

# Bài 01: LẬP TRÌNH PYTHON CHO PHÂN TÍCH DỮ LIỆU VÀ HỌC MÁY (Lập trình Python cơ bản 1)

AI Academy Vietnam

# Nội dung bài 1

---

## 1. Tại sao phải học Python?

## 2. Cài đặt và thiết lập môi trường lập trình

## 3. Xây dựng ứng dụng đầu tiên (Hello world!)

- Sử dụng Jupyter Notebook cơ bản
- Vào/ra dữ liệu – Câu lệnh/khối lệnh/chú thích trong Python – Keywords

## 4. Một số loại lỗi thường gặp trong lập trình

- Syntax Errors – Runtime Errors – Logical Errors

## 5. Từ khóa và định danh

## 6. Biến và khai báo biến trong Python

## 7. Các kiểu dữ liệu cơ bản – chuyển đổi kiểu dữ liệu

- Kiểu số (int, float) - Kiểu chuỗi (str) - Kiểu danh sách (list) – Kiểu Boolean (bool)

# 1. Tại sao phải học Python?



# Why Python?

NEXT

The most popular programming languages in 2024













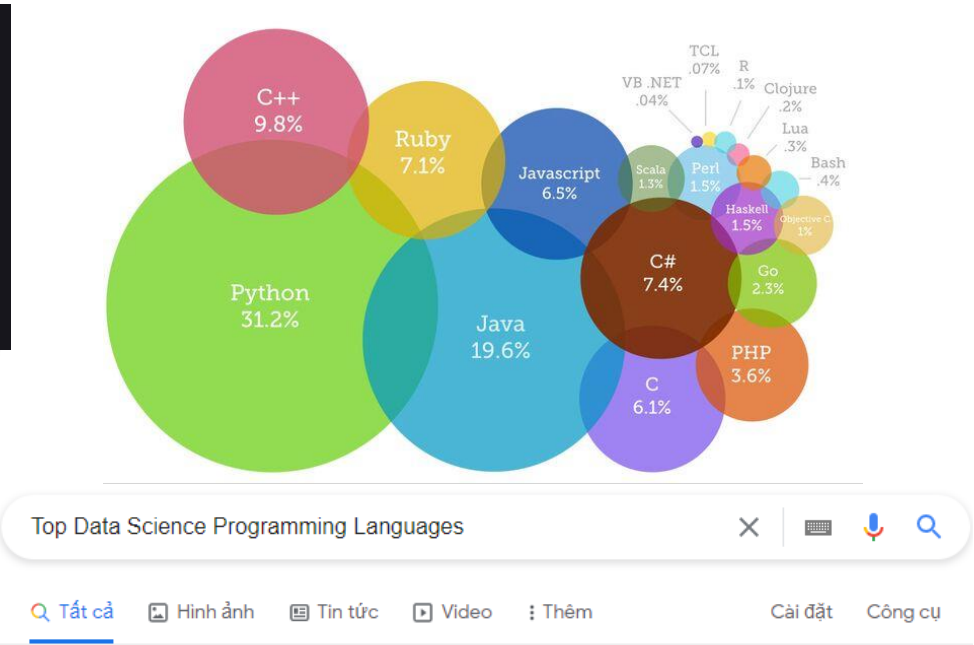








Rank	Language	Type	Score
1	Python ▾	  	100.0
2	Java ▾	  	95.3
3	C ▾	  	94.6
4	C++ ▾	  	87.0
5	JavaScript ▾		79.5
6	R ▾		78.6
7	Arduino ▾		73.2
8	Go ▾	 	73.1
9	Swift ▾	 	70.5
10	Matlab ▾		68.4



Top Data Science Programming Languages

Tất cả

Hình ảnh

Tin tức

Video

Thêm

Cài đặt

Công cụ

Khoảng 133.000.000 kết quả (0,65 giây)

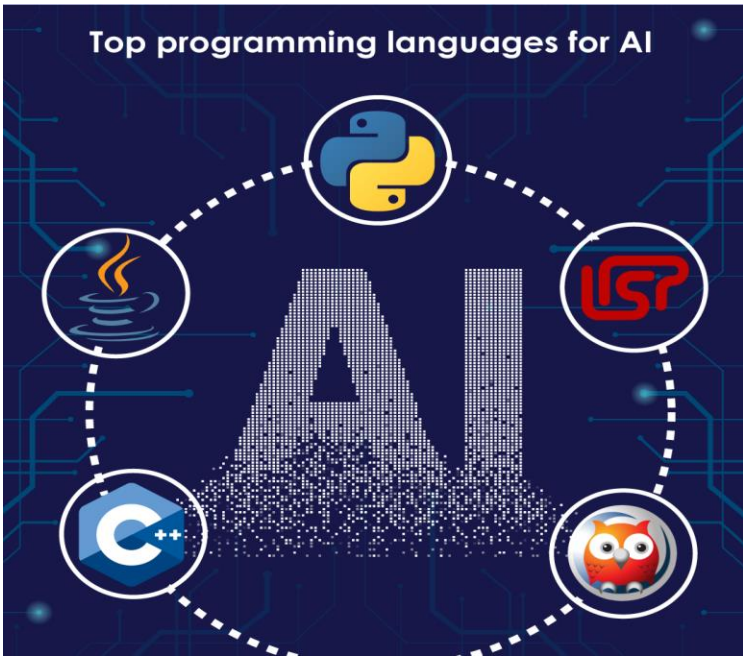
### Top programming languages for data science in 2021

1. Python. As discussed previously, Python has the **highest** popularity among **data scientists**. ...
2. JavaScript. JavaScript is the most popular **programming language** to learn. ...
3. Java. ...
4. R. ...
5. C/C++ ...
6. SQL. ...
7. MATLAB. ...
8. Scala.

Mục khác... • 4 thg 3, 2021

<https://flatironschool.com/blog/data-science-program...>

The 10 Best Data Science Programming Languages to Learn ...



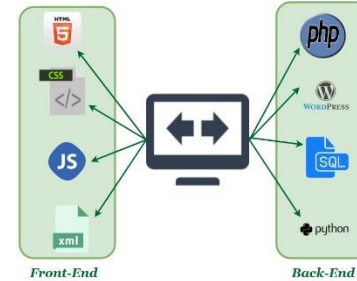
### Top 10 Programming Languages For Machine Learning

- Python
- R Programming
- JavaScript/Java
- Julia
- Lisp
- Scala
- C/C++
- TypeScript
- GO
- Shell

<https://www.spec-india.com/blog/programming-languages-for-machine-learning>

# Why python?

- Ngôn ngữ lập trình có thể sử dụng để phát triển từ các ứng dụng cho Desktop, Web đến các ứng dụng cho di động Mobi.
- Phù hợp nhất để phát triển các ứng dụng cho Backend web development (server side), phân tích dữ liệu, trí thông minh nhân tạo (AI), máy học (Machine learning), Dữ liệu lớn (Big data) và trực quan hóa dữ liệu (Data visualization).
- Nhiều lập trình viên sử dụng Python để phát triển Game và các ứng dụng trên Desktop.
- Phù hợp cho việc xây dựng các Script nhỏ để tự động hóa các công việc cụ thể, ví dụ như viết 1 đoạn script để đếm số lượng email có chứa các từ khóa cho trước.





# Why python?



# Why python?







**“DATA IS THE NEW GOLD”**

**Example**

# Uber Data Analysis



## 2. Từ khóa và định danh

# Từ khóa và định danh

- Mỗi ngôn ngữ lập trình có **từ khóa (keyword)** và **định danh (identifier)** khác nhau.
- Từ khóa trong Python là những từ chỉ dành riêng cho Python:

```
[help> keywords]
```

```
Here is a list of the Python keywords. Enter any keyword to get more help.
```

False	def	if	raise
None	del	import	return
True	elif	in	try
and	else	is	while
as	except	lambda	with
assert	finally	nonlocal	yield
break	for	not	
class	from	or	
continue	global	pass	

```
help> █
```

# Từ khóa và định danh

- Định danh là tên được đặt cho các thực thể như class, function, biến,... trong Python. Nó giúp phân biệt thực thể này với thực thể khác.
- **Quy tắc viết định danh:**
  - Định danh có thể là sự kết hợp giữa các chữ cái viết thường (từ a đến z) hoặc viết hoa (A đến Z) hoặc các số (từ 0 đến 9) hoặc dấu gạch dưới (\_).
  - Định danh không thể bắt đầu bằng một chữ số
  - Định danh phải khác các keyword.
  - Python không hỗ trợ các ký tự đặc biệt như !, @, #, \$, %, ... trong định danh.
  - Python là ngôn ngữ lập trình phân biệt chữ hoa, chữ thường

