_mon, tm_mday, tm_hour, tm_min, tm_sec, tm_wday, tm_yday, tm_isdst)

randrange(x,y): hàm random lấy số ngẫu nhiên, x <= và <y

exit(): thoát phần mềm. **# break**

eval(): tự tính chuỗi phép toán
x= eval("1+2*10")

Revision
- game đoán số
- tính Số.
- tính diện th
- tính DtS, cvi n dấu √
- vẽ hình

Basic

Hàm: khối lệnh thực hiện 1 công việc hoàn chỉnh (module), đc đặt tên và gọi thực thi nhiều vị trí trong chương trình
- còn dgl: **subroutine**(ctr con)
- có 2 loại hàm:

```
 > hàm thư viện: có sẵn, khi use phải
khai báo thư viện:
**from ... import ...,**
 > hàm tự đinh nghĩa
```

Xây dựng hàm

Cấu trúc

```
def Function_name(Parameter list):
    block
```

- hàm có thể có parameter list hoặc ko
- có thể trả về kết quả hoặc ko

```
 > def tong(a,b):   #trả kq
     return a+b
 > def xuat(data):  #ko
     print(data)
```

Gọi hàm

- xem hàm đó đ/n ntn?

```
 > có kq trả về: result= func_name([parameter])
     ... có thể gán cho biến
 > ko trả về: func_name([parameter])
     ... ko thể gán cho biến
```

- cách gọi: (xem tr)

```
 > tong(2,3)
```

- **Parameter mặc định:** đc hỗ trợ khi khai báo hàm
ex: hàm print, def print(self, *args, sep=' ', end='
', file=None):
- hỗ trợ giá trị mặc định khi **ko truyền vào**

```
def plus(a,b=5):
      return (a+b)
goi hàm: plus(2) => kq= 7 ; plus(2,2) => kq= 4
```

- có thể đổi thứ tự truyền parameter, ex hàm plus() => plus(b=2,a=1)

Global variable

```
g=5   --- > biến toàn cục
(global)
def demo():
    g=2  --- > biến local
    g=g+1
demo()
print(g) --- > biến toàn cục
```

- biến khai báo trg hàm, chỉ có pvi trong hàm(dgl: local)
- thoát khỏi hàm thì các biến local ko thể truy suất

Ngắc hđ

LIFO (last in first out)

```
 > def nhan():
     return 2*n
   x=double(3)
   print(x)
```

Viết tài liệu cho hàm

- thuận lợi cho partner dễ easy to use
- nên use """ ... """ (3 dấu ") viết ghi chú, viết đầu tiên khi khai báo hàm

- B1: tạo file documentfunction.py và viết code, giả sử có 2 hàm tong() và hieu():

```
 > def tong(a,b):
       """ hàm tính tổng 2 số """
       return (a+b)
 > def hieu(c,d):
       """ hàm tính hiệu 2 số """
       return (c-d)
```

- B2: vào tools -> chọn python or debug console
- B3: gõ from (tên file) import (tên hàm, nếu all thì "*")

```
 > from documentfunction import *
```

=> **done**
- **chú ý:** muốn xem ghi chú của tài liệu nào gõ: **help**(tên hàm)

Lambda expression

- khai báo hàm nặc danh thông qua **lambda expression**

```
lambda parameter_list: expression
lambda: từ khóa, parameter_list: ds
para muốn đ/n, expression biểu thức
đơn (ko nhập complex)
```

Ex:
- Đ/n 1 hàm

```
def handle(f,x):
   return f(x)
```

- ví lambda ko nhận hàm complex(phức tạp), nên đ/n các hàm độc lập rồi truyền vào sau.

```
def sochan(x):
    return x%2
def songuyento(x):
    dem=0;
    for i in range(1,x+1):
        if x%i==0:
            dem+=1
    return dem==2
```

- use

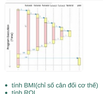
```
ret1= handle(sochan,6) # cách 1: tối
giản
print(ret1)  => True
ret2= handle(lambda x:sochan(x),6) #
cách 2: chi tiết
print(ret2) => True
ret3=handle(songuyento,5)
print(ret3) =>True
```

