

## Kỹ thuật lập trình

1

TS. Đặng Xuân Bảo

[dangxuanbao.att@gmail.com](mailto:dangxuanbao.att@gmail.com)

0964101882

11/01/2023

2

## Tổng quan môn học

- Môn học: Kỹ thuật lập trình
- Thời gian: 2TC (42 tiết: 18 LT, 24 TH)
- Hình thức thi: Thực hành
- Ngôn ngữ lập trình: Python

11/01/2023

3

## 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).

11/01/2023

4

## Nội dung cơ bản

Chương	Nội dung	Lý thuyết	Thực hành
1	Tổng quan về ngôn ngữ lập trình Python	9	6
2	Tấn công (Red Teams)	3	6
3	Phòng thủ (Blue Teams)	3	6
4	Một số vấn đề an toàn thông tin	3	6

11/01/2023

5

## Chương 1. 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

11/01/2023

6

## Ví dụ mở đầu

## ➤ Nhân 2 số lớn

➤ VD: 123456789123456789123456789123456789  
123456789123456789123456789123456789  
123456789 \* 1000000000

## Ngôn ngữ C++

- Dùng Dlist
- Quá nhiều dòng code

## Ngôn ngữ Python

```
>print
12345678912345678912345
6789123456789
12345678912345678912345
6789123456789 123456789
* 1000000000
```

11/01/2023

## 7 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 ký hiệu
    - Hướng tới sự đơn giản
      - ✓ Có while bỏ do while
      - ✓ Có elif bỏ switch - case
    - Ưu tiên cho việc đọc lại code
  - ❑ **Đa năng**
    - Lập trình web
    - Ứng dụng desktop, đồ họa, game
    - Lập trình cho điện thoại
    - Đặc biệt hiệu quả trong lập trình tính toán khoa học

11/01/2023

## 8 Giới thiệu Python

- ❑ **Đa biến hóa**
  - Cho phép sử dụng nhiều phương pháp lập trình
  - Kiểu động, kiểu mạnh,...
- ❑ **Python mạnh và nhanh**
  - Battery included
  - Viết mã nhanh với số lần gõ phím tối thiểu
- ❑ **Hòa hợp tốt với các ngôn ngữ khác**
  - Java → Jython
  - .NET → IronPython, Python for .NET
  - ...
- ❑ **Python chạy mọi nơi**
  - Unix
  - Windows
  - Mac
  - Nokia S60

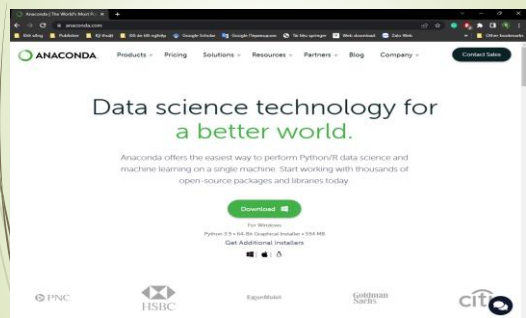
11/01/2023

## 9 Môi trường lập trình



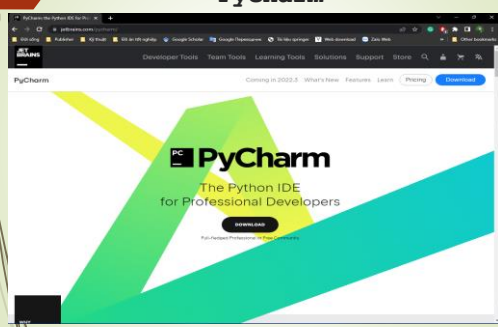
11/01/2023

## 10 Anaconda



11/01/2023

## 11 PyCharm



11/01/2023

## 12 Spyder



11/01/2023

13

## 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	3678	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

\*Không kể hoạt động thư viện trên github

11/01/2023

14

## Chương 1. 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

11/01/2023

15

## 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
```

3

11/01/2023

16

## 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 // 10
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
```

11/01/2023

17

## Lập trình cơ bản

## □ Kiểu dữ liệu

## ▪ Chuỗi: str

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

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

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

pancake

Formatted String % Insertion Tuple

```
>>> "aaaa$aaaa$aaa"%( "gcgcg", "tttt")
'aaaagcgcaaaattttaaa'
```