# Từ khóa và định danh

- **Một số quy ước thường dùng khi viết định danh:**
  - Tên lớp thường bắt đầu với một chữ cái hoa. Tất cả các định danh khác bắt đầu với chữ cái thường.
  - Định danh bắt đầu với một dấu gạch dưới `_` là định danh `protected` (chỉ lớp con mới có thể truy cập được).
  - Định danh bắt đầu với 2 dấu gạch dưới `__` là định danh `private` (chỉ lớp đó mới truy cập được).
  - Nếu định danh bắt đầu và kết thúc bằng 2 dấu gạch dưới (`__init__` chẳng hạn) thì định danh đó là tên đặc biệt được ngôn ngữ định nghĩa.
  - Nên đặt tên định danh có nghĩa

# **3. Biến và khai báo biến trong Python**



# Biến (VARIABLE)

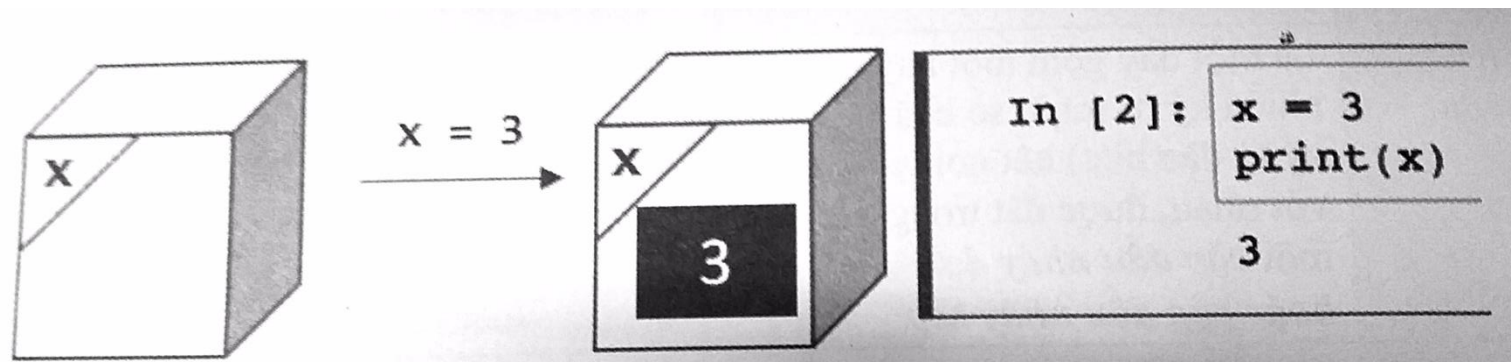
- Trong lập trình, biến (variable) là **tên của một vùng trong bộ nhớ RAM**, được sử dụng để **lưu trữ thông tin**.
- Biến giống như **một chiếc hộp** có thể giúp chúng ta lưu trữ dữ liệu. Ta có thể gán thông tin cho một biến, và có thể lấy thông tin đó ra để sử dụng.
- Khi một biến được khai báo, một vùng trong bộ nhớ sẽ dành cho các biến.
- Biến là một thứ **cực kì quan trọng trong lập trình** mà không thể thiếu trong bất cứ chương trình lớn, nhỏ nào.

```
In [3]: message = 'CHXH' #tạo biến message nhận giá trị là 'CHXH'  
n = 17 # tạo biến n nhận giá trị là 17  
pi = 3.1415926535897931 # tạo biến pi nhận giá trị là 3.14...
```

# Khai báo Biến (VARIABLE)

Cú pháp: **tên\_biến = <giá\_trị>**

```
In [3]: message = 'CHXH' #tạo biến message nhận giá trị là 'CHXH'  
n = 17 # tạo biến n nhận giá trị là 17  
pi = 3.1415926535897931 # tạo biến pi nhận giá trị là 3.14...
```



# Biến (VARIABLE)

- Trong Python không có câu lệnh khai báo biến tường minh
- Biến được tạo ra khi lần đầu tiên được gán giá trị

```
x = 5  
y = "John"
```

- Biến không cần được khai báo kiểu dữ liệu cụ thể và thậm chí có thể thay đổi kiểu dữ liệu

```
x = 4 # x is of type int  
x = "Sally" # x is now of type str
```

- Ta có thể dùng hàm `type()` để kiểm tra kiểu dữ liệu của biến
- Sau khi gán một giá trị cho biến trong Python, chúng ta có thể sử dụng biến để đại diện cho giá trị đó trong chương trình.

# Biến (VARIABLE)

---

- Gán giá trị cho nhiều biến:
  - Python cho phép gán giá trị đồng thời cho nhiều biến:

```
x, y, z = "Orange", "Banana", "Cherry"
```

- Python cho phép gán cùng một giá trị cho nhiều biến:

```
x = y = z = "Orange"
```

# Vai trò của Biến

- Biến giúp chúng ta lưu trữ các dữ liệu và cho phép chúng ta lấy các dữ liệu của chúng để tính toán được thuận tiện và chính xác hơn.
- Khi muốn lấy dữ liệu từ một nguồn không cố định từ ngoài chương trình (ví dụ: dữ liệu người dùng nhập vào, dữ liệu load từ file ...) ta cần sử dụng biến số để lưu thông tin của các dữ liệu này

```
In [4]: 52348252408 + 523482034
```

```
Out[4]: 52871734442
```

```
In [5]: 52871734442 + 545354645577
```

```
Out[5]: 598226380019
```

```
In [6]: a = 52348252408  
b = 523482034  
c = 545354645577  
d = a + b + c  
print(d)
```

```
598226380019
```

# Quy tắc đặt tên Biến

- Một số quy tắc bắt buộc khi đặt tên biến, nếu vi phạm sẽ bị báo lỗi:
  - Tên biến chỉ được chứa:
    - Các ký tự chữ cái ( $a \rightarrow z, A \rightarrow Z$ )
    - Các ký tự số ( $0 \rightarrow 9$ )
    - Dấu gạch dưới (`_`)
  - Tên biến không được bắt đầu bằng ký tự số
  - Phân biệt chữ hoa chữ thường ( $a \neq A$ )
  - Tên biến không được chứa các ký tự đặc biệt như “@,&...” hay các keywords trong Python.
  - Không được sử dụng dấu cách khi đặt tên biến
  - Tên biến nên được đặt theo một tên có ý nghĩa, có tính gợi nhớ.
  - Trường hợp tên biến gồm nhiều tiếng ghép lại, nên sử dụng dấu gạch nối.

```
In [9]: 76ab = 'asd'

File "<ipython-input-9-d8355cbf35f4>", line 1
       76ab = 'asd'
           ^
SyntaxError: invalid syntax
```

```
In [8]: them@ = 1000000

File "<ipython-input-8-0de45931b145>", line 1
       them@ = 1000000
           ^
SyntaxError: invalid syntax
```

```
In [10]: class = 'Advanced Theoretical Herpetology'

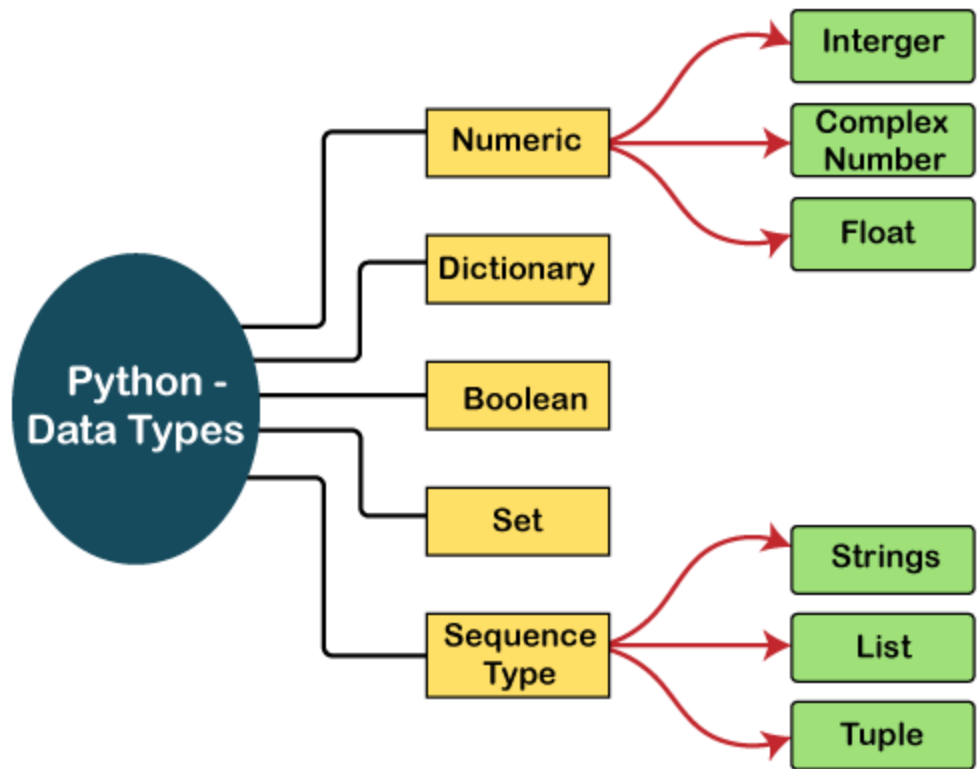
File "<ipython-input-10-9630266a2671>", line 1
       class = 'Advanced Theoretical Herpetology'
           ^
SyntaxError: invalid syntax
```



