# Scripting Languages

Laszlo SZATHMARY
University of Debrecen
Faculty of Informatics

Lab #2

- string data type (cont.)
- list data type
- for loop

(last update: 2025-08-27 [yyyy-mm-dd])

2025-2026, 1st semester

## Standard data type

Standard data types in Python:

- number
- string
- list
- tuple

$\left.\begin{array}{l}\text{string}\\\text{list}\\\text{tuple}\end{array}\right\}$ common name: sequence

- dictionary
- set

## String formatting #1:

```python
def hello(name, what, color):
    # mario, the bus is red!
    print("{0}, the {1} is {2}!".format(name, what, color))
    # or
    print("{}, the {} is {}!".format(name, what, color))
    # or
    print("{n}, the {w} is {c}!".format(n=name.capitalize(), w=what, c=color))

def main():
    hello('mario', 'bus', 'red')
    print('-' * 22)
    hello('sara', 'sky', 'blue')

if __name__ == "__main__":
    main()
```

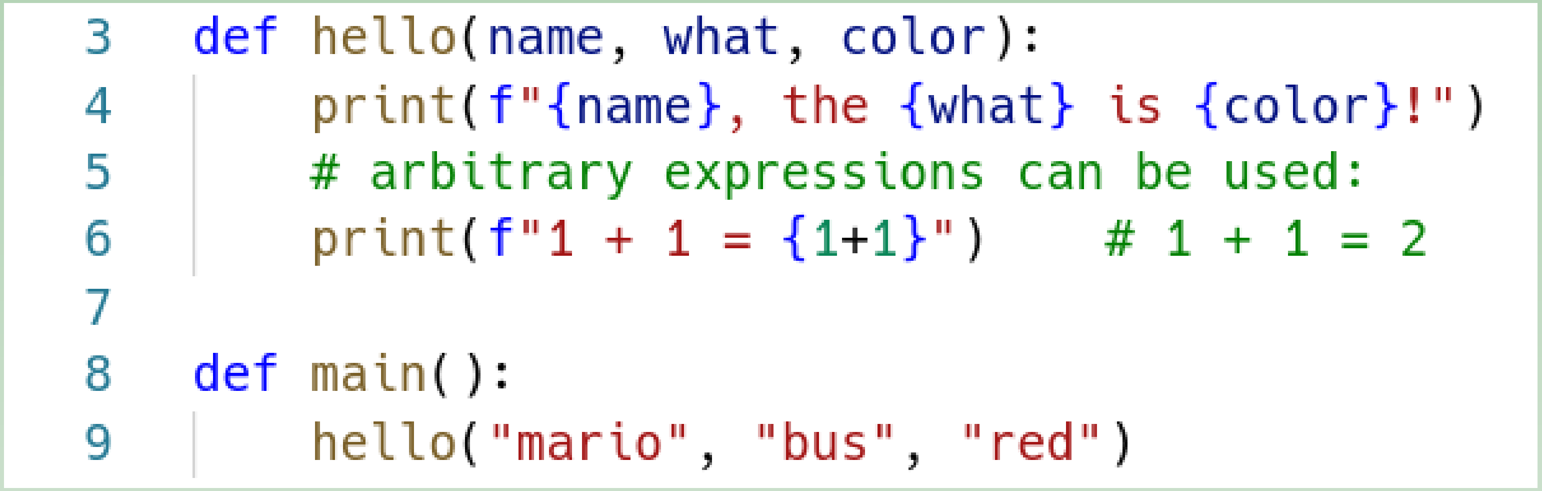
„constant”

common mistake:

```python
PI = 3.14159

# print('value of PI: ' + PI)      # bad
print('value of PI: ' + str(PI))  # better
print('value of PI:', PI)         # best
```