11/01/2023

18

## Lập trình cơ bản

## □ Kiểu dữ liệu

## ▪ Chuỗi

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
```

11/01/2023

19

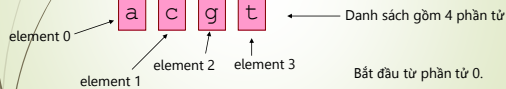
## Lập trình cơ bản

## Kiểu dữ liệu

## Kiểu danh sách: list

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

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



11/01/2023

20

## 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
```

11/01/2023

21

## Lập trình cơ bản

## Kiểu tuple

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

```
t=(12349)
q=(12349)
```

## 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

11/01/2023

22

## Lập trình cơ bản

```
>>> tlf = {"Michael" : 40062, \
"Binding" : 40064, "Andreas": 40063 }
>>> tlf.keys()
['Binding', '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
Binding 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

11/01/2023

23

## 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'])
```

11/01/2023

24

## 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
```

11/01/2023

25

## Vòng lặp

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

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

```
count = 0
while count < 5:
    print count, "la nho hon 5"
    count = count + 1
else:
    print count, "la khong nho hon 5"
```

```
0 la nho hon 5
1 la nho hon 5
2 la nho hon 5
3 la nho hon 5
4 la nho hon 5
5 la khong nho hon 5
```

11/01/2023

26

## Vòng lặp

11/01/2023

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

```
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>
```

```
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!
```

27

## 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.

11/01/2023

28

## Lập trình cơ bản

Nhớ thực 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

11/01/2023

29

## Hàm

## Cú pháp:

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

- ❑ Chồng hàm: không hỗ trợ
- ❑ Hàm inline: sử dụng từ khóa lambda
- ❑ Hàm với tham số có giá trị mặc định
- ❑ Hàm với danh sách đối số tùy ý

11/01/2023

30

## 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)
```

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

Khai báo hàm

Thân hàm

Trả về

Gọi hàm

11/01/2023

31

**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

11/01/2023

32

**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. 11/01/2023

33

**Chương 1. 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

11/01/2023

34

**Lớp**

Python là một ngôn ngữ lập trình hỗ trợ các cách tiếp cận lập trình khác nhau. Một trong những cách tiếp cận phổ biến để giải quyết vấn đề lập trình là tạo các đối tượng. Đây được gọi là Lập trình hướng đối tượng (OOP).

Một đối tượng bao gồm hai đặc điểm:

- ❑ Thuộc tính.
- ❑ Hành vi.

11/01/2023

35

**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()
```

11/01/2023

36

**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.

11/01/2023



37

## 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")

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))
```

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

11/01/2023

38

## 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")
```

11/01/2023

39

## 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()
```

11/01/2023

```
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()
```

40

## 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

11/01/2023

41

## 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)
```

11/01/2023

42

## Kế thừa

```
class lop_cha:
    Đoạn mã
class lop_con(lop_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')
```

11/01/2023

43

**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
```

11/01/2023

44

**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
```

11/01/2023

45

**Ghi đè 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(self):
        print("Đây là sinh viên ATT")
```

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

11/01/2023

46

**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)

11/01/2023

47

**Nạp chồng toán tử**

11/01/2023

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	$a != b$	<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>

48

**Chương 1. 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

11/01/2023



49

## 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
```

11/01/2023

50

## File

11/01/2023

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.

51

## File

11/01/2023

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.

52

## Ngoại lệ

11/01/2023

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")
```

53

## 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 ...")
```

11/01/2023

54

## Ngoại lệ

## try...else

```
1 try:
2     a = 1 / 3
3 except:
4     print("Sai!")
5 else:
6     b = 1 / 0
7     print(b)
```

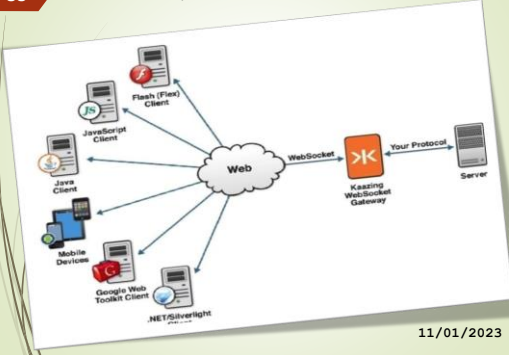
## try...finally