## **4. Các kiểu dữ liệu cơ bản trong Python**

# Kiểu dữ liệu trong Python

- Python hỗ trợ nhiều kiểu dữ liệu khác nhau



Type	Example
• Numeric: Integer, Float	<code>x = 10   x = 1.0</code>
• String	<code>x= 'Mike'</code>
• Boolean	<code>y = True   x = False</code>
• List	<code>my_list = [10, 20, 30]</code>
• Tuple	<code>my_tuple = ('Brett', 'Cisco', 'Cary', 2015)</code>
• Dictionary	<code>my_dict = {"one":1, "two":2}</code>
• Lists in Lists	<code>my_list2=[[10,20,30], ['Cisco Live', 'May', 2016]]</code>

# Kiểu dữ liệu trong Python

- Các biến sẽ được tự động nhận diện kiểu giá trị được gán

<code>x = "Hello World"</code>	str
<code>x = 20</code>	int
<code>x = 20.5</code>	float
<code>x = 1j</code>	complex
<code>x = ["apple", "banana", "cherry"]</code>	list
<code>x = ("apple", "banana", "cherry")</code>	tuple
<code>x = range(6)</code>	range
<code>x = {"name" : "John", "age" : 36}</code>	dict
<code>x = {"apple", "banana", "cherry"}</code>	set
<code>x = frozenset({"apple", "banana", "cherry"})</code>	frozenset
<code>x = True</code>	bool
<code>x = b"Hello"</code>	bytes
<code>x = bytearray(5)</code>	bytearray
<code>x = memoryview(bytes(5))</code>	memoryview

## 4.1 Kiểu dữ liệu số (int, float)

## a) Kiểu dữ liệu số

### Bao gồm 2 kiểu:

- Số nguyên – int (ví dụ: 1, 4, 111)
- Số thực – float (ví dụ: 1.1, 3.23, 11.01)

```
In [13]: a=11  
         b=12  
         c=a/b  
         print(c)  
  
0.9166666666666666
```

```
In [14]: a=11  
         b=12  
         c=int(a/b)  
         print(c)  
  
0
```

# Các phép toán với số

```
1  a = 10
2  b = 8
3  #-----
4  tong = a + b      # Tổng của hai số (+)
5  hieu = a - b      # Hiệu của hai số (-)
6  tich = a*b        # Tích của hai số (*)
7  thuong = a/b       # Thương của hai số (/)
8  thuong_nguyen = a//b # Phép chia lấy phần nguyên (//)
9  thuong_du = a % b  # Phép chia lấy phần dư (%)
10 mu = a**b         # Tính giá trị a lũy thừa b (**)
```

- **Học viên nhập code và đọc kết quả của các phép toán ở trên!**



# Kiểm tra kiểu dữ liệu của biến

- Biến trong Python rất linh hoạt, có thể chứa giá trị thuộc nhiều kiểu dữ liệu khác nhau.

Cú pháp: **type(biến)** sẽ giúp ta biết được kiểu dữ liệu của biến

```
1  #Kiểm tra kiểu dữ liệu của biến
2  x = 1985
3  y = 3.1415926535
4  z = 'AIAcademy Viet Nam'
5  n = [5, 6, 7, 8, 9]
6  b = True
7  #-----
8  print('Kiểu dữ liệu biến x:', type(x))
9  print('Kiểu dữ liệu biến y:', type(y))
10 print('Kiểu dữ liệu biến z:', type(z))
11 print('Kiểu dữ liệu biến n:', type(n))
12 print('Kiểu dữ liệu biến b:', type(b))
```

```
Kiểu dữ liệu biến x: <class 'int'>
Kiểu dữ liệu biến y: <class 'float'>
Kiểu dữ liệu biến z: <class 'str'>
Kiểu dữ liệu biến n: <class 'list'>
Kiểu dữ liệu biến b: <class 'bool'>
```

## 4.2 Kiểu dữ liệu chuỗi (str)

## b) Kiểu dữ liệu chuỗi

- Bên cạnh số, Python cũng có thể thao tác với chuỗi ký tự
- Chuỗi có thể được để trong dấu nháy đơn ('...') hoặc kép ("...")

```
1 #Khai báo chuỗi cách 1:
2 c1 = 'Python là ngôn ngữ lập trình số 1 hiện nay!'
3
4 #Khai báo chuỗi cách 2:
5 c2 = "Python là ngôn ngữ lập trình số 1 hiện nay!"
6
7 print(c1)
8 print(c2)
```

Python là ngôn ngữ lập trình số 1 hiện nay!  
Python là ngôn ngữ lập trình số 1 hiện nay!

```
1 #Kiểu dữ liệu số: a = 123
2 a = 123
3 print('Số: ', a)
4
5 #Kiểu dữ liệu chuỗi ký tự: A = '123'
6 A='123'
7 print('Chuỗi: ', A)
```

Số: 123  
Chuỗi: 123

- Cần **phân biệt** kỹ giữa **kiểu dữ liệu chuỗi và số**:

# Truy cập phần tử trong chuỗi

- Các chuỗi có thể được lập chỉ mục với ký tự đầu tiên được đánh số 0.
- Chỉ số cũng có thể là số âm, bắt đầu đếm từ bên phải
- Ngoài việc đánh số, thì cắt lát cũng được hỗ trợ. Trong khi index được sử dụng để lấy các ký tự riêng lẻ thì cắt lát sẽ được dùng để lấy chuỗi con.

```
+---+---+---+---+---+---+
| P | y | t | h | o | n |
+---+---+---+---+---+---+
0   1   2   3   4   5
      -5  -4  -3  -2  -1
```

```
In [4]: print(c[1])
```

u

```
In [5]: print(c[-1])
```

m

```
In [6]: print(c[0:3])
```

hum

# Nối chuỗi

- Sử dụng toán tử + để nối các chuỗi lại với nhau (lưu ý phải cùng kiểu str)

```
1 #Nối chuỗi:  
2 st = 'Thời đại ' + ' Hồ Chí Minh'  
3 st
```

'Thời đại Hồ Chí Minh'

```
1 #Không thể nối str + number  
2 st = 'Chủ tịch Hồ Chí Minh, sinh năm ' + 1890
```