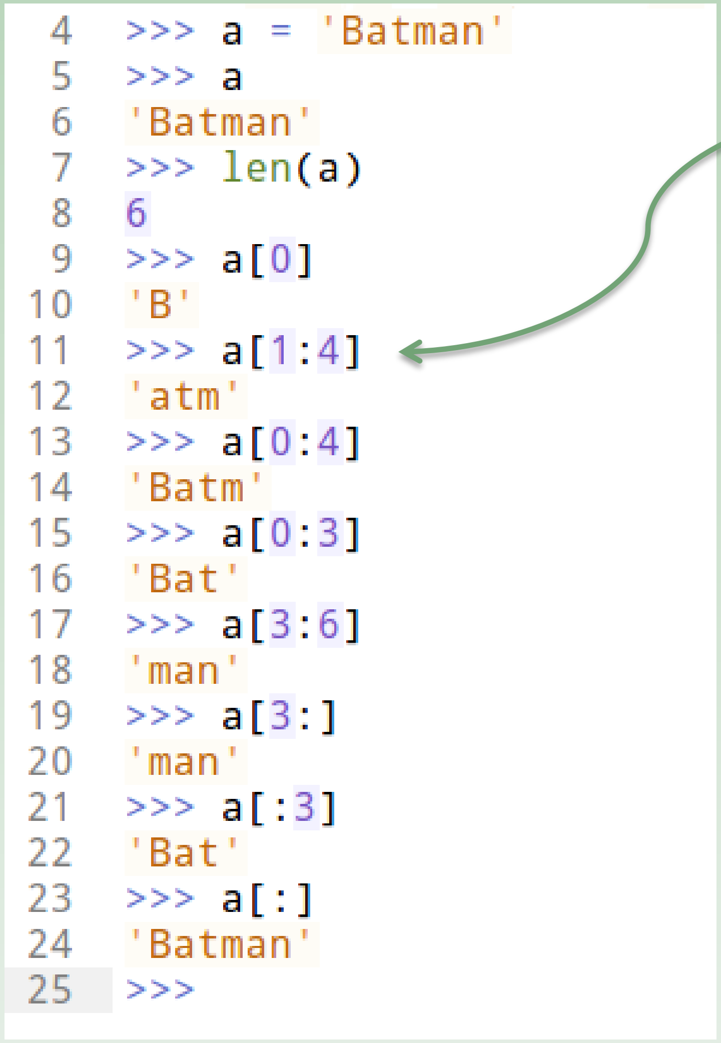
## String formatting #2:



```python
def hello(name, what, color):
    print(f"{name}, the {what} is {color}!")
    # arbitrary expressions can be used:
    print(f"1 + 1 = {1+1}")    # 1 + 1 = 2

def main():
    hello("mario", "bus", "red")
```

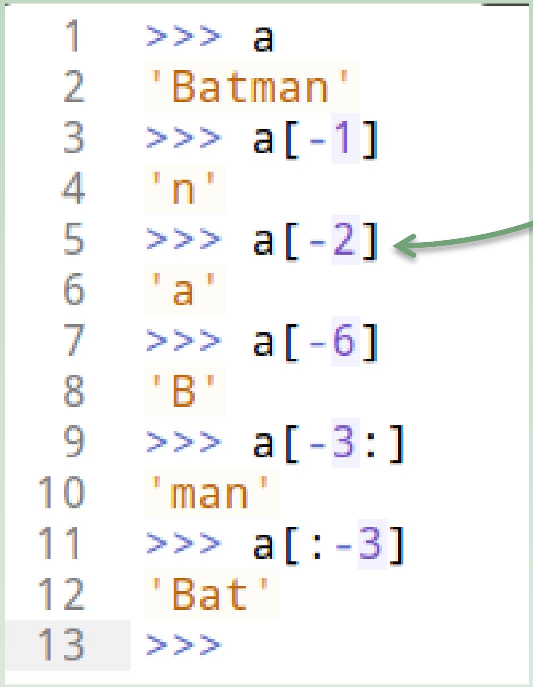
| B | a | t | m | a | n |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |



```
>>> a = 'Batman'
>>> a
'Batman'
>>> len(a)
6
>>> a[0]
'B'
>>> a[1:4]
'atm'
>>> a[0:4]
'Batm'
>>> a[0:3]
'Bat'
>>> a[3:6]
'man'
>>> a[3:]
'man'
>>> a[:3]
'Bat'
>>> a[:]
'Batman'
>>>
```

