



# KỸ THUẬT LẬP TRÌNH

ThS. Bùi Việt Thắng

E: [thangbv82@gmail.com](mailto:thangbv82@gmail.com)

T: 0983085387



# GIỚI THIỆU HỌC PHẦN

- **Thời lượng: 2 tín chỉ**
  - 18 tiết lí thuyết
  - 24 tiết thực hành
- **Đánh giá kết quả học tập**
  - Điểm chuyên cần ( $\geq 20\%$  trên tổng số 11 buổi)
    - Đi học đầy đủ, đúng giờ
    - Tham gia xây dựng bài
  - Kiểm tra giữa kì: báo cáo bài tập lớn, bài thực hành (điểm  $\geq 5$ )
  - Thi kết thúc học phần: thực hành - Python



# TÀI LIỆU THAM KHẢO

## ■ Tài liệu chính:

- ❑ [1] Jose Manuel Ortega - Mastering Python for Networking and Security\_ Leverage the scripts and libraries of Python version 3.7 and beyond to overcome networking and security issues-Packt Publishing Ltd.
- ❑ [2] TJ O'Connor - Violent Python A cookbook for hackers, forensic analysts, penetration testers and security engineers-Syngress (2013).

## ■ Tài liệu tham khảo:

- ❑ [3] Gray Hat Python - Python Programming for Hackers and Reverse Engineers (2009).



# NỘI DUNG MÔN HỌC

## ■ Gồm 4 bài:

- ❑ Bài 01: Tổng quan về ngôn ngữ Python
- ❑ Bài 02: Kiểm thử xâm nhập (red team)
- ❑ Bài 03: Phòng thủ (blue team)
- ❑ Bài 04: Một số vấn đề ATTT khác



# Bài 01: Tổng quan về ngôn ngữ Python

## 1. Làm việc với Python script

- ❑ Giới thiệu về ngôn ngữ Python
- ❑ Cài đặt môi trường thực thi
- ❑ Thư viện python

## 2. Lập trình cơ bản

- ❑ Lập trình hướng cấu trúc
- ❑ Lập trình hướng đối tượng OOP

## 3. Lập trình nâng cao

- ❑ Mô đun , gói
- ❑ Làm việc với file
- ❑ Quản lý file, lỗi và xử lý ngoại lệ
- ❑ Lập trình socket, http



# Ví dụ mở đầu

- Nhân 2 số lớn
- VD: 123456789123456789123456789123456789  
123456789123456789123456789123456789  
123456789 \* 1000000000

## Ngôn ngữ C++

```
void nhan(solon n1,solon n2,solon &n)
{
    node *p,*p1,*p2;    long long i,dem=0;
    solon a;taosolon(a);    solon b;taosolon(b);
    p=n2.cuoi;    while(p!=NULL)
    {
        nhan1so(n1,p->x,a);
        for(i=0;i<dem;i++)
```

...

Quá nhiều dòng code

## Ngôn ngữ Python

```
>print
1234567891234567891
23456789123456789
1234567891234567891
23456789123456789
123456789 *
1000000000
```



# Giới thiệu Python

- ❑ Thiết kế năm 1991, tác giả: Guido Van Rossum
- ❑ Đặc điểm:
  - **Là ngôn ngữ kịch bản và thông dịch**
  - **Trong sáng, gần gũi và dễ học**
    - Tăng cường sử dụng từ khóa tiếng Anh, hạn chế các kí hiệu
    - Hướng tới sự đơn giản
      - ✓ Có while bỏ do while
      - ✓ Có elif bỏ switch – case
    - Ưu tiên cho việc đọc lại code
- ❑ Ngôn ngữ lập trình đơn giản, dễ học: Python có cú pháp rất đơn giản, rõ ràng. Nó dễ đọc và viết hơn rất nhiều khi so sánh với những ngôn ngữ lập trình khác như C++, Java, C#. Python làm cho việc lập trình trở nên thú vị, cho phép bạn tập trung vào những giải pháp chứ không phải cú pháp.



# Giới thiệu Python

## ■ Tại sao chọn Python?

- ❑ Nó là một ngôn ngữ nguồn mở đa nền tảng: Linux, DOS, Windows, và macOS X.
  - Hòa hợp tốt với các ngôn ngữ khác: Java, .NET, C/C++, ...
  - Python chạy mọi nơi: Unix, Windows, Mac, Nokia S60
- ❑ Nhiều thư viện, mô-đun và dự án tập trung vào bảo mật máy tính được viết bằng Python.
- ❑ Có sẵn rất nhiều tài liệu, cùng với một cộng đồng người dùng rất lớn.





# Môi trường lập trình



IDLE



SPYDER



PyCharm



eric



Atom



jupyter



Anaconda



**Thonny**  
Python IDE for beginners



# Thư viện python\*

Library	Type	Commits	Contributors	Releases	Watch	Star	Fork	Commits / Contributors	Commits / Releases	Star/ Contributors
NumPy	Data wrangling	15980	522	125	280	4286	2012	31	128	8
SciPy	Data wrangling	17213	489	91	244	3043	1775	35	189	6
Pandas	Data wrangling	15089	762	76	626	9394	3709	20	199	12
Matplotlib	Visualization	21754	588	60	413	5190	2517	37	363	9
Seaborn	Visualization	1699	71	11	176	3878	580	24	154	55
Bokeh	Visualization	15724	223	40	322	5720	1401	71	393	26
Plotly	Visualization	2486	33	7	149	2044	512	75	355	62
SciKit-Learn	Machine learning	21793	842	80	1650	18246	9997	26	272	22
Keras	Machine learning	3519	428	28	1025	15043	5227	8	126	35
TensorFlow	Machine learning	16785	795	29	5002	55486	26433	21	579	70
Theano	Machine learning	25870	300	23	520	6171	2116	86	1125	21
Scrapy	Data scraping	6325	243	78	1427	20124	5353	26	81	83
NLTK	NLP	12449	196	20	376	4649	1358	64	622	24
Gensim	NLP	2878	179	43	300	4182	1595	16	67	23
Statsmodels	Statistics	8960	119	19	194	2019	977	75	472	17

\*Thống kê hoạt động thư viện trên github



# Bài 01: Tổng quan về ngôn ngữ Python

## 1. Làm việc với Python script

- ❑ Giới thiệu về ngôn ngữ Python
- ❑ Cài đặt môi trường thực thi
- ❑ Thư viện python

## 2. Lập trình cơ bản

- ❑ **Lập trình hướng cấu trúc**
- ❑ **Lập trình hướng đối tượng OOP**

## 3. Lập trình nâng cao





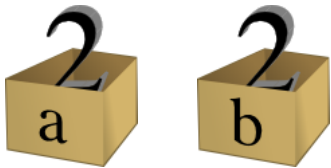

- ❑ Mô đun , gói
- ❑ Làm việc với file
- ❑ Quản lý file, lỗi và xử lý ngoại lệ
- ❑ Lập trình socket, http



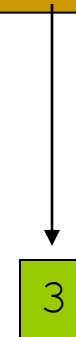
# Lập trình cơ bản

## ❑ Biến

- ❑ Khai báo một biến bằng cách gán giá trị cụ thể cho nó. Biến sẽ tự động được giải phóng khi ra khỏi phạm vi của chương trình sử dụng nó.

	Ngôn ngữ khác (biến)	Python (tên)
$a = 1$		
$a = 2$		
$b = a$		

```
x = 3  
print x
```





# Lập trình cơ bản

## ❑ Kiểu dữ liệu

- Số: int, long, float, complex
- Toán tử : + - / \* %

```
x = 14
y = 3
print "Sum: ", x + y
print "Product: ", x * y
print "Remainder: ", x % y
```

```
Sum: 17
Product: 42
Remainder: 2
```

```
x = 5
print "x started as", x
x = x * 2
print "Then x was", x
x = x + 1
print "Finally x was" ,x
```

```
x started as 5
Then x was 10
Finally x was 11
```



# Lập trình cơ bản

## □ Kiểu dữ liệu

### □ Chuỗi: str

- Đặt trong cặp dấu : ‘, “
- Nối chuỗi: +, +=
- Độ dài chuỗi x: len(x)
- Định dạng chuỗi:

```
a = "pan"  
b = "cake"  
a = a + b  
print a
```

pancake



# Lập trình cơ bản

## □ Kiểu dữ liệu

- Chuỗi
- Một số phương thức: upper, lower, split, replace ...

Chuyển chữ hoa

Chuyển chữ thường

Thay thế i thành a

```
x = "A simple sentence"
print x
print x.upper()
print x.lower()
x = x.replace("i", "a")
print x
```

```
A simple sentence
A SIMPLE SENTENCE
a simple sentence
A sample sentence
```



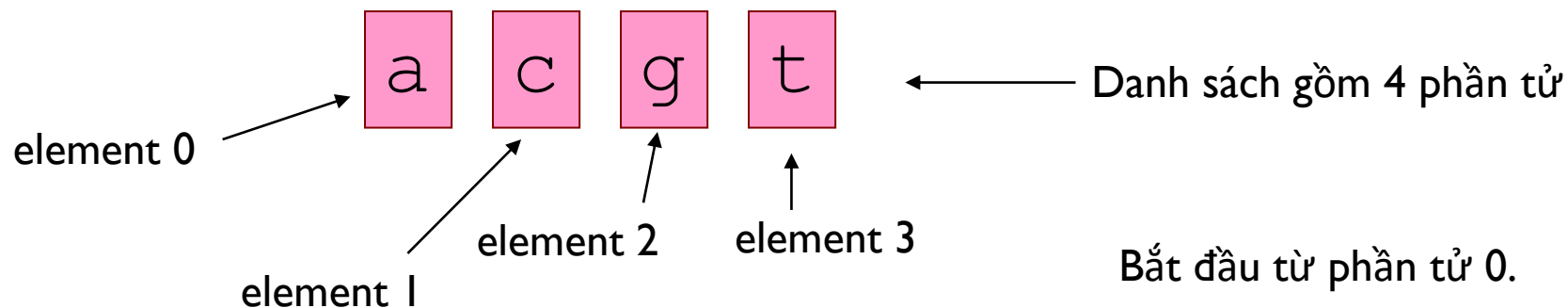
# Lập trình cơ bản

## ❑ Kiểu dữ liệu

### ▪ Kiểu danh sách: list

```
nucleotides = ['a', 'c', 'g', 't']  
print "Nucleotides: ", nucleotides
```

```
Nucleotides: ['a', 'c', 'g', 't']
```







# Lập trình cơ bản

## □ Kiểu dữ liệu

### ▪ Kiểu danh sách: list

- Truy cập đến các phần tử bằng chỉ số, -1 là chỉ số phần tử cuối
- Toán tử: +, \*
- Các phương thức: append, extend, insert, remove, pop, index, count, sort, reserve

```
x = ['a', 'c', 'g', 't']  
i=2  
print x[0], x[i], x[-1]
```

a g t

```
x = ['a', 't', 'g', 'c']  
print "x =", x  
x.sort()  
print "x =", x  
x.reverse()  
print "x =", x
```

```
x = a t g c  
x = a c g t  
x = t g c a
```



# Lập trình cơ bản

## ▪ Kiểu tuple

- ✓ Giống list nhưng dữ liệu không thay đổi được
- ✓ Sử dụng dấu () để khai báo

```
t = (1, 2, 3, 4, 5)
q = (1, 2, (3, 4), 5)
```

## ▪ Kiểu từ điển

- ✓ {từ khóa: giá trị}
- ✓ Phương thức: keys, values, pop, items, has\_key...

```
genes["cop"] = "45837"
```

