

TUTORIAL 4: MORE PYTHON AND CODING

Creating Business Value with Generative AI
Fall 2025

PLAN FOR TODAY

PLAN FOR TODAY

- This slides provide also some type of Python *Cheatsheet*
- Available online in Brightspace

PLAN FOR TODAY

- This slides provide also some type of Python *Cheatsheet*
- Available online in Brightspace

1. Recap of some Python basics

PLAN FOR TODAY

- These slides provide also some type of Python *Cheatsheet*
- Available online in Brightspace

1. Recap of some Python basics
2. Python coding in a notebook on uCloud
 - i. Start together doing live-coding

PLAN FOR TODAY

- This slides provide also some type of Python *Cheatsheet*
 - Available online in Brightspace
1. Recap of some Python basics
 2. Python coding in a notebook on uCloud
 - i. Start together doing live-coding
 - ii. Solve individually and using ChatGPT

PLAN FOR TODAY

- This slides provide also some type of Python *Cheatsheet*
- Available online in Brightspace

1. Recap of some Python basics
2. Python coding in a notebook on uCloud
 - i. Start together doing live-coding
 - ii. Solve individually and using ChatGPT
 - iii. Solve similar problem and fix existing code

PLAN FOR TODAY

- This slides provide also some type of Python *Cheatsheet*
- Available online in Brightspace

1. Recap of some Python basics

2. Python coding in a notebook on uCloud

- i. Start together doing live-coding
 - ii. Solve individually and using ChatGPT
 - iii. Solve similar problem and fix existing code
- ➡ Possible to do the tasks in different levels of complexity

PLAN FOR TODAY

- These slides provide also some type of Python *Cheatsheet*
- Available online in Brightspace

1. Recap of some Python basics
2. Python coding in a notebook on uCloud
 - i. Start together doing live-coding
 - ii. Solve individually and using ChatGPT
 - iii. Solve similar problem and fix existing code

➡ Possible to do the tasks in different levels of complexity
3. Prepare for the next step: Use OpenAI API for analyzing and retrieving data

TABLE OF CONTENTS

- [If-Elif-Else](#)
- [Operators](#)
- [Loops: For and While](#)
- [Data Types](#)
 - [Strings](#)
 - [Lists](#)
 - [Dictionaries](#)
 - [Tuples](#)
- [Function Definitions](#)
- [Helpful Functions](#)
- [Example Read CSV](#)
- [JavaScript Object Notation \(JSON\)](#)

PYTHON BASICS



DEPARTMENT OF MANAGEMENT
AARHUS UNIVERSITY

| Magnus Bender



IF-ELIF-ELSE, PART I

```
duration = 60

if duration < 0:
    print("Time can not be negative!")
elif duration == 60:
    print("Exactly one minute.")
elif duration > 60:
    print("More than one minute.")
else:
    print("It took " + str(duration) + "seconds.")

message = "That was " + ( "fast" if duration < 30 else "too slow" ) + "!"
print(message)
```

IF-ELIF-ELSE, PART I

```
duration = 60
```

```
if duration < 0:  
    print("Time can not be negative!")  
elif duration == 60:  
    print("Exactly one minute.")  
elif duration > 60:  
    print("More than one minute.")  
else:  
    print("It took " + str(duration) + "seconds.")
```

Exactly one minute.

```
message = "That was " + ( "fast" if duration < 30 else "too slow" ) + "!"  
print(message)
```

IF-ELIF-ELSE, PART I

```
duration = 60
```

```
if duration < 0:  
    print("Time can not be negative!")  
elif duration == 60:  
    print("Exactly one minute.")  
elif duration > 60:  
    print("More than one minute.")  
else:  
    print("It took " + str(duration) + "seconds.")
```

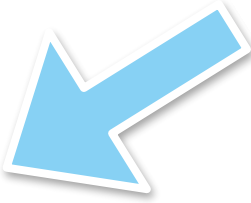
Exactly one minute.

```
message = "That was " + ( "fast" if duration < 30 else "too slow" ) + "!"  
print(message)
```

That was too slow!