*slice*

| B | a | t | m | a | n |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| -6 | -5 | -4 | -3 | -2 | -1 |



```
>>> a
'Batman'
>>> a[-1]
'n'
>>> a[-2]
'a'
>>> a[-6]
'B'
>>> a[-3:]
'man'
>>> a[:-3]
'Bat'
>>>
```

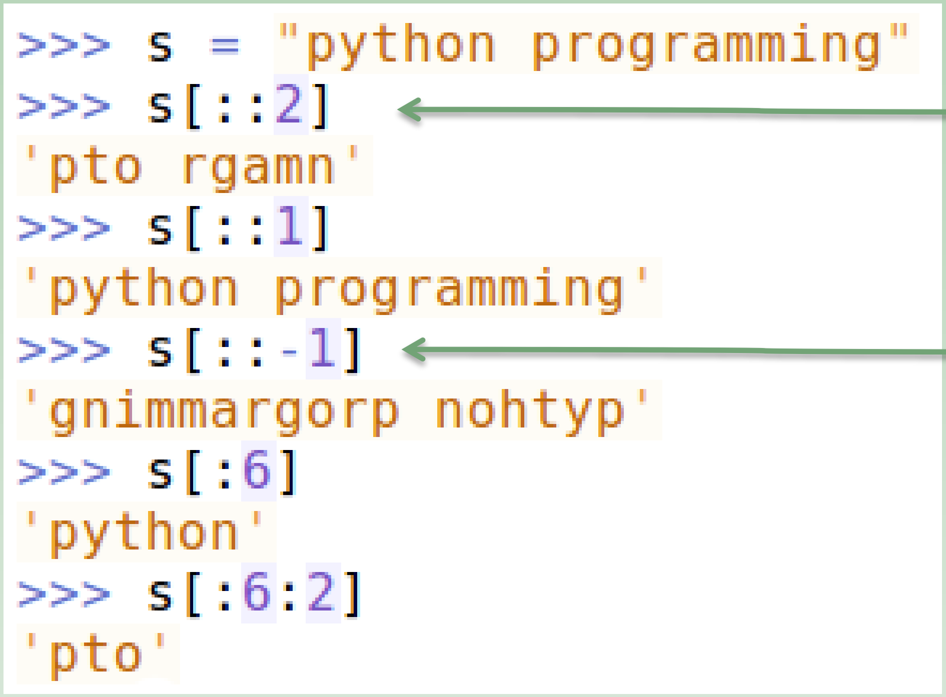
Negative indexing (from right to left).

Note:

`s[:n] + s[n:] == s`

(where `n` can be a positive or negative value)