Dùng dấu [ ] để tra cứu  
đến khóa



# Lập trình cơ bản

```
>>> tlf = {"Michael" : 40062, \
"Bingding" : 40064, "Andreas": 40063 }
>>> tlf.keys()
['Bingding', 'Andreas', 'Michael']
>>> tlf.values()
[40064, 40063, 40062]
>>> tlf["Michael"]
40062
>>> tlf.has_key("Lars")
False
>>> tlf["Lars"] = 40070
>>> tlf.has_key("Lars") # now it's there
True
>>> for name in tlf.keys():
...     print name, tlf[name]
...
Lars 40070
Bingding 40064
Andreas 40063
Michael 40062
```

Từ điển

Xem các từ khóa

Xem các giá trị

Xem các giá trị với khóa

Kiểm tra khóa

Chèn một cặp khóa-giá trị

Duyệt từ điển và in ra



# Lập trình cơ bản

- **Kiểu tập hợp: set**

- Tập các phần tử không có thứ tự và không có phần tử trùng lặp
- Các phép toán: -(hiệu), ^ (hiệu đối xứng), &(giao), |(hợp)

```
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a # unique letters in a
set(['a', 'r', 'b', 'c', 'd'])
>>> a - b # letters in a but not in b
set(['r', 'd', 'b'])
>>> a | b # letters in either a or b
set(['a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'])
>>> a & b # letters in both a and b
set(['a', 'c'])
>>> a ^ b # letters in a or b but not both
set(['r', 'd', 'b', 'm', 'z', 'l'])
```



# Điều khiển luồng

```
if <expression>:  
    <statements>  
elif <expression>:  
    <statements>  
else:  
    <statements>
```

```
var1 = 100  
if var1:  
    print "1 - Nhan mot gia tri true"  
    print var1  
else:  
    print "1 - Nhan mot gia tri false"  
    print var1
```



# Vòng lặp

```
while <expression>:  
    <statements>  
else:  
    <statements>
```

```
count = 0  
while count < 5:  
    print count, " là nhỏ hơn 5"  
    count = count + 1  
else:  
    print count, " là không nhỏ hơn 5"
```

```
while <expression>:  
    while  
    <expression>:  
        <statements>  
        <statements>
```



```
0 là nhỏ hơn 5  
1 là nhỏ hơn 5  
2 là nhỏ hơn 5  
3 là nhỏ hơn 5  
4 là nhỏ hơn 5  
5 là không nhỏ hơn 5
```



# Vòng lặp

```
for letter in 'Python'  
    print 'Chu cai hien tai :', letter
```

```
qua = ['chuoi', 'tao', 'xoai']  
for qua in qua:  
    print 'Ban co thich an :', qua
```

```
print "Good bye!"
```

```
for <name> in <container>:  
    for <name> in <container>:  
        <statements>  
    <statements>
```

```
for <name> in  
<container>:  
    <statements>  
else:  
    <statements>
```

```
Chu cai hien tai : P  
Chu cai hien tai : y  
Chu cai hien tai : t  
Chu cai hien tai : h  
Chu cai hien tai : o  
Chu cai hien tai : n  
Ban co thich an : chuoi  
Ban co thich an : tao  
Ban co thich an : xoai  
Good bye!
```



# Vòng lặp

- **break:** kết thúc vòng lặp hiện tại và truyền điều khiển tới cuối vòng lặp.
- **continue:** trả về điều khiển tới phần ban đầu của vòng lặp. Lệnh này bỏ qua lần lặp hiện tại và bắt buộc lần lặp tiếp theo của vòng lặp diễn ra.
- **pass:** được sử dụng khi một lệnh là cần thiết theo cú pháp nhưng bạn không muốn bất cứ lệnh hoặc khối code nào được thực thi.





# Lập trình cơ bản

Nhớ thụt đầu dòng

```
x = 149
y = 100
if x > y:
    print x, "is greater than", y
else:
    print x, "is less than", y
```

149 is greater than 100

```
x = 0
while x < 10:
    print x,
    x+=1
```

0 1 2 3 4 5 6 7 8 9

```
a = ['bo', 'me', 'con']
for x in a:
    print x,
```

bo me con



# Quy tắc thụt đầu dòng

```
1 if True:
2     print ("Hello")
3     print ("True")
4
5 else:
6     print ("False")
```



# Lập trình cơ bản

- Quy tắc viết một lệnh (Statement) trên nhiều dòng:
  - Sử dụng dấu \ để nói với Python rằng lệnh bao gồm cả dòng tiếp theo

```
value = 1 + \  
        2 + \  
        3
```

- Quy tắc viết nhiều lệnh trên một dòng
  - sử dụng dấu chấm phẩy ( ; ) để ngăn cách giữa các câu lệnh

```
a = 'One'; b = "Two"; c = "Three"
```



# Các phiên bản ngữ pháp Python

- Để in ra màn hình dòng chữ "Hello World", trong phiên bản 2.x, sử dụng dòng lệnh "print" mà không cần cặp dấu ngoặc ( ):

```
# Ngữ pháp Python 2.x
```

```
print "Hello World"
```

- Với Ngữ pháp Python 3.x để in ra dòng chữ "Hello World" bắt buộc phải để nó trong dấu ngoặc ( ), nếu không sẽ bị thông báo lỗi.

```
# Ngữ pháp Python 3.x
```

```
print ("Hello World")
```



# Hàm

## ❑ Cú pháp:

```
def tên_hàm (tham_biến_1, tham_biến_2, ...)  
    # lệnh ...  
    return giá_trị_hàm
```

- ❑ Một tên được bắt đầu bằng các chữ cái viết hoa (A-Z), hoặc viết thường (a-z), hoặc kí tự gạch dưới ( \_ ), theo sau đó có thể là các kí tự khác hoặc không có gì.
- ❑ Python không chấp nhận các kí tự: @, \$ và % xuất hiện ở trong tên.
- ❑ Python là một ngôn ngữ lập trình có phân biệt chữ viết hoa và chữ viết thường, MyObject và myobject là hai tên khác nhau.



# Hàm

## ■ Hàm

```
def max(a):  
    max = a[0]  
    for x in a:  
        if x > max:  
            max = x  
    return max
```

```
data = [1, 5, 1, 12, 3, 4, 6]  
print "Data:", data  
print "Maximum:", max(data)
```

← Khai báo hàm

} Thân hàm

← Trả về

← Gọi hàm

```
Data: [1, 5, 1, 12, 3, 4, 6]  
Maximum: 12
```



## Đệ quy

Đệ quy là quá trình mà trong đó một đối tượng sẽ tự gọi lại chính nó.

Trong Python, chúng ta biết rằng một hàm có thể gọi các hàm khác. Thậm chí có thể thực hiện triển khai cho một hàm tự gọi chính nó. Kiểu cấu trúc này được gọi là các hàm đệ quy.

```
def vi_du(a):  
    print(a)  
    if (a == 1):  
        return 1  
    vi_du(a - 1)  
vi_du(10)
```

10  
9  
8  
7  
6  
5  
4  
3  
2  
1



# Đệ quy

## Ưu điểm của đệ quy

- ❑ Các hàm đệ quy làm cho đoạn mã trông gọn gàng và dễ nhìn hơn.
- ❑ Một nhiệm vụ phức tạp có thể được chia thành các bài toán con đơn giản hơn bằng cách sử dụng đệ quy.
- ❑ Việc tạo hàm đệ quy dễ dàng hơn so với việc sử dụng một số phép lặp lồng nhau.

## Nhược điểm của đệ quy

- ❑ Đôi khi tính Logic đằng sau kỹ thuật đệ quy sẽ rất khó hiểu.
- ❑ Đệ quy là rất tốn kém (không hiệu quả) vì chúng chiếm nhiều bộ nhớ và thời gian.
- ❑ Các hàm đệ quy rất khó để gỡ lỗi.





# Bài 01: Tổng quan về ngôn ngữ Python

1. Làm việc với Python script
  - ❑ Giới thiệu về ngôn ngữ Python
  - ❑ Cài đặt môi trường thực thi
  - ❑ Thư viện python
2. Lập trình cơ bản
  - ❑ Lập trình hướng cấu trúc
  - ❑ **Lập trình hướng đối tượng OOP**
3. Lập trình nâng cao
  - ❑ Mô đun , gói
  - ❑ Làm việc với file
  - ❑ Quản lý file, lỗi và xử lý ngoại lệ
  - ❑ Lập trình socket, http



# Lập trình hướng đối tượng OOP

Lập trình hướng đối tượng (Object-oriented programming-OOP) là một kỹ thuật hỗ trợ, cho phép lập trình viên trực tiếp làm việc với các đối tượng mà họ định nghĩa lên. Hiệu quả của kỹ thuật này giúp tăng năng suất, đơn giản hoá độ phức tạp khi bảo trì cũng như mở rộng phần mềm. Hiện nay có khá nhiều ngôn ngữ lập trình theo hướng đối tượng như C++, Java, PHP, ... và còn cả Python..



# Lớp

## Lớp

Một lớp là một bản thiết kế cho đối tượng. Chúng ta có thể coi một lớp như một bản phác thảo của một ngôi nhà với các chi tiết. Nó chứa tất cả các chi tiết về tên, màu sắc, kích thước,...

```
class
sinh_vien:
    pass
```

## Đối tượng

Một đối tượng (thể hiện) là một khởi tạo của một lớp. Khi lớp được định nghĩa, chỉ có mô tả cho đối tượng được định nghĩa. Do đó, không có bộ nhớ nào được cấp phát.

```
a =
sinh_vien()
```



# Hàm tạo

```
class sinh_vien:
    "Đây là lớp sinh viên"
    def in_thong_tin(self):
        print('Sinh viên')
        print("ID là",
self.ID)
    def __init__(self, ID):
        self.ID = ID

sv = sinh_vien(100)
sv.in_thong_tin()
```

Hàm `__init__`() được gọi bất cứ khi nào một đối tượng mới của lớp đó được khởi tạo. Hàm này còn được gọi là hàm tạo trong lập trình hướng đối tượng (OOP). Chúng ta thường sử dụng nó để khởi tạo tất cả các đối tượng của lớp.