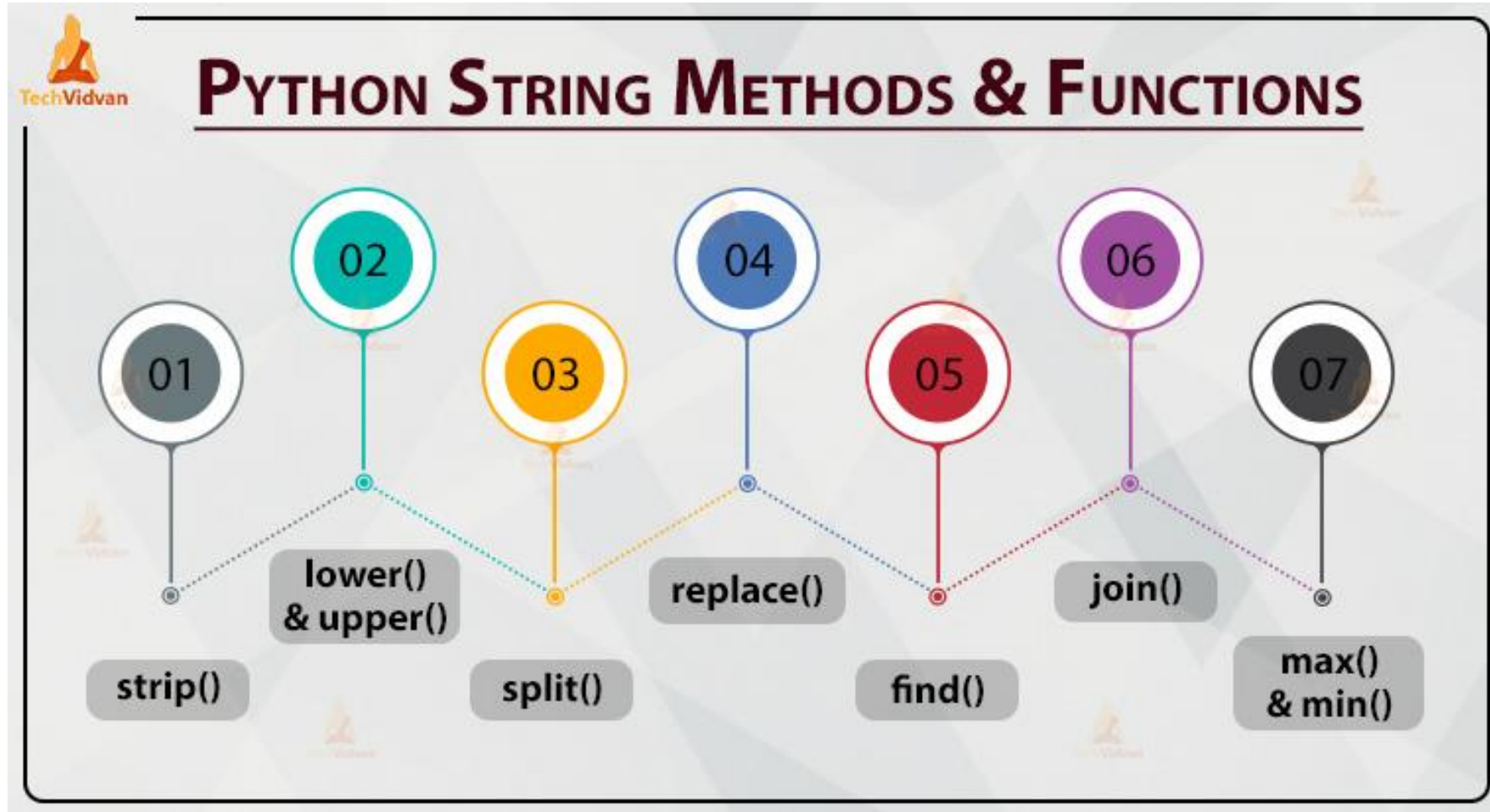
```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-23-4b2712bfc995> in <module>  
      1 #Không thể nối str + number  
----> 2 st = 'Chủ tịch Hồ Chí Minh, sinh năm ' + 1890
```

**TypeError:** can only concatenate str (not "int") to str

```
1 # Nhân chuỗi với 1 số nguyên  
2 st = 'ViệtNam '*3  
3 st
```

'ViệtNam ViệtNam ViệtNam '

# Một số thao tác với chuỗi





# Một số thao tác với chuỗi

- Cho biết số ký tự của chuỗi: **len(st)**

```
1 # 1.Cho biết số ký tự của chuỗi: len(st)
2
3 st = 'Python for Analytics'
4 print('Số ký tự trong chuỗi st là:', len(st))
```

Số ký tự trong chuỗi st là: 20

- Chuyển chuỗi st thành chữ hoa (upper) – chữ thường (lower): **st.upper() – st.lower()**

```
1 # 2.Chuyển chuỗi st thành chữ hoa (upper) – chữ thường (lower): st.upper() - st.lower()
2 print('Chữ hoa: ', st.upper())
3 print('Chữ hoa: ', st.lower())
```

Chữ hoa: PYTHON FOR ANALYTICS

Chữ hoa: python for analytics

# Một số thao tác với chuỗi (t)

- Loại bỏ khoảng trắng ở đầu và cuối chuỗi chuỗi: **st.strip()** – **st.lstrip()** – **st.rstrip()**

```
1 # 3. Loại bỏ khoảng trắng ở đầu và cuối chuỗi: st.strip()
2 st1 = "    Python for Analytics    "
3 st1.strip()
```

'Python for Analytics'

```
1 # Loại bỏ khoảng trắng ở đầu chuỗi:st.lstrip()
2 st1.lstrip()
```

'Python for Analytics '

```
1 # Loại bỏ khoảng trắng ở cuối chuỗi:st.rstrip()
2 st1.rstrip()
```

' Python for Analytics'

# Một số thao tác với chuỗi (t)

- Thay thế một chuỗi s thành s1 trong chuỗi st: **st.replace(s,s1)**

```
1 # 4. Thay thế một chuỗi s thành s1 trong chuỗi st: st.replace(s,s1)
2 print(st)
3 st_new = st.replace('Analytics', 'Machine Learning')
4 print(st_new)
```

Python for Analytics

Python for Machine Learning

- Để kiểm tra một chuỗi con có trong chuỗi st hay không? hãy dùng từ khóa **in**

```
1 # 5. Để kiểm tra một chuỗi con có trong chuỗi st hay không? hãy dùng từ khóa in
2 st_new = 'Python for Machine Learning'
3 x = 'Machine' in st_new      #Trả về True nếu có trong chuỗi
4 y = 'Analytics' in st_new    #Trả về False nếu ko có trong chuỗi
5
6 print('Kết quả kiểm tra chuỗi \'Machine\':',x )
7 print('Kết quả kiểm tra chuỗi \'Analytics\':',y )
```

Kết quả kiểm tra chuỗi 'Machine': True

Kết quả kiểm tra chuỗi 'Analytics': False

# Một số thao tác với chuỗi (t)

- Tách một chuỗi thành một danh sách (list): `string.split('separator')`

```
1 #6. Tách một chuỗi thành một danh sách (mảng 1 chiều): string.split('separator')
2 st_new = 'Python for Machine Learning'
3
4 #Mặc định tách chuỗi tại vị trí dấu cách
5 list_st = st_new.split()
6 print(list_st)
7 print(type(list_st))
```

```
['Python', 'for', 'Machine', 'Learning']
<class 'list'>
```

```
1 st_new1 = 'Đỗ Thị Hà,2001,Thanh Hóa'
2
3 #Tùy chọn vị trí tách theo tham số đưa vào
4 list_st1 = st_new1.split(',')
5
6 print(list_st1)
```

```
['Đỗ Thị Hà', '2001', 'Thanh Hóa']
```

# Một số thao tác với chuỗi (t)

- Tìm kiếm chuỗi con s trong chuỗi st: **st.find(s)**

```
1 #7. Tìm kiếm một con chuỗi trong chuỗi st
2 st_new = 'Python for Machine Learning'
3 print(st_new.find('for')) #Nếu tìm thấy trả về chỉ số nơi thấy chuỗi
4 print(st_new.find('HUMG')) #Nếu không tìm thấy trả về -1
```

7

-1

```
1 #8. Join chuỗi
2 st1 = ','.join(['Nam','Hoa','Thành','Thảo'])
3 print(st1)
4 st2 = '-'.join({'1985','2011','2015'})
5 print(st2)
```

Nam,Hoa,Thành,Thảo  
2011-2015-1985

```
1 #9. Max, Min: Ký tự có mã ASCII max, min
2 st = 'PythonforMachineLearning'
3 print(max(st))
4 print(min(st))
```

y  
L

# Phương thức kiểm tra

- **st.isalpha()** : Kiểm tra chuỗi ký tự st có phải chỉ chứa ký tự thuộc bảng chữ cái hay không (a-z, A-Z)?
- **st.isdigit()** : Kiểm tra chuỗi ký tự st có phải chỉ chứa ký tự chữ số hay không (0-9)?
- **st.isalnum()** : Kiểm tra chuỗi ký tự st có phải chỉ chứa ký tự chữ cái (a-z, A-Z) và ký tự số (0-9) hay không?
- **st.isupper()** : Kiểm tra chuỗi ký tự st có phải chỉ chứa ký tự chữ được viết hoa hay không (A-Z)?
- **st.islower()** : Kiểm tra chuỗi ký tự st có phải chỉ chứa ký tự chữ được viết hoa hay không (a-z)?

# Thực hành 1

# Bài 1: Chia kẹo cho học sinh lớp 1

Cô có  $n$  cái kẹo, có  $m$  học sinh trong lớp. Hãy giúp cô chia đều số kẹo cho tất cả học sinh.

- **Yêu cầu:**
  - Nhập vào số kẹo của cô –  $N$
  - Nhập vào số học sinh trong lớp –  $M$
- **Cho biết:**
  - Mỗi học sinh được nhận bao nhiêu cái kẹo?
  - Cô còn lại bao nhiêu cái kẹo?