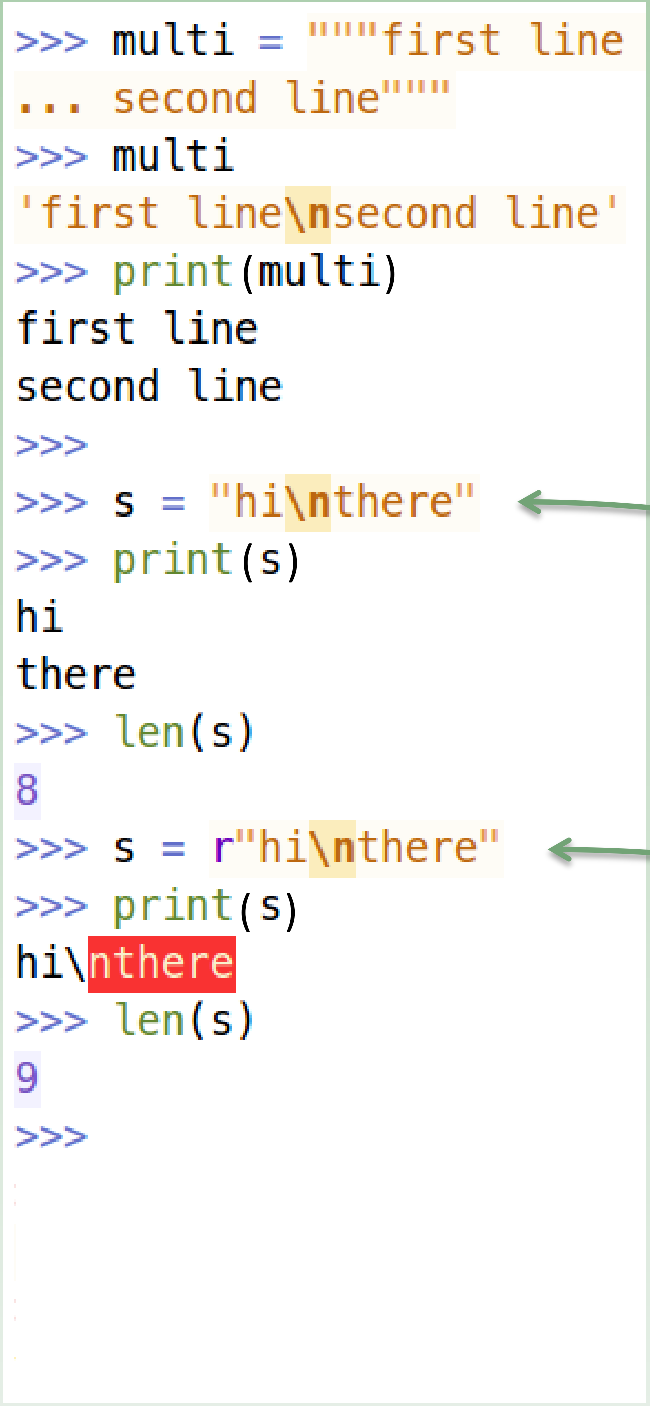
**HW:** complete `string1.py` . If you are ready, continue with `string2.py`.



```
>>> s = "python programming"
>>> s[::2]
'pto rgamn'
>>> s[::1]
'python programming'
>>> s[::-1]
'gnimmargorp nohtyp'
>>> s[:6]
'python'
>>> s[:6:2]
'pto'
```

step

reversing a string



```
>>> multi = """first line
... second line"""
>>> multi
'first line\nsecond line'
>>> print(multi)
first line
second line
>>>
>>> s = "hi\nthere"
>>> print(s)
hi
there
>>> len(s)
8
>>> s = r"hi\nthere"
>>> print(s)
hi\nthere
>>> len(s)
9
>>>
```