# Lớp

```
class sinh_vien:
    truong_hoc = "KMA"
    def __init__(self, ID_sv, ten_sv):
        self.ID_sv = ID_sv
        self.ten_sv = ten_sv
```

```
a = sinh_vien(10, "Nam")
b = sinh_vien(30, "Trung")
```

Tên là:Nam, ID là: 10  
Tên là:Trung, ID là: 30  
Trường: KMA

```
print("Tên là:{}, ID là:
{}".format(a.ten_sv, a.ID_sv))
print("Tên là:{}, ID là:
{}".format(b.ten_sv, b.ID_sv))
print("Trường:
{}".format(a.__class__.truong_hoc))
```



# Phương thức

Các phương thức là các hàm được định nghĩa bên trong phần thân của một lớp. Chúng được sử dụng để xác định các hành vi của một đối tượng.

```
class sinh_vien:
    truong_hoc = "KMA"
    def __init__(self, ID_sv, ten_sv):
        self.ID_sv = ID_sv
        self.ten_sv = ten_sv
    def study(self, mon_hoc):
        print("Sinh viên {} học môn  
{})".format(self.ten_sv, mon_hoc))

a = sinh_vien(10, "Nam")
b = sinh_vien(30, "Trung")
a.study("ATTT")
b.study("ATMMT")
```



# Xóa phương thức và đối tượng

```
1 class sinh_vien:
2     "Đây là lớp sinh viên"
3     def in_thong_tin(self):
4         print('Sinh viên')
5         print("ID là", self.ID)
6     def __init__(self, ID):
7         self.ID = ID
8
9 sv = sinh_vien(100)
10 sv.in_thong_tin()
11 del sv.ID
12 sv.in_thong_tin()
```

```
1 class sinh_vien:
2     "Đây là lớp sinh viên"
3     def in_thong_tin(self):
4         print('Sinh viên')
5         print("ID là", self.ID)
6     def __init__(self, ID):
7         self.ID = ID
8
9 sv = sinh_vien(100)
10 sv.in_thong_tin()
11 del sv
12 sv.in_thong_tin()
```



# Đóng gói

```
class sinh_vien:
    def __init__(self):
        self.__ID = 10
    def in_thong_tin(self):
        print("ID là: {}".format(self.__ID))
    def setID(self, ID):
        self.__ID = ID

sv = sinh_vien()
sv.in_thong_tin()
sv.__ID = 30
sv.in_thong_tin()
sv.setID(100)
sv.in_thong_tin()
```

1	ID là: 10
2	ID là: 10
3	ID là: 100





# Đa hình

```
1 class sinh_vien_AT:
2     def in_thong_tin(self):
3         print("Sinh viên ATTT")
4
5 class sinh_vien_CT:
6     def in_thong_tin(self):
7         print("Sinh viên CNTT")
8
9 def vi_du(sv):
10     sv.in_thong_tin()
11
12 sv1 = sinh_vien_AT()
13 sv2 = sinh_vien_CT()
14 vi_du(sv1)
15 vi_du(sv2)
```



# Kế thừa

```
class lớp_cha:  
    Đoạn mã  
class lớp_con(lớp_cha):  
    Đoạn mã 2
```

```
1 class sinh_vien:  
2     def __init__(self, ID):  
3         self.ID = ID  
4     def in_thong_tin(self):  
5         print("ID của sinh viên là: ",self.ID)  
6  
7 class sinh_vien_Y(sinh_vien):  
8     def __init__(self):  
9         sinh_vien.__init__(self,100)  
10    def in_thong_tin_2(self):  
11        print('Đây là sinh viên trường Y')
```



# Đa kế thừa

Một lớp có thể được dẫn xuất từ nhiều hơn một lớp cơ sở trong Python, tương tự như C++. Đây được gọi là đa kế thừa. Trong đa kế thừa, các tính năng của tất cả các lớp cơ sở được kế thừa trong lớp dẫn xuất.

```
1 class Lớp_cha_1:
2     Đoạn mã 1
3 class Lớp_cha_2:
4     Đoạn mã 2
5 class Lớp_dẫn_xuất(Lớp_cha_1, Lớp_cha_2):
6     Đoạn mã 3
```



# Kế thừa đa mức

Trong kế thừa đa mức, các tính năng của lớp cơ sở và lớp dẫn xuất được kế thừa trong lớp dẫn xuất mới.

```
1 class Lớp_cha:
2     Đoạn mã 1
3 class Lớp_dẫn_xuất_1(Lớp_cha):
4     Đoạn mã 2
5 class Lớp_dẫn_xuất_2(Lớp_dẫn_xuất_1):
6     Đoạn mã 3
```



# Ghi đề phương thức

❑ Hai hàm được tích hợp là `isinstance()` và `issubclass()` được sử dụng để kiểm tra các lớp kế thừa.

❑ Hàm `isinstance()` trả về giá trị `true` nếu đối tượng là một thể hiện của lớp hoặc các lớp khác dẫn xuất từ nó. Mỗi lớp trong Python kế thừa từ lớp cơ sở là `object`.

```
class sinh_vien:
    def __init__(self, ID):
        self.ID = ID
    def in_thong_tin(self):
        print("ID của sinh viên là: ",self.ID)
```

```
class sinh_vien_AT(sinh_vien):
    def __init__(self):
        sinh_vien.__init__(self,100)
    def in_thong_tin_2(self):
        print('Đây là sinh viên ATTT')
```

```
sv = sinh_vien_AT()
print(isinstance(sv,sinh_vien_AT))
print(isinstance(sv,sinh_vien))
print(isinstance(sv,float))
```



# Nạp chồng toán tử

Các toán tử trong Python hoạt động cho các lớp được tích hợp sẵn. Nhưng cùng một toán tử có thể thực hiện các thao tác khác nhau với các kiểu dữ liệu khác nhau.

Tính năng này trong Python cho phép cùng một toán tử có thể thực hiện các thao tác khác nhau tùy theo ngữ cảnh được gọi là **nạp chồng toán tử**.

```
class vi_du:
    def __init__(self, a, b):
        self.a = a
        self.b = b
    def __str__(self):
        return "({0}, {1})".format(self.a,
self.b)
    def __add__(self, other):
        a = self.a + other.a
        b = self.b + other.b
        return vi_du(a, b)
t1 = vi_du(100, 102)
t2 = vi_du(104, 108)
```

(204, 210)



# Nạp chồng toán tử

Toán tử	Biểu thức	Hàm bên trong
Nhỏ hơn	$a < b$	<code>a.__lt__(b)</code>
Nhỏ hơn hoặc bằng	$a \leq b$	<code>a.__le__(b)</code>
Bằng	$a == b$	<code>a.__eq__(b)</code>
Không bằng/ Khác	$a \neq b$	<code>a.__ne__(b)</code>
Lớn hơn	$a > b$	<code>a.__gt__(b)</code>
Lớn hơn hoặc bằng	$a \geq b$	<code>a.__ge__(b)</code>



# Bài 01: Tổng quan về ngôn ngữ Python

## 1. Làm việc với Python script

- ❑ Giới thiệu về ngôn ngữ Python
- ❑ Cài đặt môi trường thực thi
- ❑ Thư viện python

## 2. Lập trình cơ bản

- ❑ Lập trình hướng cấu trúc
- ❑ Lập trình hướng đối tượng OOP

## 3. Lập trình nâng cao

- ❑ Mô đun , gói
- ❑ Làm việc với file
- ❑ Quản lý file, lỗi và xử lý ngoại lệ
- ❑ Lập trình socket, http





# Module và package

- ❑ Các module chuẩn: sys (system) , math (mathematics) , re (regular expressions)
- ❑ Load module sử dụng từ khóa import
- ❑ Người dùng có thể tự viết module, lưu với tên .py

```
import math  
print math.sqrt(100)
```

```
10
```



# File

Các thao tác xử lý File trong Python diễn ra theo thứ tự sau: Mở tệp, Đọc hoặc ghi tệp, Đóng tệp.

Python có một hàm `open()` đã được tích hợp sẵn để mở một tệp. Hàm này trả về một đối tượng tệp, còn được gọi là *handle*, vì nó được sử dụng để đọc hoặc sửa đổi tệp.

Chế độ	Mô tả
r	Mở tệp để đọc.
w	Mở tệp để ghi dữ liệu. Tạo 1 tệp mới nếu tệp không tồn tại.
x	Tạo tệp.
a	Mở tệp để thêm dữ liệu vào cuối. Tạo 1 tệp mới nếu không tồn tại.
t	Mở tệp ở chế độ dạng văn bản.
b	Mở tệp ở chế độ dạng nhị phân.
+	Mở tệp để cập nhật.



# File

Phương thức	Mô tả
<code>close()</code>	Đóng một tệp được mở.
<code>detach()</code>	Trả về luồng dữ liệu thô được phân tách từ bộ nhớ đệm.
<code>fileno()</code>	Trả về số nguyên (Đặc tả của tệp) của tệp.
<code>flush()</code>	Làm sạch bộ đệm của tệp.
<code>isatty()</code>	Trả về giá trị true nếu luồng của tệp kết nối với một thiết bị.
<code>read(n)</code>	Đọc nhiều nhất n ký tự từ tệp. Đọc cho tới khi kết thúc tệp nếu nó là giá trị âm hoặc None.
<code>readable()</code>	Trả về true nếu luồng của tệp có thể đọc.
<code>readline(n=-1)</code>	Đọc và trả về 1 hàng từ tệp. Đọc nhiều nhất là n byte.
<code>readlines(n=-1)</code>	Đọc và trả về danh sách các hàng trong tệp. Đọc nhiều nhất là n ký tự hoặc số byte được chỉ định.



## Ngoại lệ

Python có nhiều ngoại lệ đã được tích hợp và được đưa ra khi chương trình gặp lỗi. Khi những ngoại lệ này xảy ra, trình thông dịch Python dừng quá trình xử lý hiện tại và chuyển nó cho quá trình khác cho đến khi nó được xử lý. Nếu không được xử lý, chương trình sẽ bị dừng lại.

```
try:  
    a = 1 / 0  
except ZeroDivisionError:  
    print("Lỗi 1 xảy ra")  
except (IndentationError, UnicodeError):  
    print("Lỗi 2 xảy ra")  
except:  
    print("Các lỗi còn lại xảy ra")
```



# Ngoại lệ

- ❑ Trong lập trình Python, các ngoại lệ được đưa ra khi lỗi xảy ra trong thời gian thực thi. Chúng ta cũng có thể đưa ra các ngoại lệ theo cách thủ công bằng cách sử dụng từ khóa **raise**.
- ❑ Chúng ta có thể tùy chọn truyền các giá trị cho ngoại lệ để làm rõ lý do tại sao ngoại lệ đó được đưa ra.

```
raise KeyboardInterrupt("Lỗi xảy ra  
vì ...")
```



# Ngoại lệ

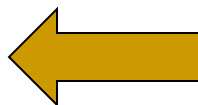
## try...else

```
try:
    x = 1/1
    print(x)
except ZeroDivisionError:
    print("Cannot divide by 0!")
else:
    print("Nothing wrong.")
```



```
1.0
Nothing wrong.
```