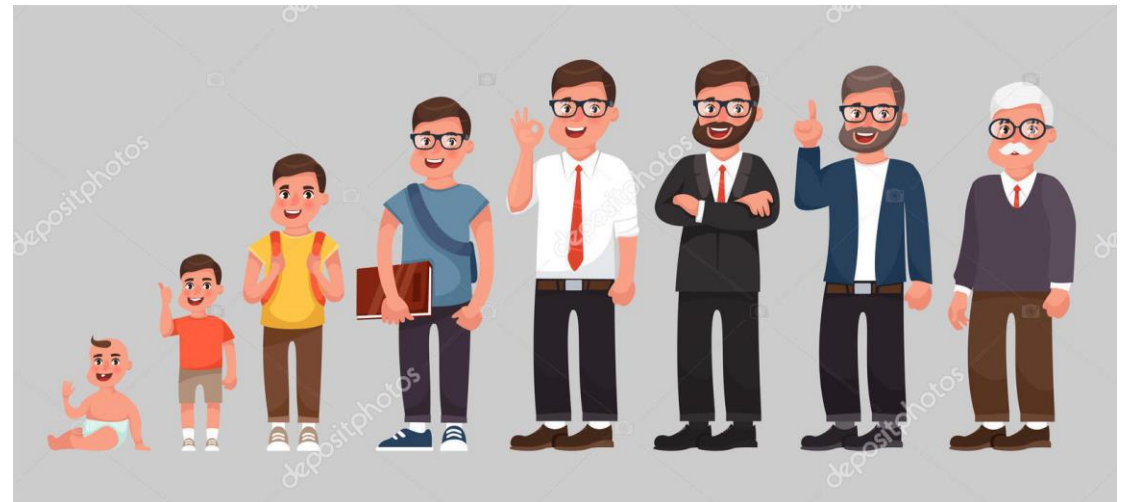


# Bài 2: Tính tuổi

Nhập vào tên và năm sinh của một người, tính tuổi của người đó và hiển thị ra màn hình thông báo

- Yêu cầu:
  - Nhập vào họ tên: mai phương thúy
  - Nhập vào năm sinh: 2000
- Hiển thị thông báo kết quả:

**Bạn “MAI PHƯƠNG THÚY” năm nay 19 tuổi!**



# Bài 3: Cho biết số thỏ trong rừng

Ban đầu trong rừng có một cặp thỏ, biết rằng cứ sau 1 tháng thì số lượng thỏ trong rừng tăng lên gấp đôi. Hỏi sau  $x$  tháng thì trong rừng có bao nhiêu con thỏ (Giả thiết các con thỏ không chết)

- Yêu cầu: Nhập vào số tháng  $x$ :
- Hiển thị thông báo cho biết số thỏ trong rừng sau  $x$  tháng:

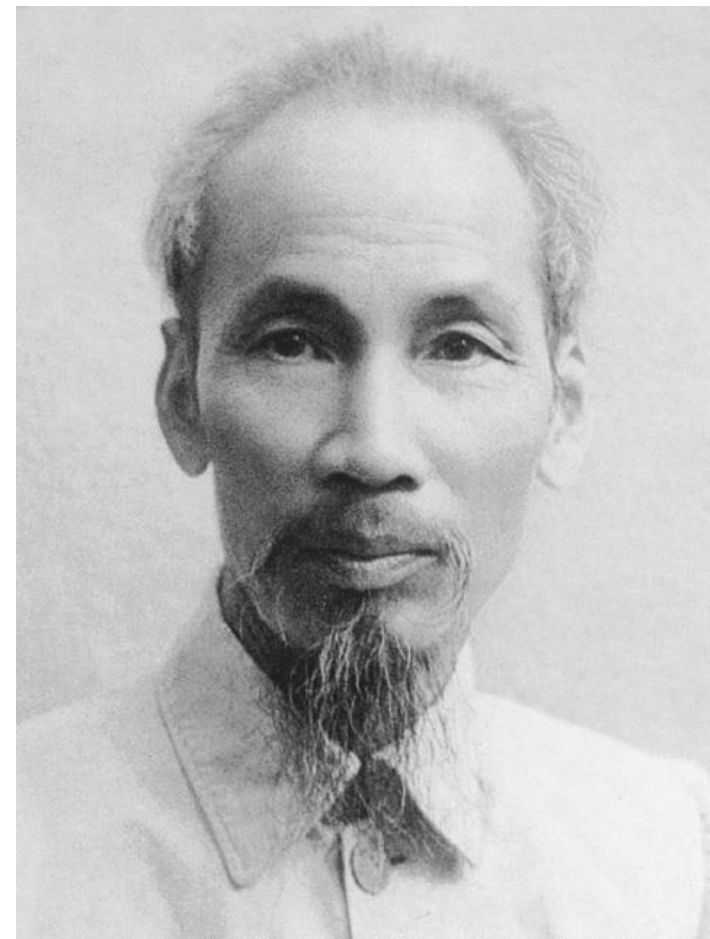
Sau  $x$  tháng số thỏ trong rừng là:  $y$  con



# Bài 4.1: Chuỗi văn bản

**“Nước Việt Nam là một, dân tộc Việt Nam là một. Sông có thể cạn núi có thể mòn, song chân lý ấy không bao giờ thay đổi. (HỒ CHÍ MINH, 1890 – 1969)”**

- **Yêu cầu:**
  - Cho biết số ký tự có trong đoạn văn trên.
  - Cho biết trong đoạn có chứa từ nào sau đây không (Không phân biệt chữ hoa, chữ thường)?
    - “hồ chí minh”
    - “non sông”
  - Tách đoạn văn thành các câu bởi dấu .
  - Cho biết trong đoạn văn trên có ký tự nào khác ký tự chữ và số hay không?
  - Thay thế các các từ ‘Việt Nam’ bằng ‘VIỆT NAM’ trong đoạn văn trên.



# Bài 4.2: Chuẩn hóa chuỗi

Chuỗi st ban đầu: `s = ' python FOR Machine LearNiNg '`

- Yêu cầu:

- Chuẩn hóa chuỗi về chuỗi s1: Loại bỏ hết khoảng trắng ở đầu và cuối chuỗi , thêm ký tự '.' ở cuối câu. chuyển hết về chữ thường và viết hoa ký tự đầu tiên trong mỗi từ → `'Python For Machine Learning.'`
- Chuẩn hóa chuỗi về chuỗi s2: Loại bỏ hết khoảng trắng ở đầu và cuối chuỗi, chuyển toàn bộ sang chữ hoa và nối các từ bằng dấu '-' → `'PYTHON-FOR-MACHINE-LEARNING'`

```
1 #Chuỗi ban đầu:  
2 s
```

```
' python FOR Machine LearNiNg '
```

```
1 #Chuỗi s1 sau khi chuẩn hóa:  
2 s1
```

```
'Python For Machine Learning!'
```

```
1 #Chuỗi s2 sau khi chuẩn hóa:  
2 s2
```

```
'PYTHON-FOR-MACHINE-LEARNING'
```

## 4.3 Kiểu dữ liệu danh sách (list)

## c) Kiểu dữ liệu danh sách (list)

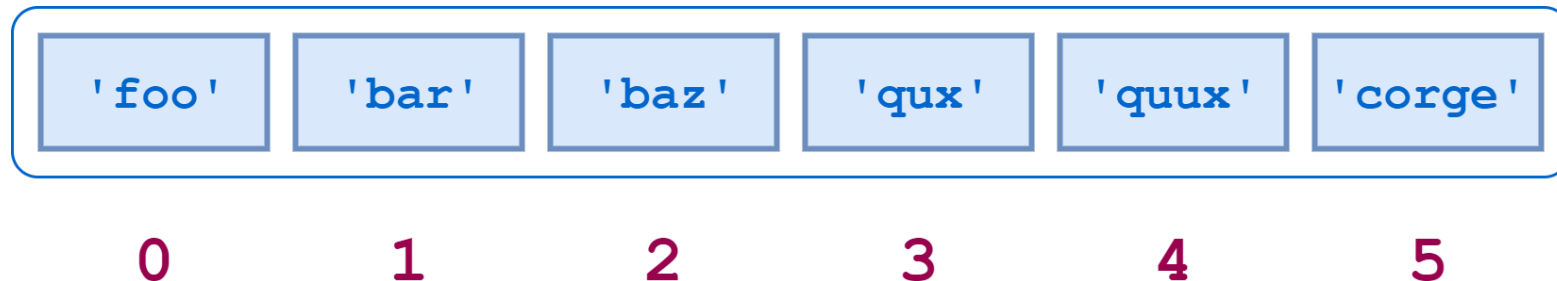
- Giả sử ta có nhu cầu lưu danh sách tên tất cả học sinh trong lớp học. Nếu lớp học ít:

```
1  #khai báo danh sách học sinh
2  hoc_sinh1 = 'Lê Thùy Dung'
3  hoc_sinh2 = 'Trần Đức Hùng'
4  hoc_sinh3 = 'Nguyễn Lan Anh'
```

- Nhưng nếu trong lớp có hàng chục, hàng trăm học sinh thì điều đó thật sự rất khó khăn, khó kiểm soát, quản lý.
- Kiểu dữ liệu danh sách (list) ra đời giúp chúng ta có thể khai báo, lưu trữ, truy xuất một dãy các đối tượng.