- **Note:** hàm sochan(), sole() đơn giản, có thể dùng lambda luôn ko cần viết hàm cụ thể.
kq1= handle(lambda x:x%2==0,7)
kq2= handle(lambda x:x%2!=0,7)

Đệ qui

- hàm tự gọi lại chính nó, nếu ko xử lý ổn bị tràn bộ đệm, nên dùng loop(for, while) giải quyết, đệ qui sau cùng.
- ex: tính giai thừa, tính dãy fibonacci
- use đệ qui: biết điểm dừng + quy luật

```
n!=? ; điểm dừng n=0 => n!=1; quy
luật: với n>=1 thì n!= n*(n-1)!
```



Revision

- tính BMI(chỉ số cân đối cơ thể)
- tính ROI
- tính dãy fibonacci
- note biến global, local
- kiểm tra số hoàn thiện & số thịnh vượng

Conditional statements

If: .. elif: ..else:

Boolean Expression
- True / False: check khi use if .. else, vòng lặp #

If <> : / else <> :

If <>: / elif <>: / else: lồng nhau

```
if n>5: print('a')
elif a>1: print('2')  else: print(' ')
```

pass: cho qua chỗ chưa code

```
if n>5:
    pass
else:
    print('a')
```

so sánh số thực

```
- # với int, kiểu float cần ngưỡng sai số
khi ss
- Ex:
d1 = 1.11 - 1.10
d2 = 2.11 - 2.10
print('d1= ',d1)
print('d2= ',d2)
diff = d2-d1
if diff <0:
    diff = -diff
if diff < 0.0000001:
    print('d1=d2')
else:
    print('d1!=d2')
```

use if/else như phép gán

```
a=5; b=7
c = 20 if a != b else 30
print(c)
```

```
x= int(input('Nhập 1 số: '))
print("Số đó là= ", "chẵn" if x%2 ==0 else
"lẻ")
```

Revision

- **năm nhuận:** (⋮ 4 and %100!=0) or ⋮ 400
- **đếm số ngày trong tháng**

```
 if month in (1,3,5,7,8,10,12):
     print("Tháng", month, "Có 31 ngày")
```

- **PTB 2**

chú ý dùng if: /elif: /else: và if: /else:

nhập 1 số 2 chữ số, in ra cách đọc: use if: / else: ,nếu dùng for nhanh hơn

```
chuc= n//10 dùng if <>: ... print ...
elif: <> ....
dvi= n%10  dùng if <>: ... print ...
elif: <> ....
```

- **nhập ngày(d/m/y), in ra ngày kế tiếp**
- **nhập a,b và phép(+,-,*,/), in KQ theo phép nhập.**
- **nhập 1 tháng, in ra xem thuộc quý mấy?**

Loop

- lặp đi lặp lại

While

```
while condition:
    for-block
else:
    else-block
```

For

```
for i in
range(0,5,1):
    print('i')
```

iterate the list/dictionary/tuple

range(start,end,step): step: bước nhảy
- **khuyết start**: tính từ 0
- **khuyết step**: zomi 1

```
for i in shoppingList: ...
```

while/else

```
while condition:
    while-block
else:
    else-block
```

=> nếu while end bthg(ko dừng break or ... đề end) thì else-block đc thực hiện ngay sau đó.

for/else

```
for expression:
    for-block
else:
    else-block
```

=> same while/else

Lệnh while/else, for/else

Loops lồng nhau

```
for i in range(n):
    for j in range(n):
        ....blocks....
```

same: while lồng while, or: for lồng while, ...

break: end a while loop prematurely

continue: jumps back to the top of the loop, các lệnh bên dưới continue sẽ ko thực thi trong lần lặp đó.

pass: filler statement

```
if <>:
else:
  pass [else not do anything]
```

Loop control statements

Revision

- **End** để xuất 1 dòng

- tính tổng dãy số
- check số nguyên tố
- xuất bảng cửu chương
- vẽ hình