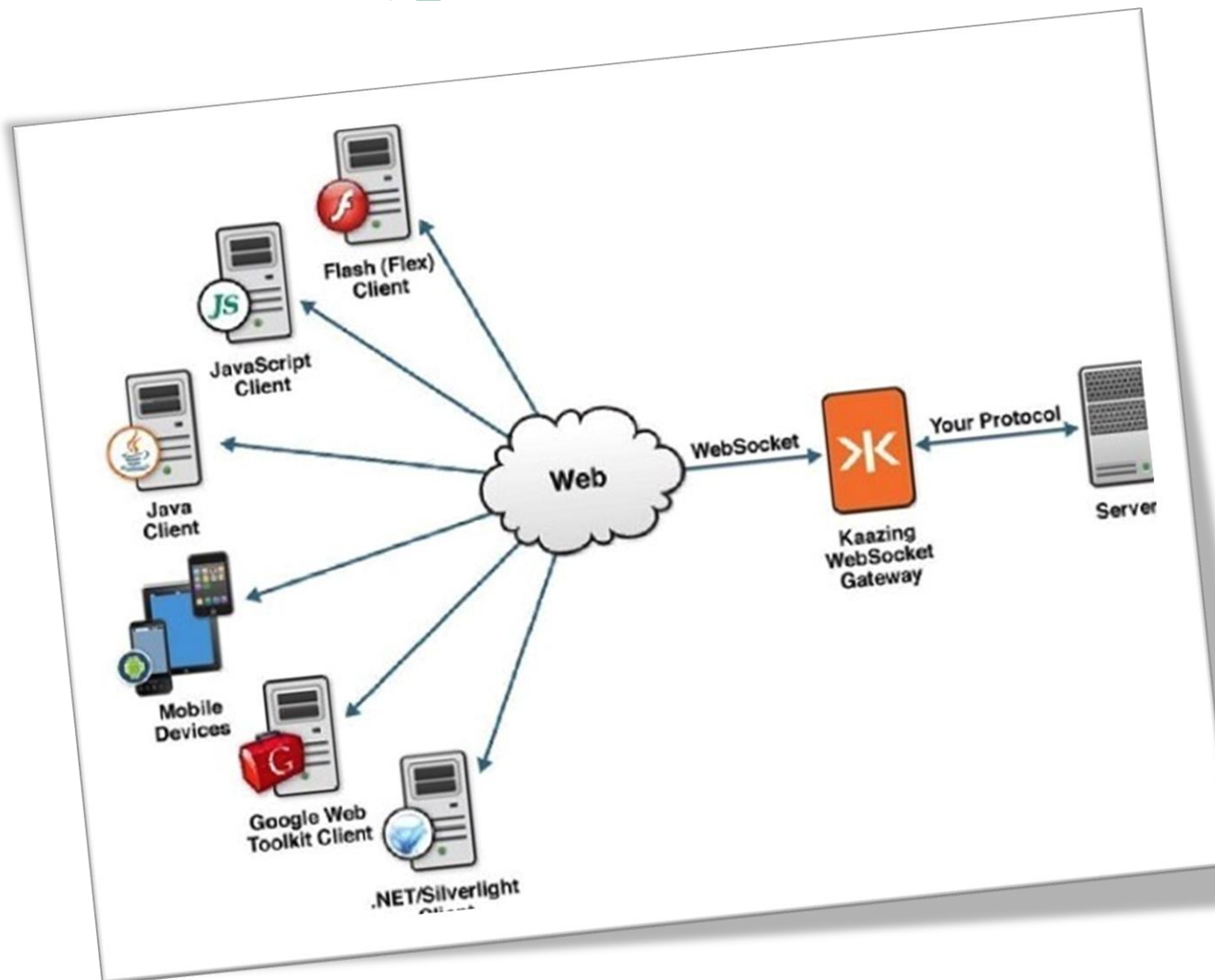
```
Cannot divide by 0!
The 'try except' is finished!
```



## try...finally

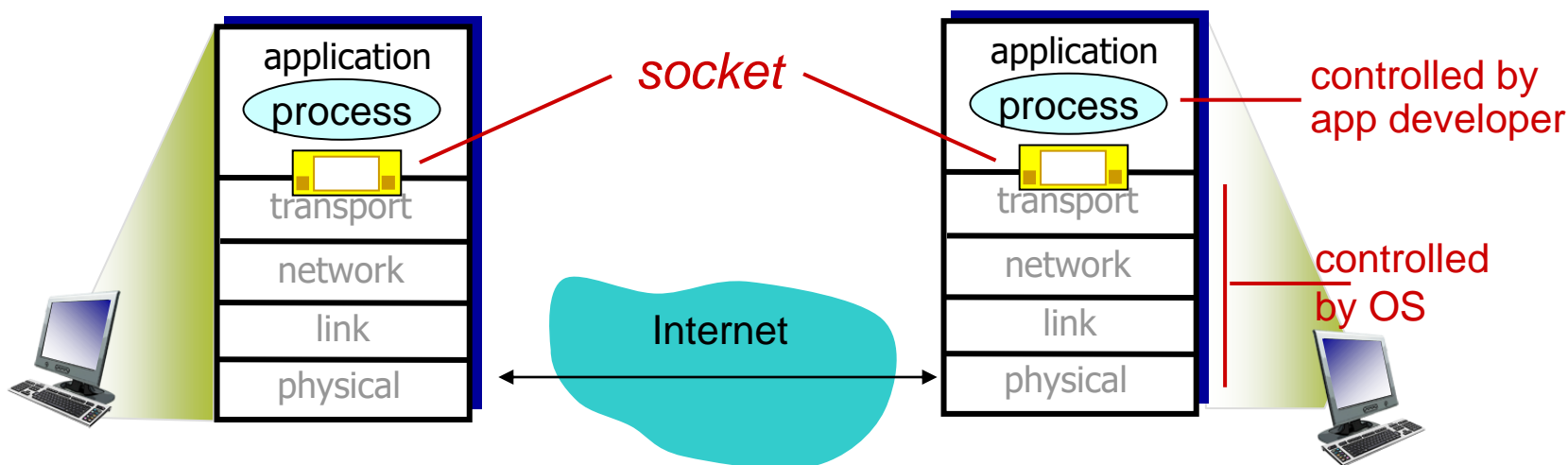
```
try:
    x = 1/0
    print(x)
except ZeroDivisionError:
    print("Cannot divide by 0!")
finally:
    print("The 'try except' is finished!")
```

# Lập trình Socket



# Sockets

- process sends/receives messages to/from its **socket**
- Process analogous to home, socket analogous to door
  - ❑ sending process shoves message out door
  - ❑ sending process relies on transport infrastructure on other side of door which brings message to socket at receiving process



- ❑ API: (1) choice of transport protocol; (2) ability to fix a few parameters





# Addressing processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- Q: does IP address of host on which process runs suffice for identifying the process?
  - A: No, *many* processes can be running on same host
- To identify the receiving process, two pieces of information need to be specified:
  - (1) *the address of the host* and
  - (2) *port number* that specifies the receiving *process* in the destination host.
- *identifier* includes both *IP address* and *port numbers* associated with process on host.
- to send HTTP message to gaia.cs.umass.edu web server:
  - *IP address:* 128.119.245.12
  - *Port number:* 80



# Lập trình Socket

- ❑ Một socket là một điểm cuối (endpoint) của một liên kết thông tin liên lạc 2 chiều giữa 2 chương trình chạy trên hệ thống mạng.
- ❑ Một socket được liên kết với một cổng (PORT) để tầng TCP có thể định danh ứng dụng nào đã gửi dữ liệu đến.
- ❑ Socket được sử dụng cho phép một process (hay program) này nói chuyện với một process (hay program) khác và duy trì kết nối này.
- ❑ Lập trình socket là lập trình cho phép người dùng kết nối các máy tính truyền tải và nhận dữ liệu từ máy tính thông qua mạng.



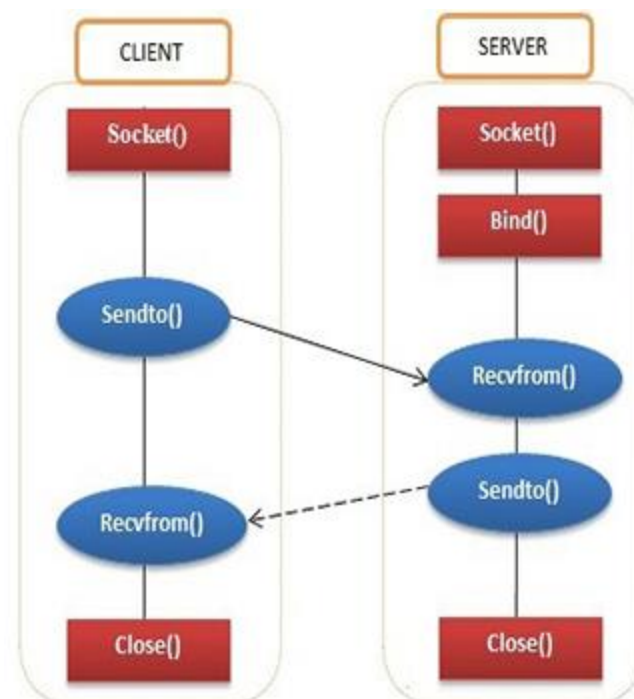
# Lập trình socket

*Có 02 loại socket tương ứng với các dịch vụ ở tầng transport:*

- ❑ **UDP:** unreliable datagram
- ❑ **TCP:** reliable, byte stream-oriented

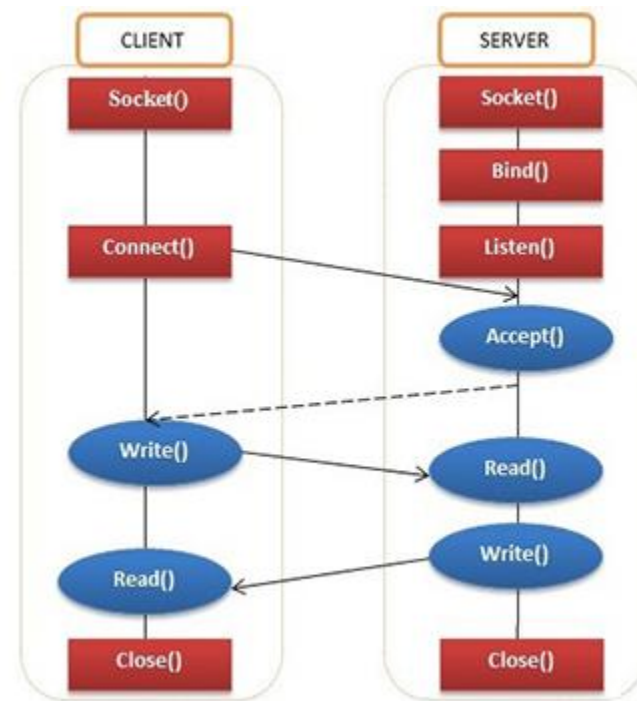
# Datagram Socket

- Dựa trên giao thức UDP( User Datagram Protocol) việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 quá trình. Do đó, hình thức này được gọi là **socket không hướng kết nối**.
- **Ưu điểm:** Do không yêu cầu thiết lập kết nối, không phải có những cơ chế phức tạp nên tốc độ giao thức khá nhanh, thuận tiện cho các ứng dụng truyền dữ liệu nhanh như chat, game.....
- **Hạn chế:** Ngược lại với giao thức TCP thì dữ liệu được truyền theo giao thức UDP không được tin cậy, có thể không đúng trình tự và lặp lại.



# Stream Socket

- Dựa trên giao thức TCP( Transmission Control Protocol), việc truyền dữ liệu chỉ thực hiện giữa 2 quá trình đã thiết lập kết nối. Do đó, hình thức này được gọi là **socket hướng kết nối**.
- **Ưu điểm:** Có thể dùng để liên lạc theo mô hình client và sever. Nếu là mô hình client /sever thì sever lắng nghe và chấp nhận từ client. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng thứ tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn. Đồng thời, mỗi thông điệp gửi phải có xác nhận trả về và các gói tin chuyển đi tuần tự.
- **Hạn chế:** Có một đường kết nối (địa chỉ IP) giữa 2 tiến trình nên 1 trong 2 tiến trình phải đợi tiến trình kia yêu cầu kết nối.