## c) Kiểu dữ liệu danh sách (list)

- List được sử dụng để lưu trữ dữ liệu của các loại dữ liệu khác nhau một cách tuần tự.
- Có các địa chỉ được gán cho mọi thành phần của danh sách, được gọi là Index.
- Giá trị chỉ mục bắt đầu từ 0 và tiếp tục cho đến khi phần tử cuối cùng được gọi là chỉ số dương.
- Ngoài ra còn có lập chỉ mục bắt đầu từ -1 cho phép bạn truy cập các phần tử từ cuối đến trước.



## c) Kiểu dữ liệu danh sách (list)

- Khai báo list:

**<tên\_danh\_sách> = [<Phần tử 1>,<Phần tử 2>,...,<Phần tử n>]**

```
1  #KHAI BÁO DANH SÁCH LIST
2  #khai báo danh sách học_sinh gồm các chuỗi tên của học sinh
3  hoc_sinh=['Lê Thùy Dung','Trần Đức Hùng','Nguyễn Lan Anh','Mai Phương Thúy']
4  print(hoc_sinh)
5
6  #Khai báo danh sách diem_chu gồm các chuỗi ký tự
7  diem = ['A+','A','B+','B','C+','C','D+','D','F']
8  print(diem)
9
10 #Khai báo danh sách gồm các số nguyên
11 list_so = [9,5,8,13,0,4,7,-9,11]
12 print(list_so)
13
14 #Khai báo danh sách với nhiều kiểu dữ liệu khác nhau
15 person_info = ['Mary',1998,'Tokyo, Japan', 1.70, 65]
16 print(person_info)
17
18 #Khai báo danh sách rỗng
19 list_rong = []
20 print(list_rong)
```

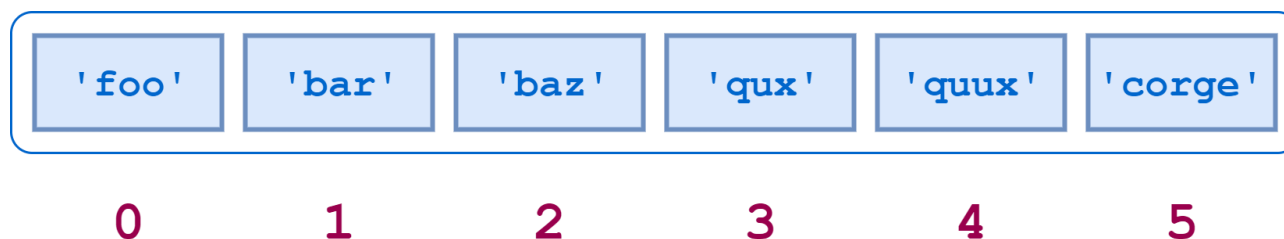
```
['Lê Thùy Dung', 'Trần Đức Hùng', 'Nguyễn Lan Anh', 'Mai Phương Thúy']
['A+', 'A', 'B+', 'B', 'C+', 'C', 'D+', 'D', 'F']
[9, 5, 8, 13, 0, 4, 7, -9, 11]
['Mary', 1998, 'Tokyo, Japan', 1.7, 65]
[]
```



# Truy xuất phần tử trong danh sách

- Để truy cập lấy giá trị tại một phần tử trong danh sách

**<tên\_danh\_sách> [<chỉ số>]**



```
1 #Truy xuất dữ liệu trong danh sách:
2 #Hiển thị tên học sinh ở vị trí thứ 3
3 print('Học sinh vị trí thứ 3: ', hoc_sinh[2])
4
5 #hiển thị tên người - chiều cao trong biến person_info
6 print('Họ tên: ', person_info[0], ' ---Chiều cao:', person_info[3])
7
8 #Truy xuất nhiều phần tử trong danh sách
9 print(list_so[3:8])
```

Học sinh vị trí thứ 3: Nguyễn Lan Anh  
Họ tên: Mary ---Chiều cao: 1.7  
[13, 0, 4, 7, -9]

# Thay đổi giá trị phần tử trong list

- Để thay đổi giá trị tại một phần tử trong danh sách:

**<tên\_danh\_sách> [<chỉ số>] = <giá trị mới>**

```
1 #Thay đổi giá trị của một phần tử trong danh sách
2 hoc_sinh=['Lê Thùy Dung','Trần Đức Hùng','Nguyễn Lan Anh','Mai Phương Thúy']
3 print('Ban đầu: ', hoc_sinh)
4 hoc_sinh[2] = 'Nguyễn Thị Lan Anh'
5 print('Thay đổi: ', hoc_sinh)
```

Ban đầu: ['Lê Thùy Dung', 'Trần Đức Hùng', 'Nguyễn Lan Anh', 'Mai Phương Thúy']

Thay đổi: ['Lê Thùy Dung', 'Trần Đức Hùng', 'Nguyễn Thị Lan Anh', 'Mai Phương Thúy']

```
1 list_so = [9,5,8,13,0,4,7,-9,11]
2 print('Ban đầu: ', list_so)
3
4 #Thay giá trị của phần tử cuối cùng trong list = 0
5 list_so[-1] = 0
6 print('Thay đổi:', list_so)
7
```

Ban đầu: [9, 5, 8, 13, 0, 4, 7, -9, 11]

Thay đổi: [9, 5, 8, 13, 0, 4, 7, -9, 0]

# Các thao tác khác với list

- **Nối hai danh sách (+):** kết quả trả về một danh sách mới.

```
1 list_a = [8,4,8,2]
2 list_b = [3,0,7,6,5]
3 #----Cộng hai danh sách (+)----:
4 list_ab = list_a + list_b
5 print('List mới: ', list_ab)
```

List mới: [8, 4, 8, 2, 3, 0, 7, 6, 5]

- **Lấy độ dài của danh sách len(list):** kết quả trả về một số nguyên là độ dài của danh sách

```
1 #Lấy độ dài của danh sách list_ab: len(list_ab)
2 print(list_ab, ' Có số phần tử là: ', len(list_ab))
```

[8, 4, 8, 2, 3, 0, 7, 6, 5] Có số phần tử là: 9

# Các thao tác khác với list (t)

- Kiểm tra một phần tử có thuộc danh sách hay không (in): kết quả trả về một danh sách mới.

```
1  #Kiểm tra phần tử có thuộc danh sách ko?  
2  print(list_ab)  
3  
4  #Kiểm tra phần tử 0:  
5  bol_0 = 0 in list_ab  
6  print('Phần tử 0 có thuộc list_ab ? ', bol_0)  
7  
8  #Kiểm tra phần tử 9:  
9  bol_9 = 9 in list_ab  
10 print('Phần tử 9 có thuộc list_ab ? ', bol_9)
```

[8, 4, 8, 2, 3, 0, 7, 6, 5]

Phần tử 0 có thuộc list\_ab ? True

Phần tử 9 có thuộc list\_ab ? False

# Các thao tác khác với list (t)

- **Thêm một phần tử vào danh sách:** có hai cách

- Thêm phần tử vào cuối danh sách:

**<danh\_sách>.append(<giá trị>)**

- Thêm phần tử vào vị trí bất kỳ trong danh sách:

**<danh\_sách>.insert(<vị trí>,<giá trị>)**

```
1 print('Danh sách ban đầu: \n', list_ab)
2
3 #Thêm phần tử vào cuối danh sách:
4 list_ab.append(10)
5 print('Danh sách thêm với append:', list_ab)
```

Danh sách ban đầu:

[8, 4, 8, 2, 3, 0, 8, 6, 5, 8]

Danh sách thêm với append: [8, 4, 8, 2, 3, 0, 8, 6, 5, 8, 10]