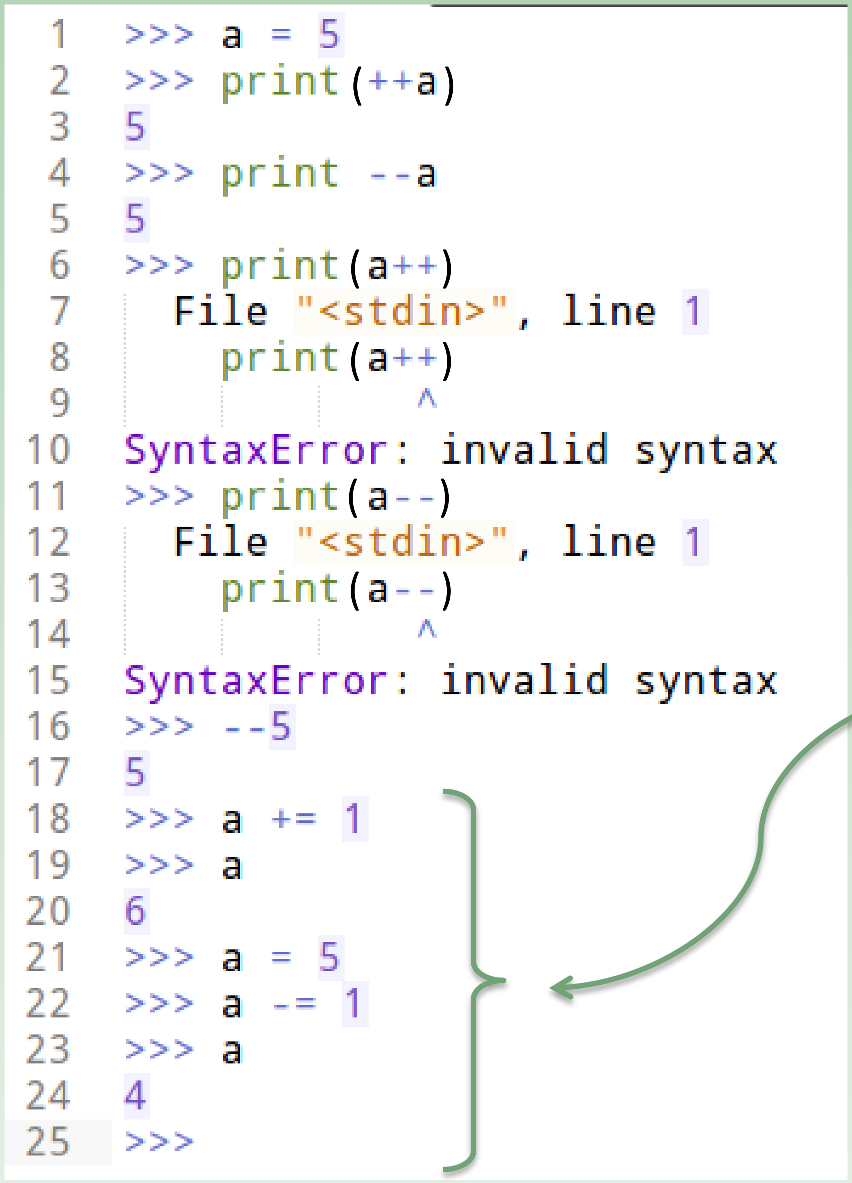
multiline string

normal string

*raw* string
(mainly used in
regular expressions)

In Python 3, every string
is a Unicode string (by default).

## Common mistake for beginners

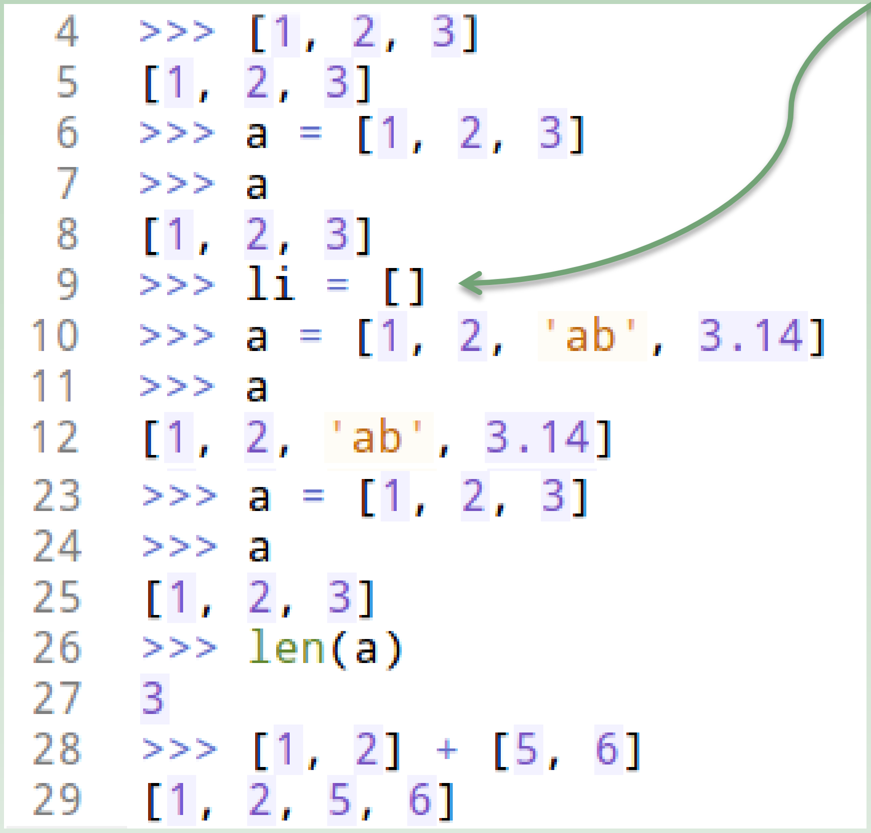


```
>>> a = 5
>>> print(++a)
5
>>> print --a
5
>>> print(a++)
  File "<stdin>", line 1
    print(a++)
            ^
SyntaxError: invalid syntax
>>> print(a--)
  File "<stdin>", line 1
    print(a--)
            ^
SyntaxError: invalid syntax
>>> --5
5
>>> a += 1
>>> a
6
>>> a = 5
>>> a -= 1
>>> a
4
>>>
```