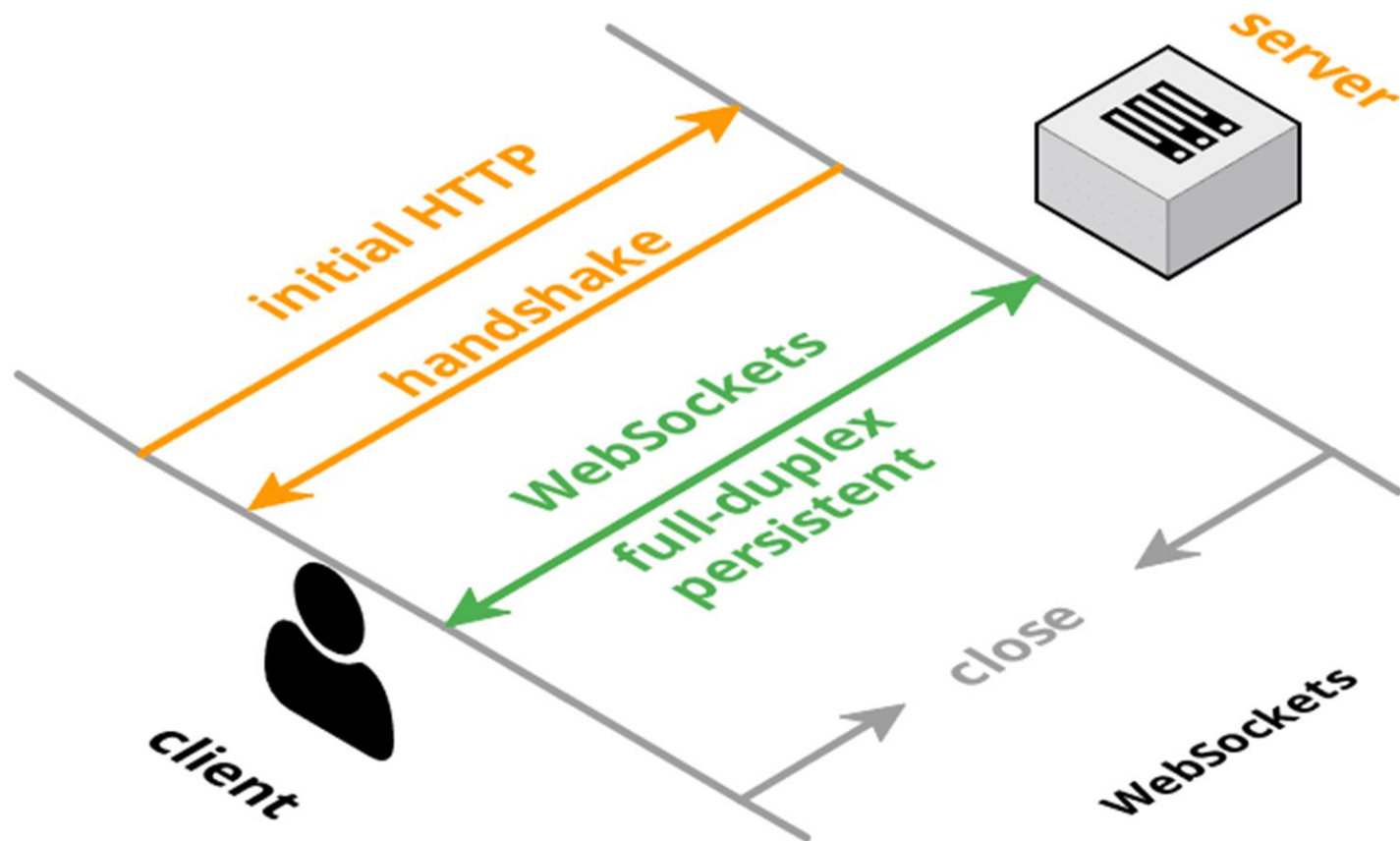


# Phân loại Socket

❑ **Web socket** là công nghệ hỗ trợ giao tiếp 2 chiều giữa **client** và **server** dựa trên một giao thức kết nối (thường là **TCP**) để tạo một kết nối hiệu quả và ít tốn kém.

❑ **Unix socket** là một kết nối chia sẻ dữ liệu giữa các **process** khác nhau trong cùng một máy tính. Khác với Web socket sử dụng một giao tiếp mạng để kết nối trên môi trường internet, Unix socket được thực hiện ở **nhân hệ điều hành** nhờ vậy có thể tránh được các bước như kiểm tra routing, do đó đem lại tốc độ nhanh hơn và nhẹ hơn.

# Web socket





# Cơ chế hoạt động Socket

Đầu tiên client sẽ mở một kết nối TCP và cố gắng kết nối với server qua một PORT quy định.

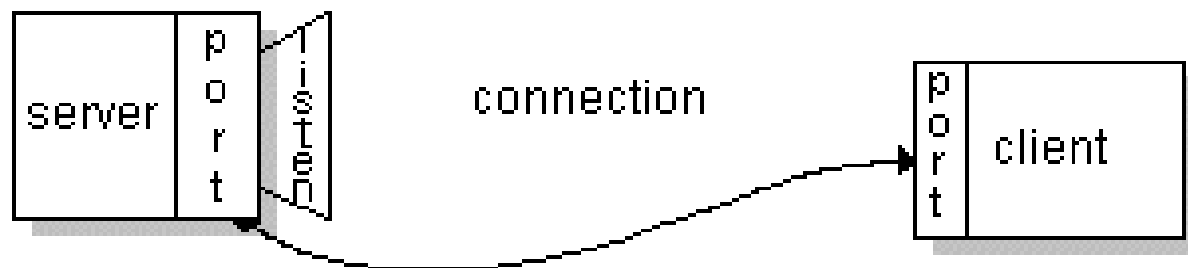






# Cơ chế hoạt động Socket

Nếu kết nối thành công, server chấp nhận kết nối nó sẽ mở ra một PORT và duy trì kết nối này. Kể từ đây, cặp endpoint `<Client_IP, PORT1>` vs `<Server_IP, PORT2>` được đặt một trạng thái là Keep-Alive, tức là kết nối có còn sống hay không.

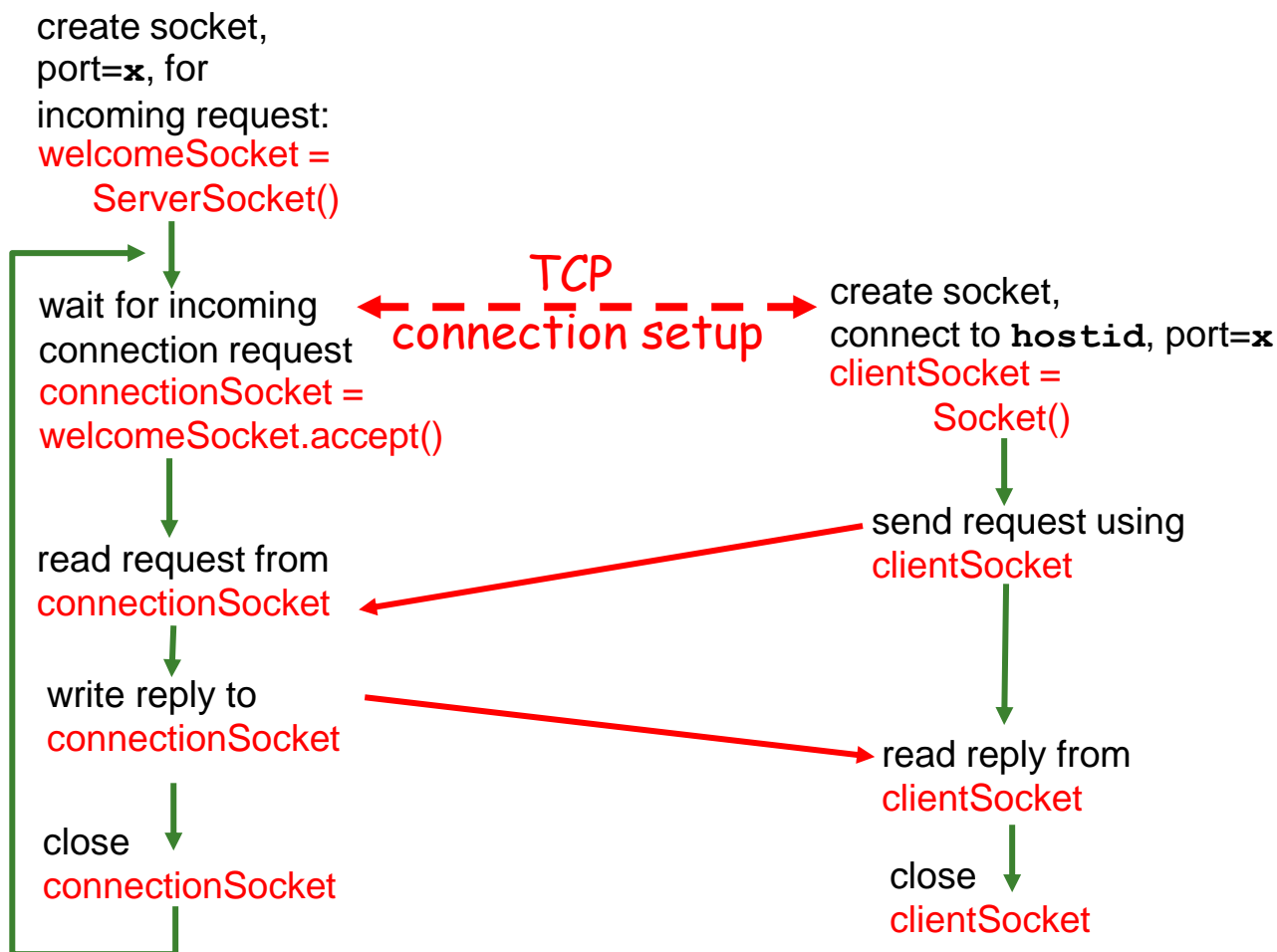




# Client/server socket interaction: TCP

Server (running on `hostid`)

Client





# Socket module trong python

Import module socket

```
import socket
```

Khởi tạo đối tượng socket trong module này với cú pháp:

```
socket.socket(AddressFamily,  
socketType, Protocol)
```



# Socket module trong python

- ❑ **AddressFamily** là cách chúng ta thiết lập địa chỉ kết nối. Trong Python thì hỗ trợ chúng ta 3 kiểu.
  - **AF\_INET** kiểu này là thiết lập dưới dạng ipv4.
  - **AF\_INET6** kiểu này là thiết lập dưới dạng ipv6.
  - **AF\_UNIX**
- ❑ **SocketType** là cách thiết lập giao thức cho socket. Thông thường thì sẽ là **SOCK\_STREAM (TCP)** hoặc **SOCK\_DGRAM (UDP)**.



# Socket module trong python

**Protocol** tham số thiết lập loại giao thức. Tham số này có thể không cần thiết lập. Mặc định sẽ bằng 0.

- ❖ **bind(ip\_address, port)** : Dùng để lắng nghe đến địa chỉ ip và cổng.
- ❖ **connect(ip\_address)** : Thiết lập một kết nối từ client đến server.
- ❖ **recv(bufsize, flag)** : Phương thức này được sử dụng để nhận dữ liệu qua giao thức TCP.
- ❖ **recvfrom(bufsize, flag)** : Nhận dữ liệu qua UDP
- ❖ **send(byte, flag)** : Phương thức này để gửi dữ liệu qua TCP.
- ❖ **sendto(bytes, flag)** : Gửi dữ liệu qua UDP.
- ❖ **close()** : Đóng một kết nối.



# Ví dụ:

- Lập trình socket theo mô hình client-server với UDP và TCP:
  1. Client đọc dữ liệu từ bàn phím và gửi dữ liệu đến Server.
  2. Server nhận dữ liệu và chuyển ký tự thành chữ hoa.
  3. Server gửi dữ liệu đã sửa đổi cho Client.
  4. Client nhận dữ liệu đã sửa đổi và hiển thị dòng trên màn hình.



# Client/server socket interaction: UDP

Server (running on `hostid`)

Client

create socket,  
port= x.

`serverSocket =  
DatagramSocket()`

read datagram from  
`serverSocket`

write reply to  
`serverSocket`  
specifying  
client address,  
port number

create socket,

`clientSocket =  
DatagramSocket()`

Create datagram with server IP and  
port=x; send datagram via  
`clientSocket`

read datagram from  
`clientSocket`

close  
`clientSocket`



# UDP Socket: Client\_UDP.py

```
import socket
```

Khai báo module

Mô-đun socket tạo thành cơ sở của tất cả các giao tiếp mạng trong Python. Bằng cách bao gồm dòng này, sẽ có thể tạo các socket trong chương trình





# UDP Socket: Client\_UDP.py

```
import socket
```

```
serverName = '127.0.0.1'
```

```
serverPort = 12000
```

Khai báo tên server

Khai báo port



# UDP Socket: Client\_UDP.py

```
import socket

serverName = '127.0.0.1'
serverPort = 12000

clientSocket =
socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
```

Tạo socket của client, clientSocket.

