

# Module & Function

Th.S Trần Đức Lợi  
[Pythonvietnam.info](http://Pythonvietnam.info)

# Ôn tập bài cũ


- Ôn lại nội dung đã học về **tập hợp**
- Chữa bài **Quản lý học sinh**


# Mục đích bài học

- Tìm hiểu về hàm và module trong python

# Function

- 
- Print 'pythonvietnam'
  - Print 'loitd'

- 
- Print 'lấy thông tin 1'
  - Print 'lấy thông tin 2'

- 
- Print 'Kết quả của bạn là ...'
  - Print 'xin chào'

# Function: lưu trữ và sử dụng lại

Welcome()

- Print 'pythonvietnam'
- Print 'loitd'

Processing()

- Print 'lấy thông tin 1'
- Print 'lấy thông tin 2'

getResult()

- Print 'Kết quả của bạn là ...'
- Print 'xin chào'

# Fuction

- Có 2 loại hàm trong python
  - Hàm Built-in
  - Các hàm do người dùng tự định nghĩa
- Tránh sử dụng tên biến là tên các hàm built-in

# Function: Định nghĩa

- Một hàm được định nghĩa bằng từ khóa **def**
- Về mặt định nghĩa:
  - Reusable code
  - Parameters
  - Results
- Chúng ta có thể gọi hàm bằng tên\_hàm(tham số)

# Function: Định nghĩa

- `N = len([2,3,4.567,8])`
- Ta đã gọi hàm `len()` với tham số là một list
- Def **bay**(ga):
  - `Ga.canh = dap(1000)`
  - `Ga.chay = False`
- **Bay**(ga\_quay)



# Function: built-in

- Các hàm ép kiểu
- Các hàm khởi tạo tập hợp
- Các hàm i/o

# Function: tự xây dựng

- Từ khóa def + tên hàm + ( + tham số + ):
- Căn lề thân (nội dung) hàm
- Hàm được định nghĩa nhưng chưa được chạy
- Để chạy hàm cần được gọi
- Khi đã định nghĩa một hàm, ta có thể gọi lại vô số lần -> sử dụng lại code

# Function: argument & parameters & results

- Def **bay**(ga):
  - Ga.canh = dap(1000)
  - Ga.chay = False
  - Return True
- **Bay**(ga\_quay)
- Hãy phân tích và chỉ ra 3 thông số

# Function: nhiều tham số

- Def **bay**(ga, docao, tocdo):
- Def **bay**(ga, docao, tocdo=100):
- Def **bay**(ga, docao=101, tocdo):
- Def **bay**(ga, docao=101, tocdo=100):

# Function: tham số mặc định

- Def **bay**(ga, docao=100):
  - Ga.canh = dap(1000)
  - Ga.chay = False
  - Ga.docao = docao
  - Return True
- **Bay**(ga\_quay)
- **Bay**(ga\_quay, 1000)

# Function

- Bài tập:
  - Hãy viết chương trình tính số tiền lương trong tháng của người lao động với hệ số lương a, lương cơ bản b (mặc định 400), số giờ làm thêm c, hệ số làm thêm d (mặc định 1.5), bảo hiểm e, công đoàn phí f. Phím 'Q' để thoát

# Function: biến toàn cục và biến cục bộ

- Biến cục bộ:
- Ví dụ:
  - *Def func(x):*
  - *Print 'x1: ', x*
  - *X = 43 #declare the local x*
  - *Print 'x2: ', x*
  - *X = 50*
  - *Func(x)*
  - *Print 'x3: ', x*

# Function: biến toàn cục và biến cục bộ

- Biến toàn cục
  - Từ khóa global
- Ví dụ:
  - *Def func(x):*
  - *Print 'x1: ', x*
  - *Global X = 43 #declare the local x*
  - *Print 'x2: ', x*
  - *X = 50*
  - *Func(x)*
  - *Print 'x3: ', x*



# Function: chỉ định tham số truyền giá trị

- Nếu như với một hàm có rất nhiều tham số nhưng ta chỉ muốn truyền vào 1-2 tham số?
- *Def func(a,b,c=1,d=2,e=3):*
- Chỉ truyền giá trị cho các biến a,b,e?

# Function: pass & void

- Lệnh return
- Void functions: return None
- Từ khóa pass sau lời khai báo hàm

# Function: docstrings

- Document Strings
- Tăng tính readable
- Ví dụ:
  - *Def func():*
  - *""" Function name*
  - *I am the description"""*
  - *A = b*

# Function: docstrings

- Xem nội dung docstrings của một hàm
- Print `function.__doc__`
- `Help(function)`

# Function: bài tập

- Viết lại bài quản lý học sinh sử dụng hàm với đầy đủ docstrings, tham số mặc định bổ sung thêm chức năng tìm kiếm theo tên?

# Module

- Sử dụng lại một tập hợp các hàm, biến
- Lưu tên file .py và sử dụng câu lệnh import, from ... import ...
- Import sys
- From sys import path

# Module: bài tập

- Hãy dùng sys module và biến sys.argv để in ra các tham số truyền vào khi chạy chương trình

# Module: pyc file

- Khi thực hiện import module, pyc file sẽ được sinh ra (bytecode)
- Thực nghiệm



# Module: import

- Import module
- From module import object
- Import module as x
  - Đặt alias trùng nhau
  - Chỉ hiểu alias
  - Nhiều alias cho cùng module
- Sự khác biệt và ưu nhược điểm?

# Module: import

- Import module as x
- Import module01 as x
- Import Module as x
- Module.methodx()
- X.methodx()
- Import Module as x
- Import module as y

# Module: `__name__`

- Tên của module
- Khi chạy trực tiếp thì module có tên là `'__main__'`
- Bài tập:
  - Hãy viết một module chỉ thực thi in ra câu chào “helloworld” nếu như được gọi trực tiếp, nếu được import thì in ra câu “imported”

# Module

- Bài tập: tự xây dựng một module đầu tiên với biến version và một hàm sayHello()
- Import module này vào chương trình và gọi các hàm, biến của module
- Xem chi tiết mô tả các hàm trong module đã nêu

# Module: path

- Thư mục hiện tại
- PYTHONPATH
- Sys.path

# Module: reload

- Một ví dụ điển hình là: khi đã import 1 module và đang thực hiện chương trình, ta thực hiện thay đổi source nguồn của module đó. Nếu không reload lại, các thay đổi này sẽ không có tác dụng.
- Reload()

# Module: dir

- *dir()*
- *import mod*
- *dir(mod)*
- *x = 1*
- *dir()*
- *import mod as hg*
- *dir()*
- *del x*
- *dir()*

# Module

- Bài tập:
  - Chuyển bài quản lý học sinh về module để thực hiện import và sử dụng thêm chức năng sửa một học sinh bất kỳ?



# Package

- Thư mục có cấu trúc
- Modules
- Package con
- Bài tập: tự xây dựng một package mới?

# Package

- module girls.py:
- *def say():*
- *print 'we are girls'*
- module boys.py:
- *def say():*
- *print 'we are boys'*
- `__init__.py`:
  - `import girls`
  - `import boys`
- - import và sử dụng package pkg:
- `import pkg`
- `pkg.girls.say()`
- `pkg.boys.say()`

# Tổng kết bài học

- Function
  - Built-in
  - Người dùng tự định nghĩa
  - Void
  - Nhiều tham số
  - Tham số mặc định
- Module
  - Reload
  - Path
  - `__name__`
  - Import commands
- Package
  - Giới thiệu package

# Bài tập

- Xây dựng chương trình **quản lý thư viện sách** với function, module, import, built-in