```
1 try:
2     a = 1 / 0
3 except:
4     print("Lỗi xảy ra")
5 finally:
6     print("Chương trình thực hiện xong")
```

11/01/2023

55

## Lập trình Socket



56

## Lập trình Socket

"A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to" - Oracle

11/01/2023

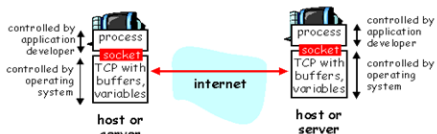
57

## Lập trình Socket

Socket-programming using TCP

**Socket:** a door between application process and end-end-transport protocol (TCP or UDP)

**TCP service:** reliable transfer of bytes from one process to another



58

## Phân loại Socket

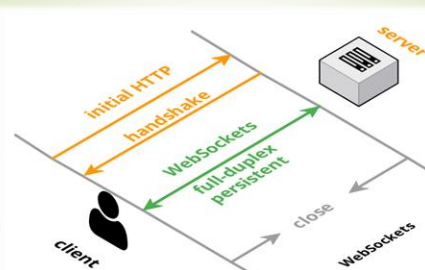
❑ **Web socket** là công nghệ hỗ trợ giao tiếp 2 chiều giữa **client** và **server** dựa trên một giao thức kết nối (thường là **TCP**) để tạo một kết nối hiệu quả và ít tốn kém.

❑ **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.

11/01/2023

59

## Web socket



60

## Cơ chế hoạt động Socket

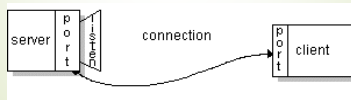
Đầu tiên client sẽ mở một kết nối TCP và cố gắng kết nối với server qua một PORT quy định.



61

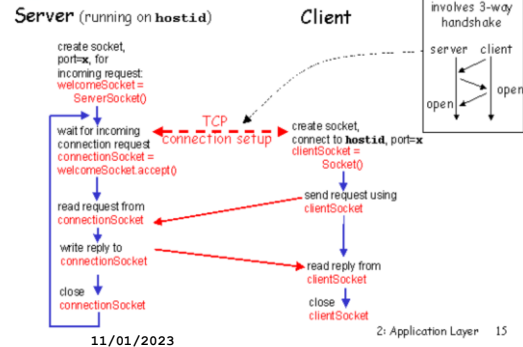
### Cơ chế hoạt động Socket

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.



11/01/2023

### Client/server socket interaction: TCP



11/01/2023

2: Application Layer 15

63

### Cơ chế hoạt động Socket

- ❑ Client thiết lập một kết nối đến server để giao tiếp và nó chỉ việc ngồi nghe (listening) mà không phải gửi request liên tục. Server thấy có dữ liệu mới sẽ kiểm tra xem trạng thái kết nối Keep-Alive, nếu kết nối còn nó sẽ gửi dữ liệu qua cho client.
- ❑ Socket sử dụng giao thức UDP, client và server sẽ không thiết lập kết nối, server chỉ cần biết địa chỉ client và đẩy dữ liệu đến.

11/01/2023

64

### Phân loại Web Socket

#### Stream socket:

Dựa trên giao thức TCP, việc truyền dữ liệu chỉ thực hiện khi client với server đã thiết lập kết nối. Stream socket còn gọi là socket hướng kết nối.

- ❑ Ưu điểm : Kết nối cần được xác định rõ ràng nên đảm bảo dữ liệu sẽ được truyền đi có tính tin cậy và toàn vẹn.
- ❑ Nhược điểm : Cần đợi client thiết lập kết nối.

11/01/2023

65

### Phân loại Web Socket

#### Datagram socket:

Dựa trên giao thức UDP, việc truyền dữ liệu không cần thiết lập kết nối. Còn gọi là socket không hướng kết nối.

- ❑ Ưu điểm : Do không cần thực hiện kết nối nên giảm thời gian trễ cho các quá trình xác thực, làm tăng tốc độ truyền dữ liệu.
- ❑ Nhược điểm : Không tin cậy và không đảm bảo dữ liệu toàn vẹn khi gửi tới client.

11/01/2023

66

### 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)
```

11/01/2023

67

## 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).

11/01/2023

68

## 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.

11/01/2023

69

## Socket module trong python

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Mar 24 20:27:42 2021
4
5 @author: chieu
6 """
7
8 import socket
9 socket.socket.
```

accept  
close  
detach  
dup  
family  
get\_inheritable  
makefile  
sendfile  
set\_inheritable

11/01/2023

70

## 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 dontri.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
```