The `+` and – are unary operators, i.e. `++5` means: `+(+5)`, whose value is `5`.
Similarly, `--5` means: `-(-5)`, which is also `5`…

For incrementing / decrementing, use the `+=` and `-=` operators.

# Lists

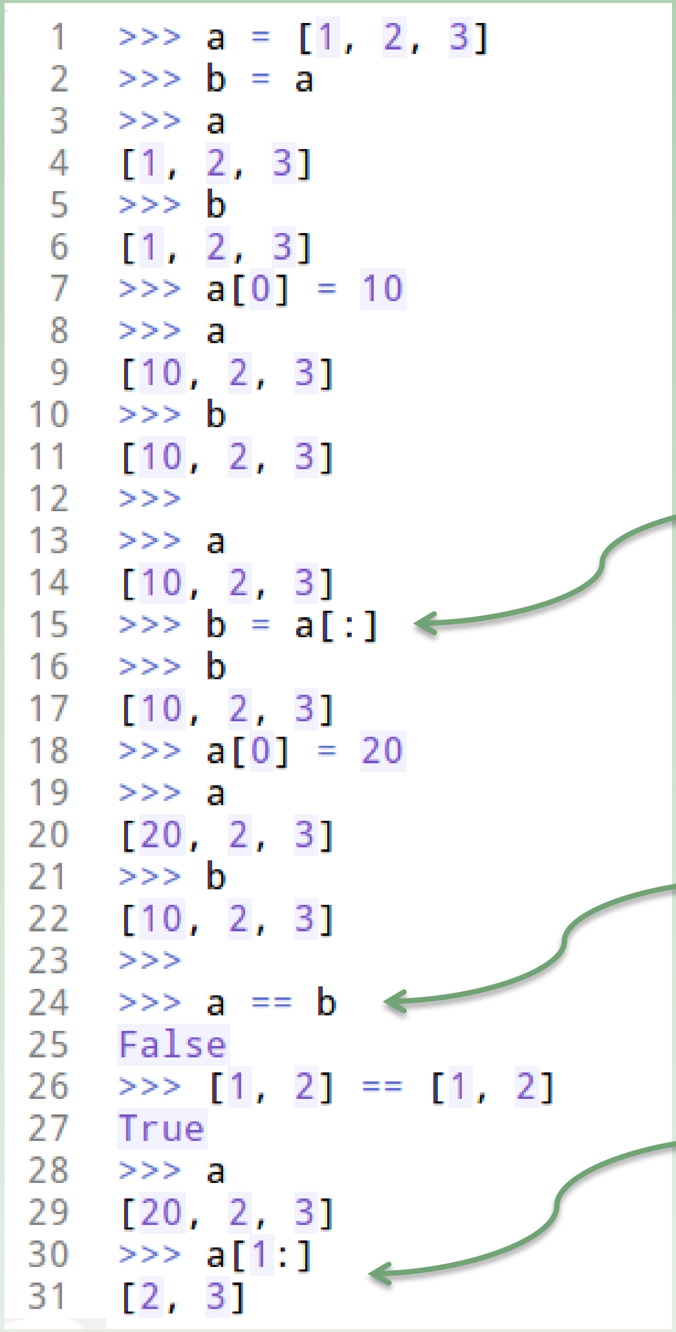


```
>>> [1, 2, 3]
[1, 2, 3]
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]
>>> li = []
>>> a = [1, 2, 'ab', 3.14]
>>> a
[1, 2, 'ab', 3.14]
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]
>>> len(a)
3
>>> [1, 2] + [5, 6]
[1, 2, 5, 6]
```