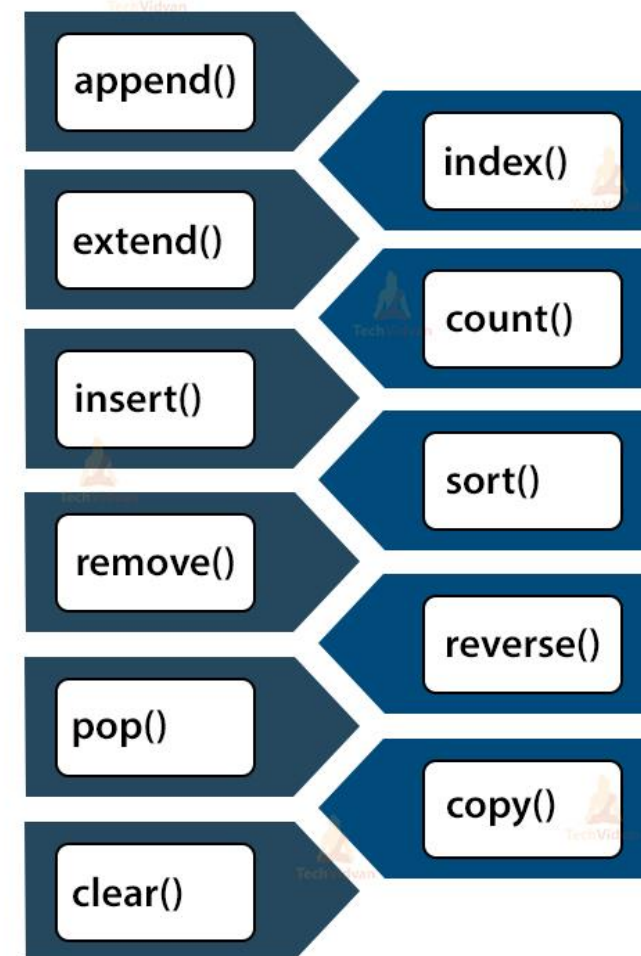
```
1 #thêm vào vị trí bất kỳ trong danh sách
2 list_ab.insert(1, 100)
3 print('Thêm phần tử với insert:', list_ab)
```

Thêm phần tử với insert: [8, 100, 4, 8, 2, 3, 0, 8, 6, 5, 8, 10]

```
1 list_ab.extend([7,8,0])
2 print('Thêm phần tử với extend:',list_ab)
```

Thêm phần tử với extend: [8, 100, 4, 8, 2, 3, 0, 8, 6, 5, 8, 10, 7, 8, 0]

## PYTHON LIST METHODS



# Các thao tác khác với list (t)

- **Xóa phần tử khỏi danh sách:**
  - Xóa tất cả các phần tử trong danh sách:  
`<danh_sách>.clear()`
  - Xóa phần tử tại vị trí cuối cùng trong danh sách:  
`<danh_sách>.pop()`
  - Xóa phần tử tại vị trí xác định trong danh sách:  
`del <danh_sách>[<vị trí>]`
  - Xóa phần tử có giá trị xuất hiện đầu tiên trong danh sách:  
`<danh_sách>.remove(<giá trị>)`

# Các thao tác khác với list (t)

```
1  #Xóa phần tử khỏi danh sách:
2  print('Danh sách ban đầu: \n', list_ab)
3
4  #Xóa phần tử cuối
5  list_ab.pop()
6  print('Danh sách xóa phần tử cuối:\n', list_ab)
7
8  #Xóa phần tử tại vị trí số 2
9  del list_ab[1]
10 print('Danh sách xóa phần tử ở vị trí số 2:\n', list_ab)
11
12 #Xóa phần tử có giá trị 0 xuất hiện đầu tiên
13 list_ab.remove(0)
14 print('Xóa phần tử có giá trị 0 đầu tiên:\n', list_ab)
```

Danh sách ban đầu:

[8, 100, 4, 8, 2, 3, 0, 7, 6, 5, 10]

Danh sách xóa phần tử cuối:

[8, 100, 4, 8, 2, 3, 0, 7, 6, 5]

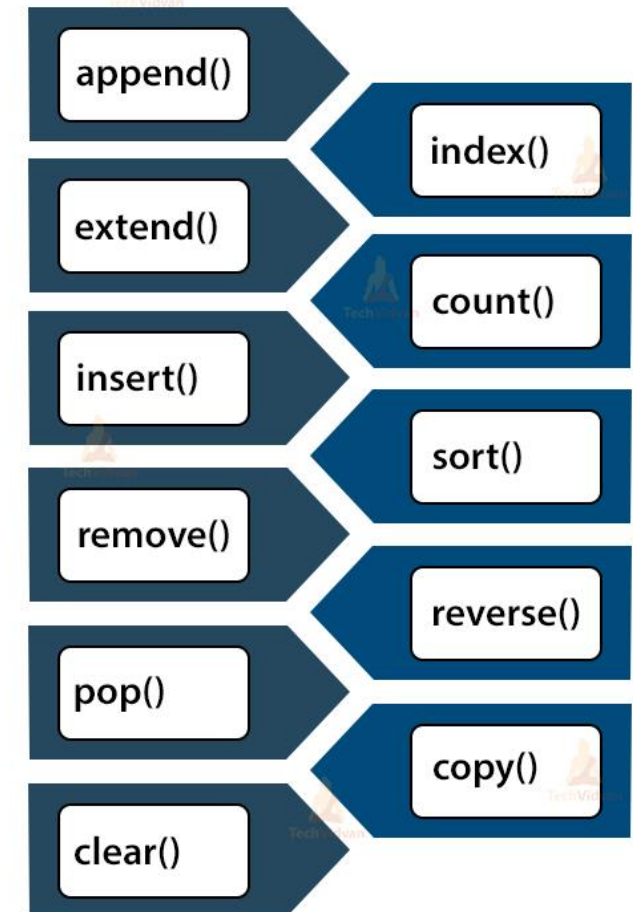
Danh sách xóa phần tử ở vị trí số 2:

[8, 4, 8, 2, 3, 0, 7, 6, 5]

Xóa phần tử có giá trị 0 đầu tiên:

[8, 4, 8, 2, 3, 7, 6, 5]

## PYTHON LIST METHODS



# Các thao tác khác với list (t)

- Đếm số lần xuất hiện của một phần tử trong danh sách:

**<danh\_sách>.count(<giá trị>)**

```
1  #Đếm số lần xuất hiện của một phần tử trong danh sách:  
2  print('Danh sách ban đầu: \n', list_ab)  
3  
4  #Số lần xuất hiện số 8 trong danh sách  
5  print(' Số 8 xuất hiện: ', list_ab.count(8))  
6  
7  #Số lần xuất hiện số 1 trong danh sách  
8  print(' Số 1 xuất hiện: ', list_ab.count(1))
```

Danh sách ban đầu:

[8, 4, 8, 2, 3, 7, 6, 5]

Số 8 xuất hiện: 2

Số 1 xuất hiện: 0



# Các thao tác khác với list (t)

- Sắp xếp các phần tử trong danh sách:

`<danh_sách>.sort()`

```
1 #Sắp xếp danh sách list_ab: Tăng dần
2 list_ab = [8, 4, 8, 2, 3, 0, 8, 6, 5, 8]
3 print('Danh sách ban đầu      : ',list_ab)
4
5 list_ab.sort() #Mặc định sắp xếp theo thứ tự tăng dần
6 print('Danh sách sắp xếp tăng dần:',list_ab)
```

Danh sách ban đầu : [8, 4, 8, 2, 3, 0, 8, 6, 5, 8]  
Danh sách sắp xếp tăng dần: [0, 2, 3, 4, 5, 6, 8, 8, 8, 8]

```
1 #Sắp xếp danh sách list_ab: Giảm dần
2 list_ab = [8, 4, 8, 2, 3, 0, 8, 6, 5, 8]
3 print('Danh sách ban đầu      : ',list_ab)
4
5 list_ab.sort(reverse=True)
6 print('Danh sách sắp xếp giảm dần:',list_ab)
```

Danh sách ban đầu : [8, 4, 8, 2, 3, 0, 8, 6, 5, 8]  
Danh sách sắp xếp giảm dần: [8, 8, 8, 8, 6, 5, 4, 3, 2, 0]

- Đảo ngược các phần tử trong danh sách:

`<danh_sách>.reverse()`

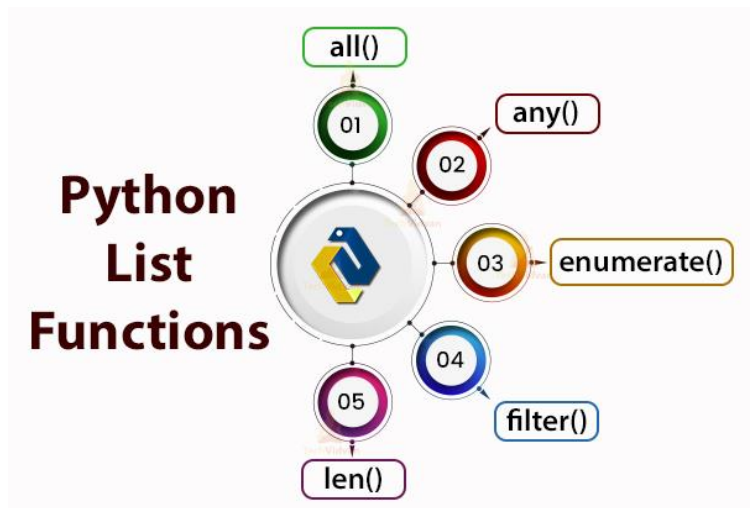
```
1 #Đảo ngược danh sách list_ab:
2 list_ab = [8, 4, 8, 2, 3, 0, 8, 6, 5, 8]
3 print('Danh sách ban đầu      : ',list_ab)
4
5 list_ab.reverse()
6 print('Danh sách sau khi đảo ngược:',list_ab)
```