- Tham số đầu tiên cho biết kiểu địa chỉ IP, cụ thể AF\_INET chỉ ra mạng đang sử dụng IPv4.
- Tham số thứ 2 cho biết loại socket là UDP.



# UDP Socket: Client\_UDP.py

```
import socket

serverName = '127.0.0.1'
serverPort = 12000

clientSocket =
socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)

message = input('Input lowercase
sentence:')
```

input() là một hàm tích hợp sẵn trong Python. Khi lệnh này được thực thi, người dùng tại client được nhắc với dòng chữ “Input data:”. Sau đó, người dùng sử dụng bàn phím để nhập dữ liệu, dữ liệu này được đưa vào biến *message*.



# UDP Socket: Client\_UDP.py

```
import socket

serverName = '127.0.0.1'
serverPort = 12000

clientSocket =
socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)

message = input('Input lowercase
sentence:')
clientSocket.sendto(message.encode(),
(serverName, serverPort))
```

Bây giờ chúng ta có một socket và một message, chúng ta sẽ gửi message qua socket đến server đích

Phương thức `sendto()` đính kèm địa chỉ đích (`serverName`, `serverPort`) vào message và gửi gói kết quả vào socket của process, `clientSocket`.



# UDP Socket: Client\_UDP.py

```
import socket

serverName = '127.0.0.1'
serverPort = 12000

clientSocket =
socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)

message = input('Input lowercase
sentence:')
clientSocket.sendto(message.encode(),
(serverName, serverPort))

modifiedMessage, serverAddress =
clientSocket.recvfrom(2048)
```

khi một gói từ Internet đến socket của client, dữ liệu của gói được đưa vào biến `modifiedMessage` và địa chỉ nguồn của gói được đưa vào biến `serverAddress`

Phương thức `recvfrom` lấy kích thước bộ đệm 2048 làm đầu vào.



# UDP Socket: Client\_UDP.py

```
import socket

serverName = '127.0.0.1'
serverPort = 12000

clientSocket =
socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)

message = input('Input lowercase
sentence:')
clientSocket.sendto(message.encode()
,(serverName, serverPort))

modifiedMessage, serverAddress =
clientSocket.recvfrom(2048)
print (modifiedMessage.decode())

clientSocket.close()
```

Xuất ra màn hình

Đóng Socket



# UDP Socket: Server\_UDP.py

```
import socket
```

← Khai báo module

```
serverPort = 12000
```

← Khai báo port

```
serverSocket =  
socket.socket(socket.AF_INET,  
socket.SOCK_DGRAM)  
serverSocket.bind(('', serverPort))
```

← Tạo socket

```
print('The server is ready to  
receive:')
```

```
while 1:
```

```
    message, clientAddress =  
serverSocket.recvfrom(2048)  
    print(message)  
    modifiedMessage = message.upper()  
    serverSocket.sendto(modifiedMessage,  
clientAddress)
```



# UDP Socket: Server\_UDP.py

```
import socket
```

Khai báo module

```
serverPort = 12000
```

Khai báo port

```
serverSocket =
```

```
socket.socket(socket.AF_INET,
```

Tạo socket

```
socket.SOCK_DGRAM)
```

```
serverSocket.bind(('', serverPort))
```

Gán số cổng 12000 cho  
socket của máy chủ

```
print('The server is ready to  
receive:')
```

```
while 1:
```

```
    message, clientAddress =
```

```
serverSocket.recvfrom(2048)
```

```
    print(message)
```

```
    modifiedMessage = message.upper()
```

```
    serverSocket.sendto(modifiedMessage,  
clientAddress)
```

vòng lặp while sẽ cho  
phép UDPServer nhận và  
xử lý các gói từ máy  
khách vô thời hạn





# Kiểm tra các dịch vụ mở

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Mar 24 20:24:33 2021
4
5 @author: chieu
6 """
7
8 import socket
9 ip = '42.113.206.26' # IP dantri.com.vn
10 portlist = [21,22,23,80,443]
11 for port in portlist:
12     sock= socket.socket(socket.AF_INET,socket.SOCK_STREAM)
13     result = sock.connect_ex((ip,port))
14     print(port,":", result)
15     sock.close()
```

```
In [4]: runfile('D:/Google Drive/ml_waf/untitled0.py')
```

```
21 : 10061
```

```
22 : 10061
```

```
23 : 10061
```

```
80 : 0
```

```
443 : 0
```



# Ứng dụng socket

```
8 import socket
9 try:
10     print("gethostname:", socket.gethostname())
11     print("gethostbyname", socket.gethostbyname('www.actvn.edu.vn'))
12     print("gethostbyname_ex", socket.gethostbyname_ex('www.actvn.edu.vn'))
13     print("gethostbyaddr", socket.gethostbyaddr('103.21.148.154'))
14     print("getfqdn", socket.getfqdn('www.actvn.edu.vn'))
15     print("getaddrinfo", socket.getaddrinfo("www.actvn.edu.vn", None, 0, socket.SOCK_STREAM))
16 except socket.error as error:
17     print(str(error))
18     print("Connection error")
```

In [15]: `runfile('D:/Google Drive/ml_waf/untitled0.py', wdir='D:/Google Drive/ml_waf')`

gethostname: DLL310

gethostbyname 103.21.148.154

gethostbyname\_ex ('actvn.edu.vn', ['www.actvn.edu.vn'], ['103.21.148.154'])

gethostbyaddr ('WIN-KPBK3PJF1Q6', [], ['103.21.148.154'])

getfqdn WIN-KPBK3PJF1Q6

getaddrinfo [(<AddressFamily.AF\_INET: 2>, <SocketKind.SOCK\_STREAM: 1>, 0, '', ('103.21.148.154', 0))]



# Socket TCP

```
8 import socket
9 SERVER_IP = "127.0.0.1"
10 SERVER_PORT = 9998
11 # family = Internet, type = stream socket means TCP
12 server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13 server.bind((SERVER_IP,SERVER_PORT))
14 server.listen(5)
15 print("[*] Server Listening on %s:%d" % (SERVER_IP,SERVER_PORT))
16 client,addr = server.accept()
17 client.send("I am the server accepting connections...".encode())
18 print("[*] Accepted connection from: %s:%d" %(addr[0],addr[1]))
19 def handle_client(client_socket):
20     request = client_socket.recv(1024)
21     print("[*] Received request : %s from client %s" %(request, client_socket.getpeername()))
22     client_socket.send("ACK".encode())
23     while True:
24         handle_client(client)
25     client_socket.close()
26 server.close()
```



# Socket TCP

```
7 import socket
8 host="127.0.0.1"
9 port = 9998
10 try:
11     mysocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12     mysocket.connect((host, port))
13     print('Connected to host ' + str(host) + ' in port: ' + str(port))
14     message = mysocket.recv(1024)
15     print("Message received from the server", message)
16     while True:
17         message = input("Enter your message > ")
18         mysocket.send(bytes(message.encode('utf-8')))
19         print("Message received from the server", mysocket.recv(1024))
20         if message == "quit":
21             break
22 except socket.errno as error:
23     print("Socket error ", error)
24 finally:
25     mysocket.close()
```



# Sinh viên chuẩn bị trước

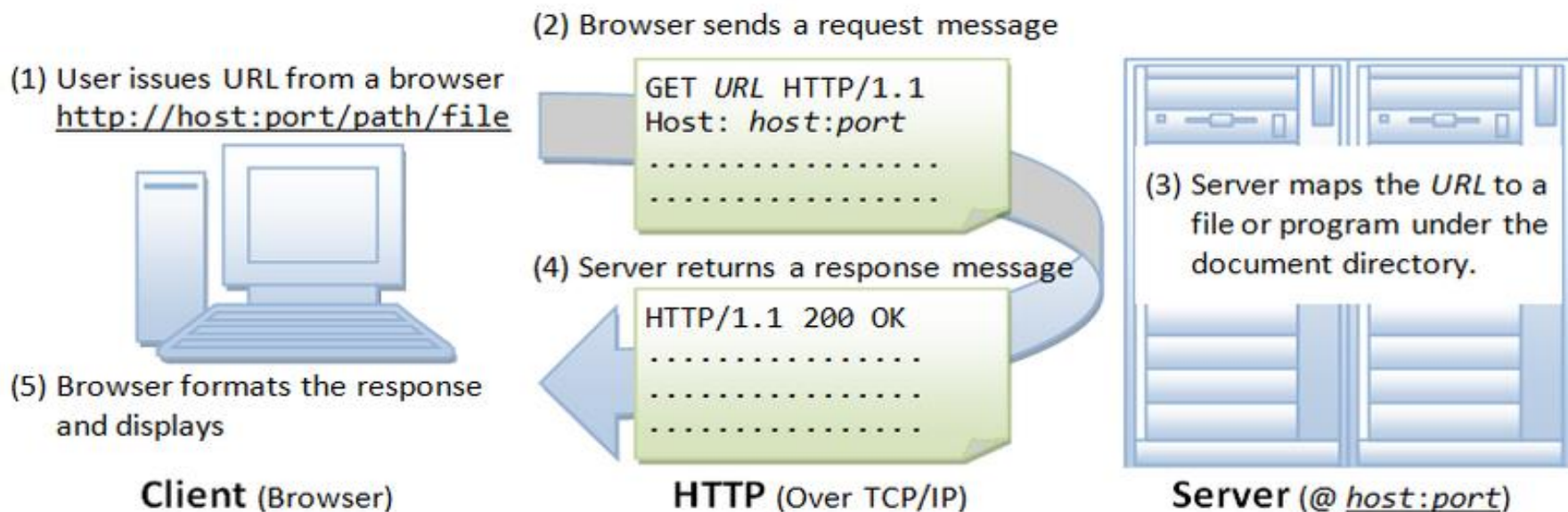
- ❑ Tìm hiểu module **requests**
- ❑ Tạo POST request với REST API
- ❑ Quản lý exception
- ❑ Xây dựng http client với https
- ❑ Tìm hiểu asyncio

**\*Mỗi đề tài 1 sinh viên**

# HTTP



Http (HyperText Transfer Protocol) là giao thức truyền tải siêu văn bản được sử dụng trong www dùng để truyền tải dữ liệu giữa Web server đến các trình duyệt Web và ngược lại qua cổng 80.





# HTTP

GET /index.html HTTP/1.1

**Request Line**

Date: Thu, 20 May 2004 21:12:55 GMT

Connection: close

**General Headers**

Host: www.myfavoriteamazingsite.com

From: joeblow@somewebsitesomewhere.com

Accept: text/html, text/plain

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

**Request Headers**

**HTTP  
Request**

HTTP/1.1 200 OK

**Status Line**

Date: Thu, 20 May 2004 21:12:58 GMT

Connection: close

**General Headers**

Server: Apache/1.3.27

Accept-Ranges: bytes

**Response Headers**

Content-Type: text/html

Content-Length: 170

Last-Modified: Tue, 18 May 2004 10:14:49 GMT

**Entity Headers**

**Message Body**

**HTTP  
Response**

<html>

<head>

<title>Welcome to the Amazing Site!</title>

</head>

<body>

<p>This site is under construction. Please come  
back later. Sorry!</p>

</body>

</html>

**Message Body**



# HTTP message format

- two types of HTTP messages: *request, response*
- HTTP request message:
  - ASCII (human-readable format)

request line  
(GET, POST,  
HEAD commands)

header  
lines

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/5.0
Connection: close
Accept-language: fr
```