empty list

the majority of the operations that we saw at the strings also work here

$$\begin{tikzcd}
a \arrow[dr] & \\
& {[1, 2, 3]} \\
b \arrow[ur] &
\end{tikzcd}$$



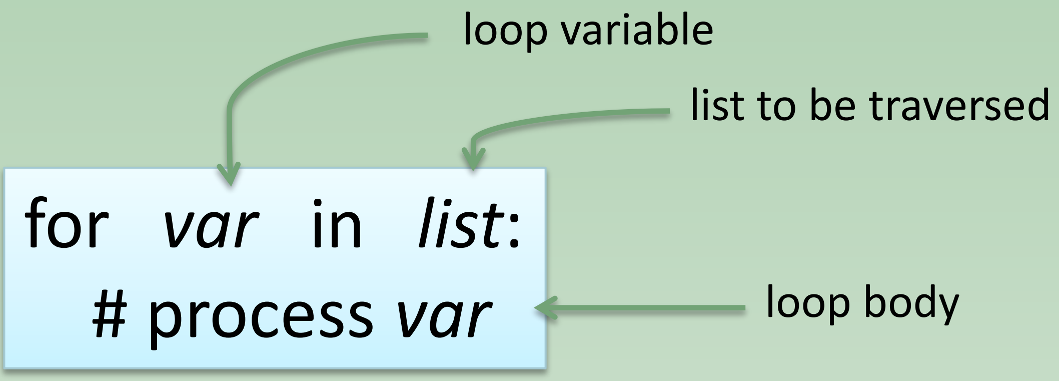
```
>>> a = [1, 2, 3]
>>> b = a
>>> a
[1, 2, 3]
>>> b
[1, 2, 3]
>>> a[0] = 10
>>> a
[10, 2, 3]
>>> b
[10, 2, 3]
>>>
>>> a
[10, 2, 3]
>>> b = a[:]
>>> b
[10, 2, 3]
>>> a[0] = 20
>>> a
[20, 2, 3]
>>> b
[10, 2, 3]
>>>
>>> a == b
False
>>> [1, 2] == [1, 2]
True
>>> a
[20, 2, 3]
>>> a[1:]
[2, 3]
```