11/01/2023

71

## Ứng dụng socket

11/01/2023

```
1 import socket
2 try:
3     print("gethostname:", socket.gethostname())
4     print("gethostname", socket.gethostname('www.actvn.edu.vn'))
5     print("gethostname_ex", socket.gethostname_ex('www.actvn.edu.vn'))
6     print("gethostname_ex", socket.gethostname_ex('103.21.148.154'))
7     print("getfqdn", socket.getfqdn('www.actvn.edu.vn'))
8     print("getaddrinfo", socket.getaddrinfo('www.actvn.edu.vn', None, 0, socket.SOCK_STREAM))
9 except socket.error as error:
10     print(str(error))
11     print("Connection error")

In [15]: runfile('D:/Google Drive/ml_waf/untitled0.py', wdir='D:/Google Drive/ml_waf')
gethostname: DLL310
gethostname 103.21.148.154
gethostname_ex ('actvn.edu.vn', ['www.actvn.edu.vn'], ['103.21.148.154'])
gethostname_ex ('WIN-KPBK3P3F1Q6', [], ['103.21.148.154'])
getfqdn WIN-KPBK3P3F1Q6
getaddrinfo [(AddressFamily.AF_INET: 2), (SocketKind.SOCK_STREAM: 1), 0, '', ('103.21.148.154', 0)]
```

72

## Socket TCP

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

11/01/2023

73

## 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()

```

11/01/2023

74

## Sinh viên chuẩn bị trước

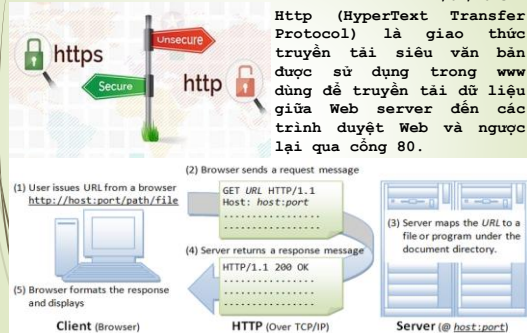
- ❑ Tìm hiểu module **requests**
- ❑ Tạo POST request với REST API
- ❑ Quản lý exception
- ❑ Xây dựng http client với https
- ❑ Tìm hiểu **asyncio**

11/01/2023  
\*Mỗi đề tài 1 sinh viên

75

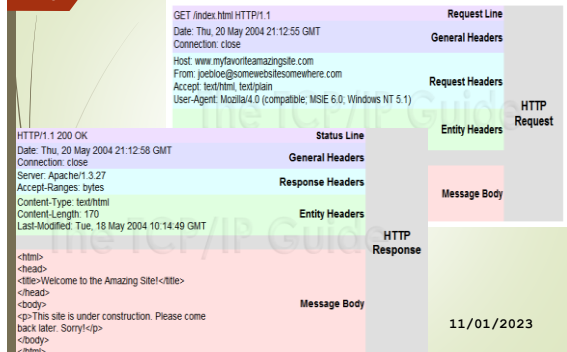
## HTTP

11/01/2023



76

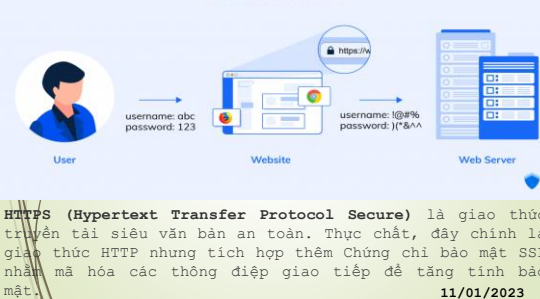
## HTTP



77

## HTTP

## Giao thức HTTPS



78

## 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.

\*<https://quantrimang.com/danh-sach-ma-trang-thai-http-http-status-code-day-du-149916>

11/01/2023







85

## HTTP Basic Authentication 11/01/2023

```
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-L10d9PCoEga800RmSdzHXCBQ1
+3828HVRy0DCAc560734JkR0882ND204V0N2R256U9K112zrsc34Q==" rel="stylesheet" href="https://github.githubassets.co
frameworks-3c839d80c3c2a841a8e4084278dccc.css" />

    <link crossorigin="anonymous" media="all" integrity="sha512-n17DnvaA9P9fy3Yf6bzCryj86Vn77GAewNAvt/
V5h4c4uff0p4uqfSc8b0GZTVI3f77pg975QdqG5g==" rel="stylesheet" href="https://github.githubassets.com/assets/
behaviors-9f5ec332f680f45f5fca3cb761f7bacc.css" />
```

86

## HTTP Digest Authentication 11/01/2023

```
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)
```

87

## Q&amp;A



11/01/2023