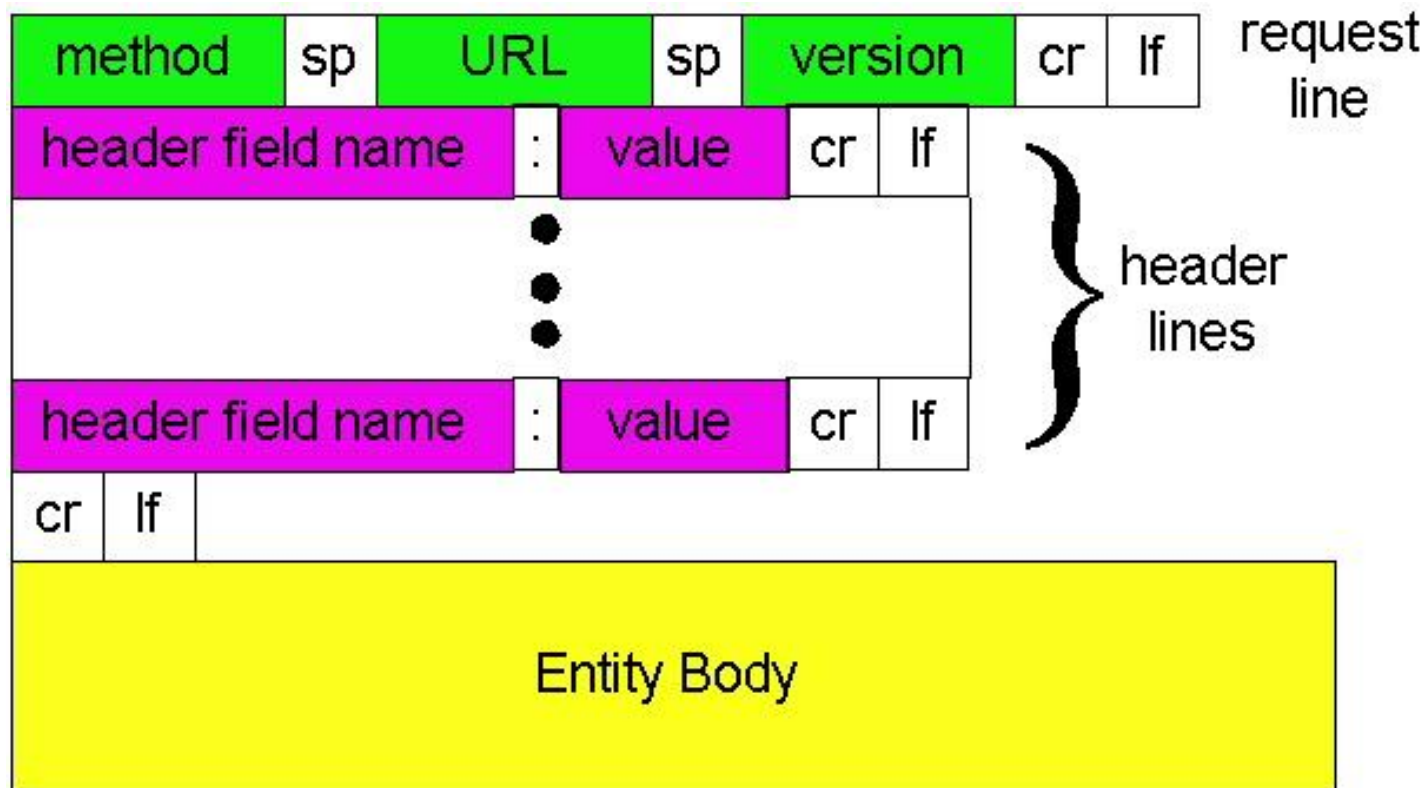
Carriage return,  
line feed  
indicates end  
of message

(extra carriage return, line feed)