let's create a complete copy of *a*

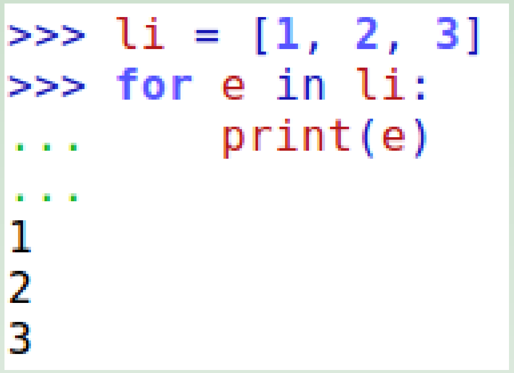
two arrays can be compared

*slices* : they work like
in the case of strings

## foreach loop in Python



```
for var in list:
    # process var
```

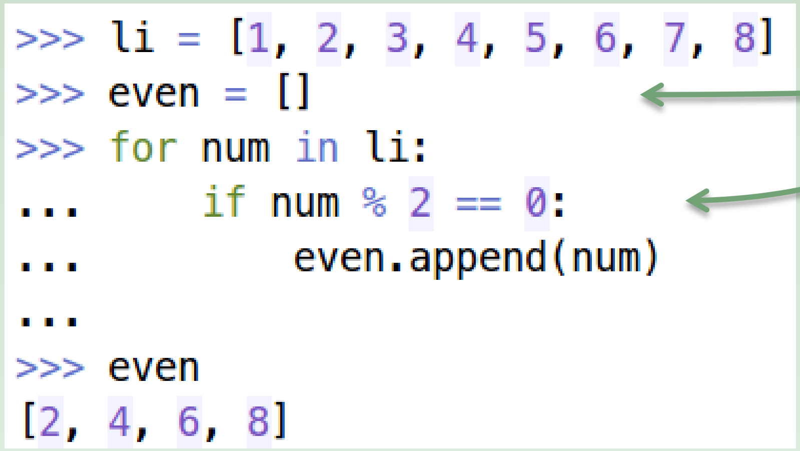


```
>>> li = [1, 2, 3]
>>> for e in li:
...     print(e)
...
1
2
3
```

- works with strings too
- don’t call your list a „list”, because „list” is the name of a built-in function
- similarly, don’t call your string „str”, because „str” is the name of a built-in function

## common pattern

```
res = []    # empty list
for  e  in  your_list:
   res.append(e)
# process res
```



```
>>> li = [1, 2, 3, 4, 5, 6, 7, 8]
>>> even = []
>>> for num in li:
...     if num % 2 == 0:
...         even.append(num)
...
>>> even
[2, 4, 6, 8]
```

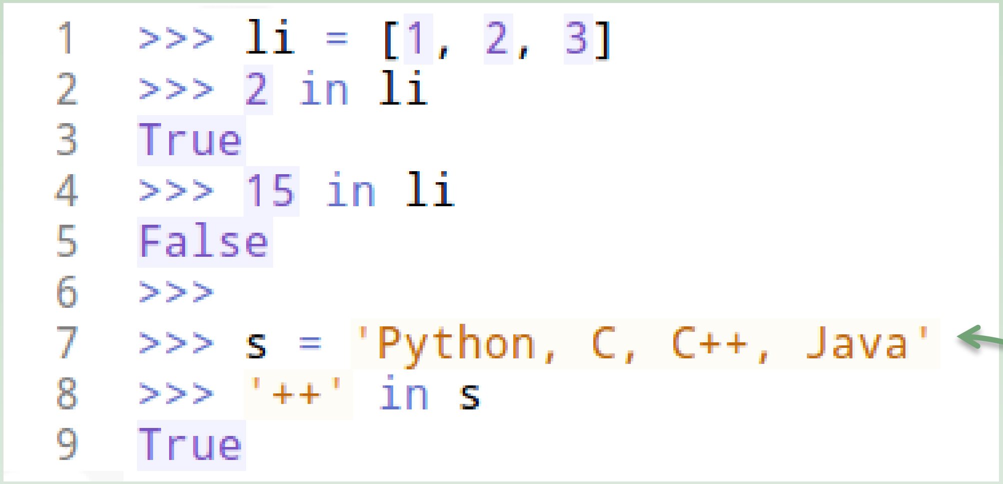
**style:** leave a space <u>before</u> and <u>after</u> an operator

## check if a value is in a list

*value* in *list*

$\longrightarrow$ True

$\longrightarrow$ False



```
>>> li = [1, 2, 3]
>>> 2 in li
True
>>> 15 in li
False
>>>
>>> s = 'Python, C, C++, Java'
>>> '++' in s
True
```

works with strings too

□ homework

# Exercises

1. [20120815b] strings #1
2. [20120815c] strings #2
3. [20130218a] a beautiful mind
4. [20120815e] palindrome (trivial and recursive methods)
5. [20120815j] reverse a whole number
6. [20120818j] number of digits
7. [20120815a] sum of two numbers
8. [20141005a] something_1 or something_2 or … something_N
9. [20141005b] advanced string formatting