



# Scripting Languages

Laszlo SZATHMARY

University of Debrecen  
Faculty of Informatics

## Lab #2

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

(last update: 2023-02-06 [yyyy-mm-dd])

2022-2023, 2nd semester






# Standard data type

## Standard data types in Python:

- number
  - string
  - list
  - tuple
  - dictionary
  - set
- } common name: sequence

# String formatting #1:

```
4 def hello(name, what, color):
5     # mario, the bus is red!
6     print("{0}, the {1} is {2}!".format(name, what, color))
7     # or
8     print("{} , the {} is {}!".format(name, what, color))
9     # or
10    print("{n}, the {w} is {c}!".format(n=name.capitalize(), w=what, c=color))
11
12 def main():
13     hello('mario', 'bus', 'red')
14     print('-' * 22)
15     hello('sara', 'sky', 'blue')
16
17 if __name__ == "__main__":
18     main()
```

common mistake:  „constant”

```
4 PI = 3.14159
5
6 # print('value of PI: ' + PI)      # bad
7 print('value of PI: ' + str(PI))  # better
8 print('value of PI:', PI)         # best
```

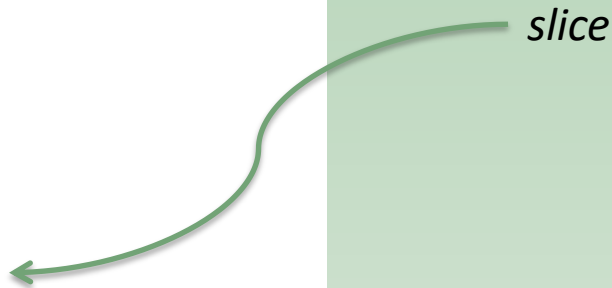


## String formatting #2:

```
3 def hello(name, what, color):
4     print(f"{name}, the {what} is {color}!")
5     # arbitrary expressions can be used:
6     print(f"1 + 1 = {1+1}")      # 1 + 1 = 2
7
8 def main():
9     hello("mario", "bus", "red")
```

<b>B</b>	<b>a</b>	<b>t</b>	<b>m</b>	<b>a</b>	<b>n</b>
0	1	2	3	4	5

```
4 >>> a = 'Batman'
5 >>> a
6 'Batman'
7 >>> len(a)
8 6
9 >>> a[0]
10 'B'
11 >>> a[1:4]
12 'atm'
13 >>> a[0:4]
14 'Batm'
15 >>> a[0:3]
16 'Bat'
17 >>> a[3:6]
18 'man'
19 >>> a[3:]
20 'man'
21 >>> a[:3]
22 'Bat'
23 >>> a[:]
24 'Batman'
25 >>>
```



<b>B</b>	<b>a</b>	<b>t</b>	<b>m</b>	<b>a</b>	<b>n</b>
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

```
1 >>> a
2 'Batman'
3 >>> a[-1]
4 'n'
5 >>> a[-2]
6 'a'
7 >>> a[-6]
8 'B'
9 >>> a[-3:]
10 'man'
11 >>> a[:-3]
12 'Bat'
13 >>>
```

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

```
1 >>> a = 5
2 >>> print(++a)
3 5
4 >>> print --a
5 5
6 >>> print(a++)
7 File "<stdin>", line 1
8     print(a++)
9         ^
10 SyntaxError: invalid syntax
11 >>> print(a--)
12 File "<stdin>", line 1
13     print(a--)
14         ^
15 SyntaxError: invalid syntax
16 >>> --5
17 5
18 >>> a += 1
19 >>> a
20 6
21 >>> a = 5
22 >>> a -= 1
23 >>> a
24 4
25 >>>
```

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

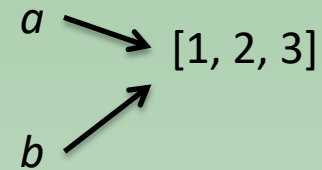
For incrementing / decrementing, use the += and -= operators.

# Lists

empty list

```
4 >>> [1, 2, 3]
5 [1, 2, 3]
6 >>> a = [1, 2, 3]
7 >>> a
8 [1, 2, 3]
9 >>> li = []
10 >>> a = [1, 2, 'ab', 3.14]
11 >>> a
12 [1, 2, 'ab', 3.14]
23 >>> a = [1, 2, 3]
24 >>> a
25 [1, 2, 3]
26 >>> len(a)
27 3
28 >>> [1, 2] + [5, 6]
29 [1, 2, 5, 6]
```

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



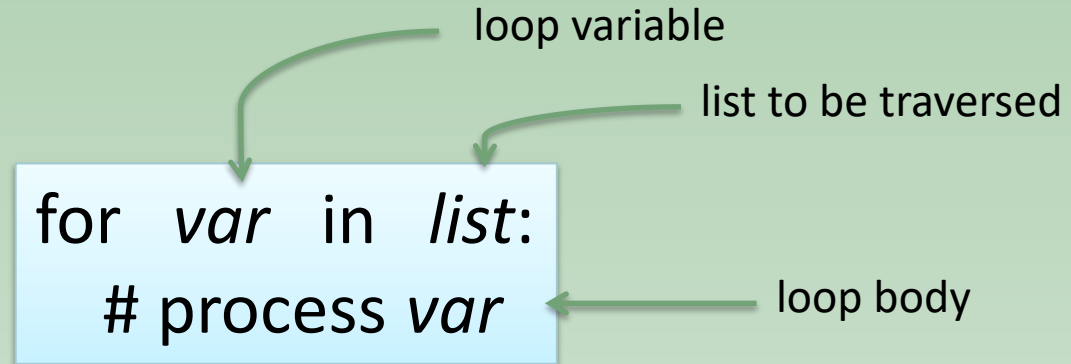
```
1 >>> a = [1, 2, 3]
2 >>> b = a
3 >>> a
4 [1, 2, 3]
5 >>> b
6 [1, 2, 3]
7 >>> a[0] = 10
8 >>> a
9 [10, 2, 3]
10 >>> b
11 [10, 2, 3]
12 >>>
13 >>> a
14 [10, 2, 3]
15 >>> b = a[:]
16 >>> b
17 [10, 2, 3]
18 >>> a[0] = 20
19 >>> a
20 [20, 2, 3]
21 >>> b
22 [10, 2, 3]
23 >>>
24 >>> a == b
25 False
26 >>> [1, 2] == [1, 2]
27 True
28 >>> a
29 [20, 2, 3]
30 >>> a[1:]
31 [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



```
>>> 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  
before and after an  
operator



# check if a value is in a list

*value in list*

→ True

→ False

```
1 >>> li = [1, 2, 3]
2 >>> 2 in li
3 True
4 >>> 15 in li
5 False
6 >>>
7 >>> s = 'Python, C, C++, Java'
8 >>> '++' in s
9 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