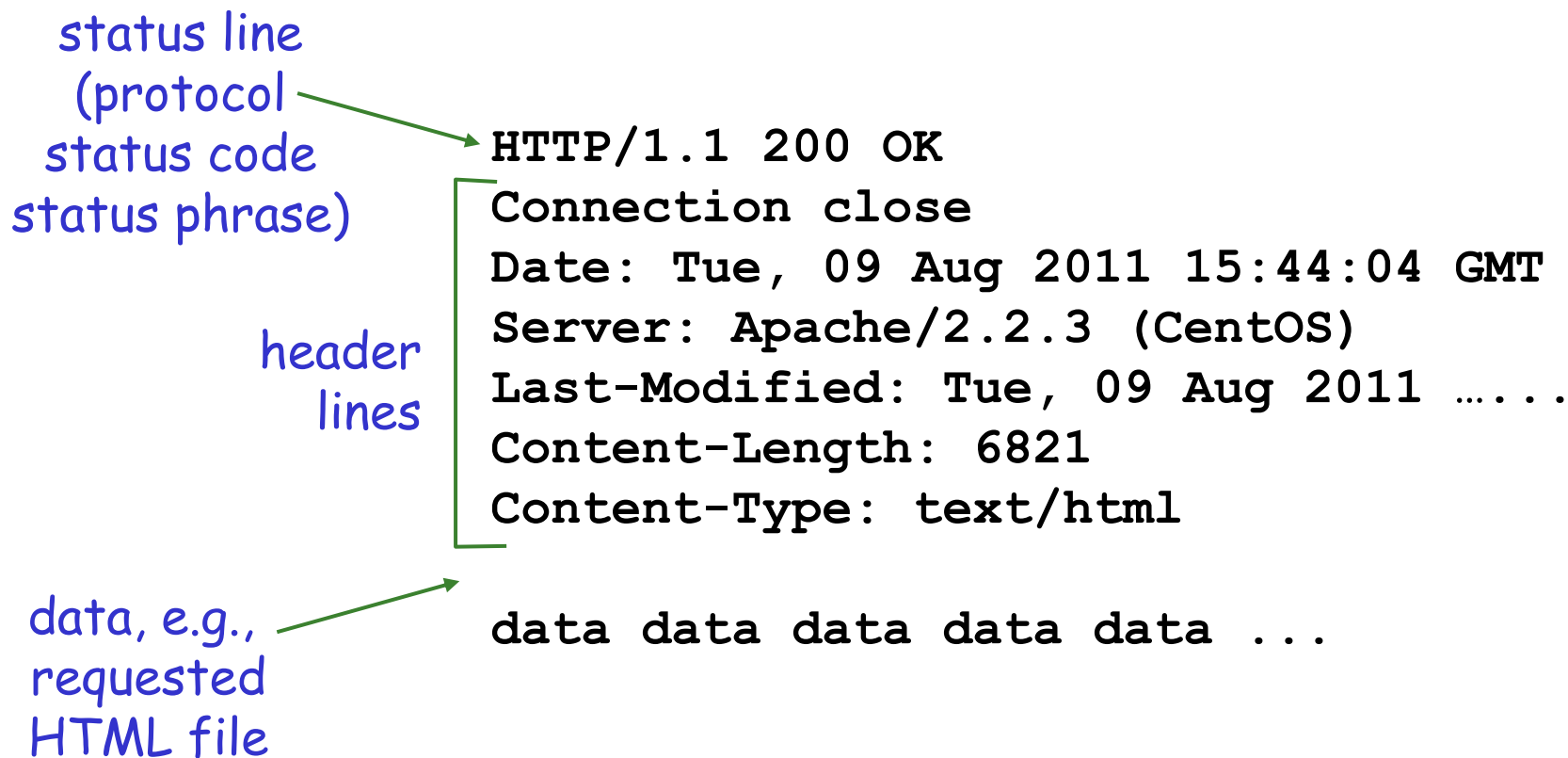


# HTTP request message: general format



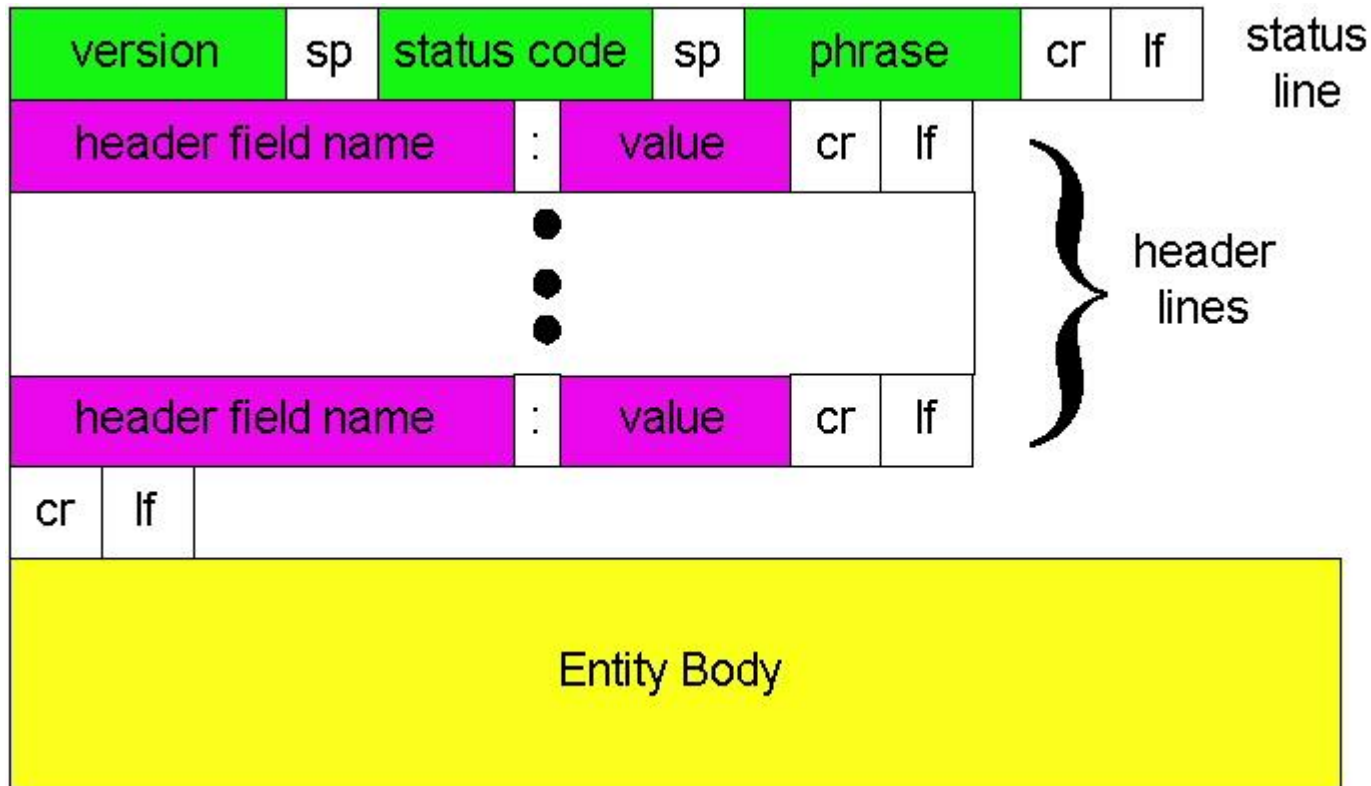


# HTTP response message



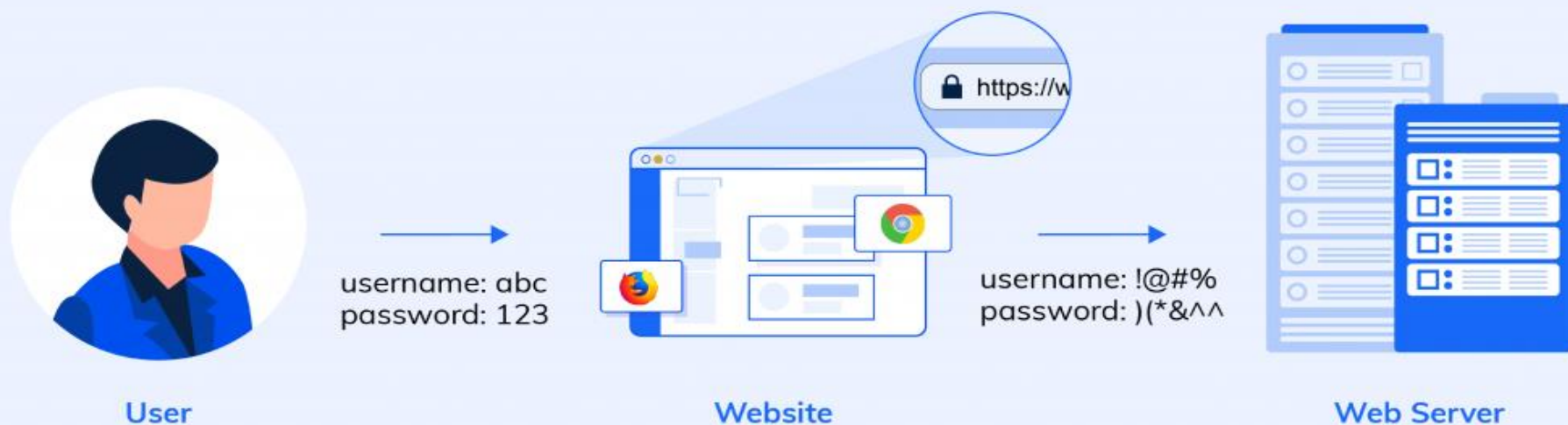


# HTTP response message: general format



# HTTP

## Giao thức HTTPS



**HTTPS (Hypertext Transfer Protocol Secure)** là giao thức truyền tải siêu văn bản an toàn. Thực chất, đây chính là giao thức HTTP nhưng tích hợp thêm Chứng chỉ bảo mật SSL nhằm mã hóa các thông điệp giao tiếp để tăng tính bảo mật.



# HTTP

- ❑ **1XX** - Thông tin: Yêu cầu được chấp nhận hoặc quá trình tiếp tục.
- ❑ **2XX** - Thành công: Xác nhận rằng hành động đã hoàn tất thành công hoặc đã được hiểu.
- ❑ **3XX** - Chuyển hướng: Client phải thực hiện hành động bổ sung để hoàn thành yêu cầu.
- ❑ **4XX** - Lỗi từ client chỉ ra rằng yêu cầu không thể hoàn thành hoặc chứa cú pháp sai.
- ❑ **5XX** - Lỗi từ phía máy chủ: Cho biết máy chủ không thể hoàn tất yêu cầu được cho là hợp lệ.



# HTTP

```
8 import http.client
9 connection = http.client.HTTPConnection("www.google.com")
10 connection.request("GET", "/")
11 response = connection.getresponse()
12 print(type(response))
13 print(response.status, response.reason)
14 if response.status == 200:
15     data = response.read()
16     print(data)
```

In [48]: runfile('D:/Google Drive/ml\_waf/untitled2.py', wdir='D:/Google Drive/ml\_waf')

<class 'http.client.HTTPResponse'>

200 OK

b'<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="vi"><head><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta content="images/branding/googlelog/1x/googlelog\_standard\_color\_128dp.png" itemprop="image"><title>Google</title><script nonce="ihF+Q+IVHp7jp5r8P7gQPg==">(function() {\nvar f,h=[];function k(a){for(var b;a&&!a.getAttribute("eid"));a=a.parentNode;return b||f}function l(a){for(var a.getAttribute("eid"));a=a.parentNode;return b}\nfunction m(a,b,c,d,g){var e="";c||-1!=b.search("&ei=")||\n(e="&ei="+k(d),-1==b.search("&lei=")&&(d=l(d))&&(e+="&lei="+d));d="";!c&&window.\_csid&&-1==b.search("&csid=")&&"slh"!a&&(d="&csid="+window.\_csid\n+"?atyp=i&ct="+a+"&cad="+b+"&zx="+Date.now()+d)/^http:/i.test(c)&&"https:"==window.location.protocol&&(google.ml&&google.ml(Error("a"),!1,{src:c,cj:f=google.kEI;google.getEI=k;google.getLEI=l;google.ml=function(){return null};google.log=function(a,b,c,d,g){if(c=m(a,b,c,d,g)){a=new Image;var



# Lấy header HTTP request và

```
8 import urllib.request
9 from urllib.request import Request
10 url="http://python.org"
11 USER_AGENT = 'Mozilla/5.0 (Linux; Android 10) AppleWebKit/537.36
12 def chrome_user_agent():
13     opener = urllib.request.build_opener()
14     opener.addheaders = [('User-agent', USER_AGENT)]
15     urllib.request.install_opener(opener)
16     response = urllib.request.urlopen(url)
17     print("Response headers")
18     print("-----")
19     for header,value in response.getheaders():
20         print(header + ":" + value)
21     request = Request(url)
22     request.add_header('User-agent', USER_AGENT)
23     print("\nRequest headers")
24     print("-----")
25     for header,value in request.header_items():
26         print(header + ":" + value)
27 if __name__ == '__main__':
28     chrome_user_agent()
```

In [49]: runfile('D:/Google Drive/ml\_waf/untitled2.py', wdir='D:/Google Drive/ml\_waf')

Response headers

-----

Connection:close

Content-Length:50666

Server:nginx

Content-Type:text/html; charset=utf-8

X-Frame-Options:DENY

Via:1.1 vegur, 1.1 varnish, 1.1 varnish

Accept-Ranges:bytes

Date:Wed, 19 May 2021 03:02:30 GMT

Age:1803

X-Served-By:cache-bwi5166-BWI, cache-hkg17930-HKG

X-Cache:HIT, HIT

X-Cache-Hits:1, 4816

X-Timer:S1621393351.831488,V50,V00

Vary:Cookie

Strict-Transport-Security:max-age=63072000; includeSubDomains

Request headers

-----

User-agent:Mozilla/5.0 (Linux; Android 10) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.210 Mobile Safari/537.36



# HTTP

## Schemes

HTTP ▼

## HTTP Methods Testing different HTTP verbs

**DELETE**

**/delete** "The request's DELETE parameters."

**GET**

**/get** The request's query parameters.

**PATCH**

**/patch** The request's PATCH parameters.

**POST**

**/post** The request's POST parameters.

**PUT**

**/put** The request's PUT parameters.

REST API và phương thức HTTP trong dịch vụ httpbin





# Tạo GET request với REST

```
8 import requests, json
9 response = requests.get("http://httpbin.org/get", timeout=5)
10 print("HTTP Status Code: " + str(response.status_code))
11 print(response.headers)
12 if response.status_code == 200:
13     results = response.json()
14     for result in results.items():
15         print(result)
16     print("Headers response: ")
17     for header, value in response.headers.items():
18         print(header, '-->', value)
19     print("Headers request : ")
20     for header, value in response.request.headers.items():
21         print(header, '-->', value)
22     print("Server:" + response.headers['Server'])
23 else:
24     print("Error code %s" % response.status_code)
```

HTTP Status Code: 200

Headers response:

Date --> Wed, 19 May 2021 03:14:09 GMT

Content-Type --> application/json

Content-Length --> 307

Connection --> keep-alive

Server --> gunicorn/19.9.0

Access-Control-Allow-Origin --> \*

Access-Control-Allow-Credentials --> true

Headers request :

User-Agent --> python-requests/2.21.0

Accept-Encoding --> gzip, deflate

Accept --> \*/\*

Connection --> keep-alive

Server:gunicorn/19.9.0



# SINH VIÊN TRÌNH BÀY

- A. Tìm hiểu module **requests**
- B. Tạo POST request với REST API
- C. Quản lý exception
- D. Xây dựng http client với https
- E. Tìm hiểu asyncio



# Cơ chế xác thực

Giao thức HTTP nguyên bản hỗ trợ:

## ☐ Xác thực cơ bản HTTP (HTTP Basic

**Authentication)**: Base64 dựa trên cơ chế xác thực cơ bản HTTP để mã hóa người dùng được tạo bằng mật khẩu sử dụng định dạng **user: password**.

## ☐ Xác thực thông báo HTTP (HTTP Digest

**Authentication)**: Cơ chế này sử dụng MD5 xác thực người dùng.

## ☐ Xác thực token HTTP (HTTP Bearer

**Authentication)**: Cơ chế này sử dụng xác thực dựa trên access\_token. Một trong những giao thức phổ biến nhất sử dụng loại xác thực này là OAuth.



# HTTP Basic Authentication

```
In [56]: runfile('D:/Google Drive/ml_waf/http_authen_basic .py', wdir='D:/Google Drive/ml_waf')
```

```
Enter username:thang310
```

```
Warning: QtConsole does not support password mode, the text you type will be visible.
```

```
Response.status_code:200
```

```
Login successful :
```

```
<!DOCTYPE html>
```

```
<html lang="en" data-color-mode="auto" data-light-theme="light" data-dark-theme="dark">
```

```
  <head>
```

```
    <meta charset="utf-8">
```

```
    <link rel="dns-prefetch" href="https://github.githubassets.com">
```

```
    <link rel="dns-prefetch" href="https://avatars.githubusercontent.com">
```

```
    <link rel="dns-prefetch" href="https://github-cloud.s3.amazonaws.com">
```

```
    <link rel="dns-prefetch" href="https://user-images.githubusercontent.com/">
```

```
    <link crossorigin="anonymous" media="all" integrity="sha512-LIOdgMPCoEgaBOBpRHSdzMIXC0BQl  
+3B20hYRhyoDICAcs6GTJ4jRrobBZhDZW04VGNzMZ56U9kllzZzrsc34Q==" rel="stylesheet" href="https://github.githubassets.co  
frameworks-2c839d80c3c2a0481a04e06944749dcc.css" />
```

```
    <link crossorigin="anonymous" media="all" integrity="sha512-n17DMvaA9F9fyjy3Yfe6zCrYjB6VN7TGAewNAAvt/  
V5Mw4cWluf9zpAWqF5cRBzGCEZTVI3f7Ppg975QDqRScg==" rel="stylesheet" href="https://github.githubassets.com/assets/  
behaviors-9f5ec332f680f45f5fca3cb761f7bacc.css" />
```



# HTTP Digest Authentication

```
8 import requests
9 from requests.auth import HTTPDigestAuth
10 from getpass import getpass
11 user=input("Enter user:")
12 password = getpass()
13 url = 'http://httpbin.org/digest-auth/auth/user/pass'
14 response = requests.get(url, auth=HTTPDigestAuth(user,password))
15 print("Headers request : ")
16 for header, value in response.request.headers.items():
17     print(header, '-->', value)
18 print('Response.status_code:'+ str(response.status_code))
19 if response.status_code == 200:
20     print('Login successful :'+str(response.json()))
21 print("Headers response: ")
22 for header, value in response.headers.items():
23     print(header, '-->', value)
```