Danh sách ban đầu : [8, 4, 8, 2, 3, 0, 8, 6, 5, 8]  
Danh sách sau khi đảo ngược: [8, 5, 6, 8, 0, 3, 2, 8, 4, 8]

# Các thao tác khác với list (t)

Ngoài ra, Python tích hợp sẵn một số hàm thường được sử dụng với list:

- **all():** Trả về giá trị True nếu tất cả các phần tử của list đều là true hoặc list rỗng.
- **any():** Trả về True khi bất kỳ phần tử nào trong list là true. Nếu list rỗng hàm trả về giá trị False.
- **enumerate():** Trả về đối tượng enumerate, chứa index và giá trị của tất cả các phần tử của list dưới dạng tuple.
- **list():** Chuyển đổi một đối tượng có thể lặp (tuple, string, set, dictionary) thành list.
- **max():** Trả về phần tử lớn nhất trong list.
- **min():** Trả về phần tử nhỏ nhất trong list.
- **sum():** Trả về tổng của tất cả các phần tử trong list.



# Các thao tác khác với list (t)

- Lưu ý về việc gán danh sách bởi 1 danh sách khác

```
1 #Trường hợp số, chuỗi
2 a = 10      #Khai báo biến a có giá trị =10
3 b = a      # Gán giá trị của biến a cho biến b
4 b = 5      # thay đổi giá trị của biến b, bằng giá trị mới
5 #-----
6 print('Giá trị của biến a: ', a)
7 print('Giá trị của biến b: ', b)
```

Giá trị của biến a: 10

Giá trị của biến b: 5

```
1 #Trường hợp danh sách:
2 ds_a = [4,5,8,9] #Khai báo danh sách ds_a
3 ds_b = ds_a      #Gán giá trị của biến ds_a cho ds_b
4 ds_b[1] = 10     #Thay đổi giá trị vị trí số 2 trong ds_b
5 #-----
6 print('Biến ds_a: ', ds_a)
7 print('Biến ds_b: ', ds_b)
```

Biến ds\_a: [4, 10, 8, 9]

Biến ds\_b: [4, 10, 8, 9]

Với mỗi sự thay đổi ở một trong 2 danh sách quản lý bởi biến ds\_a, ds\_b đều dẫn đến sự thay đổi ở danh sách còn lại.

# Các thao tác khác với list (t)

- Để tạo một danh sách độc lập với danh sách hiện có, các phần tử trong danh sách mới được sao chép giống hoàn toàn các phần tử trong danh sách hiện có:

**<danh sách>.copy()**

```
1  #Sao chép một danh sách độc lập:
2  ds_a = [4,5,8,9]   #Khai báo danh sách ds_a
3  ds_b = ds_a.copy() #Sao chép ds_a cho ds_b
4  ds_b[1] = 10       #Thay đổi giá trị vị trí số 2 trong ds_b
5  #-----
6  print('Biến ds_a: ', ds_a)
7  print('Biến ds_b: ', ds_b)
```

```
Biến ds_a: [4, 5, 8, 9]
Biến ds_b: [4, 10, 8, 9]
```

## 4.4 Kiểu dữ liệu Boolean (bool)

## d) Kiểu dữ liệu boolean

- Kiểu dữ liệu Boolean chỉ có hai giá trị True (đúng) và False (sai):

```
1  #Khai báo biến kiểu dữ liệu Boolean:
2  x = True
3  y = False
4  #Khai báo biến kiểu dữ liệu boolean qua biểu thức
5  z = 5>8
6  w = 12 == 12
7  #-----
8  print ('Kiểu dữ liệu của biến x:', type(x), ', Giá trị: ', x)
9  print ('Kiểu dữ liệu của biến y:', type(y), ', Giá trị: ', y)
10 print ('Kiểu dữ liệu của biến z:', type(z), ', Giá trị: ', z)
11 print ('Kiểu dữ liệu của biến w:', type(w), ', Giá trị: ', w)
```

```
Kiểu dữ liệu của biến x: <class 'bool'> , Giá trị: True
Kiểu dữ liệu của biến y: <class 'bool'> , Giá trị: False
Kiểu dữ liệu của biến z: <class 'bool'> , Giá trị: False
Kiểu dữ liệu của biến w: <class 'bool'> , Giá trị: True
```

# Thực hành 2

# Bài 5: Thống kê điểm Học viên

Khởi tạo một danh sách gồm các điểm thi môn “Python for Analysis” của lớp VIN\_01 (Điểm chữ: A, B, C, D, F)

- Yêu cầu:

- 1) Cho biết số sinh viên trong lớp
- 2) Có bao nhiêu sinh viên phải học lại môn này (điểm F).
- 3) Có bao nhiêu sinh viên có điểm từ B trở lên.
- 4) Sinh viên đầu tiên và cuối cùng trong lớp đã nghỉ học, tạo một bảng điểm mới và loại bỏ điểm của các sinh viên này ra khỏi danh sách điểm.



```
['C', 'B', 'D', 'A', 'F', 'A', 'B', 'F', 'B', 'B', 'C', 'A', 'D', 'F', 'B']
```

-----THỐNG KÊ-----

1. Tổng số học sinh trong lớp: 15
2. Số học sinh phải học lại là: 3
3. Số học sinh có điểm từ B trở lên là: 8
4. Bảng điểm mới: ['B', 'D', 'A', 'F', 'A', 'B', 'F', 'B', 'B', 'C', 'A', 'D', 'F']



## 5. Chuyển đổi kiểu dữ liệu

# Chuyển đổi kiểu dữ liệu

---

- Quá trình chuyển đổi giá trị của một kiểu dữ liệu (số nguyên, chuỗi, số float, v.v.) sang kiểu dữ liệu khác được gọi là chuyển đổi kiểu.
- Python có hai kiểu chuyển đổi kiểu.
  - Chuyển đổi kiểu ngầm định (tự động chuyển đổi kiểu dữ liệu thấp hơn (int) sang kiểu dữ liệu cao hơn (float) để tránh mất dữ liệu)
  - Chuyển đổi kiểu rõ ràng (người dùng chuyển đổi kiểu dữ liệu của một đối tượng thành kiểu dữ liệu bắt buộc)

# Chuyển đổi kiểu dữ liệu

---

- Python là một ngôn ngữ lập trình hướng đối tượng, điều đó có nghĩa là Python sử dụng các class để định nghĩa kiểu dữ liệu, do đó chuyển đổi dữ liệu trong Python có thể thực hiện dễ dàng khi sử dụng hàm khởi tạo (constructor):
  - `int()`: khởi tạo một số nguyên từ một số nguyên, một số thực (bằng cách làm tròn số đó xuống) hoặc một chuỗi (chuỗi đó đại diện cho một số)
  - `float()`: khởi tạo một số thực từ một số nguyên, một số thực hoặc một chuỗi (chuỗi đó đại diện cho một số nguyên hoặc một số thực)
  - `str()`: khởi tạo một chuỗi từ các kiểu dữ liệu bao gồm chuỗi, số nguyên và số thực


# Chuyển đổi kiểu dữ liệu

- **str(biến\_a)** : Chuyển kiểu dữ liệu của biến\_a về kiểu chuỗi
- **int(biến\_b)** : Chuyển kiểu dữ liệu của biến\_b về kiểu số nguyên
- **float(biến\_c)** : Chuyển kiểu dữ liệu của biến\_c về kiểu số thực
- **bool(biến\_d)** : Chuyển kiểu dữ liệu của biến\_d về kiểu bool
- **list(biến\_e)** : Chuyển kiểu dữ liệu của biến\_e về kiểu danh sách

```
x = int(1)           # x bằng 1
y = int(2.8)         # y bằng 2
z = int("3")         # z bằng 3
```

```
x = str("s1")       # x bằng 's1'
y = str(2)           # y bằng '2'
z = str(3.0)         # z bằng '3.0'
```

```
x = float(1)         # x bằng 1.0
y = float(2.8)        # y bằng 2.8
z = float("3")        # x bằng 3.0
w = float("4.2")      # w bằng 4.2
```



**Q & A**  
**Thank you!**