IF-ELIF-ELSE, PART I

duration = 29

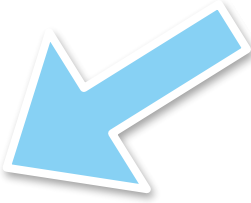


```
if duration < 0:
    print("Time can not be negative!")
elif duration == 60:
    print("Exactly one minute.")
elif duration > 60:
    print("More than one minute.")
else:
    print("It took " + str(duration) + "seconds.")

message = "That was " + ( "fast" if duration < 30 else "too slow" ) + "!"
print(message)
```

IF-ELIF-ELSE, PART I

duration = 29



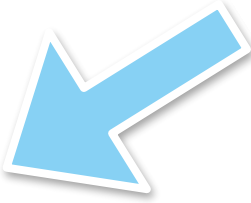
```
if duration < 0:
    print("Time can not be negative!")
elif duration == 60:
    print("Exactly one minute.")
elif duration > 60:
    print("More than one minute.")
else:
    print("It took " + str(duration) + "seconds.")
```

It took 29 seconds.

```
message = "That was " + ( "fast" if duration < 30 else "too slow" ) + "!"
print(message)
```

IF-ELIF-ELSE, PART I

duration = 29



```
if duration < 0:
    print("Time can not be negative!")
elif duration == 60:
    print("Exactly one minute.")
elif duration > 60:
    print("More than one minute.")
else:
    print("It took " + str(duration) + "seconds.")
```

It took 29 seconds.

```
message = "That was " + ( "fast" if duration < 30 else "too slow" ) + "!"
print(message)
```

That was fast!

OPERATORS

Comparison	==	Equality
	> and <	Greater than and less than
	>= and <=	Greater than equal and less than equal
Logical	not	Negation
	and	And
	or	Or
Mathematical	* and **	Multiplication and exponentiation
	/ and //	Division and integer division
	+ and -	Addition and subtraction
	%	Modulo/remainder
	in	Checking for inclusion in tuples, strings, sets
Assign	=	Assignment
	+=	Addition and assignment
	-=	Subtraction and assignment
	/=	Division and assignment

- This is only a small selection.
- Operators are defined for the respective data types.

LOOPS: FOR AND WHILE

```
values = [1, 2, 3, 4]
print(values)
```

```
for v in values:
    print("v is", v)
```

```
values2 = [ v*2 for v in values ]
print(values2)
```

```
while len(values2) > 0:
    print(values2.pop())
```

```
print(values2)
```

LOOPS: FOR AND WHILE

```
values = [1, 2, 3, 4]  
print(values)
```

```
[1, 2, 3, 4]
```

```
for v in values:  
    print("v is", v)
```

```
values2 = [ v*2 for v in values ]  
print(values2)
```

```
while len(values2) > 0:  
    print(values2.pop())
```

```
print(values2)
```

LOOPS: FOR AND WHILE

```
values = [1, 2, 3, 4]  
print(values)
```

```
[1, 2, 3, 4]
```

```
for v in values:  
    print("v is", v)
```

```
v is 1  
v is 2  
v is 3  
v is 4
```

```
values2 = [ v*2 for v in values ]  
print(values2)
```

```
while len(values2) > 0:  
    print(values2.pop())
```

```
print(values2)
```

LOOPS: FOR AND WHILE

```
values = [1, 2, 3, 4]  
print(values)
```

```
[1, 2, 3, 4]
```

```
for v in values:  
    print("v is", v)
```

```
v is 1  
v is 2  
v is 3  
v is 4
```

```
values2 = [ v*2 for v in values ]  
print(values2)
```

```
[2, 4, 6, 8]
```

```
while len(values2) > 0:  
    print(values2.pop())
```

```
print(values2)
```

LOOPS: FOR AND WHILE

```
values = [1, 2, 3, 4]
print(values)
```

```
[1, 2, 3, 4]
```

```
for v in values:
    print("v is", v)
```

```
v is 1
v is 2
v is 3
v is 4
```

```
values2 = [ v*2 for v in values ]
print(values2)
```

```
[2, 4, 6, 8]
```

```
while len(values2) > 0:
    print(values2.pop())
```

```
8
6
4
2
```

```
print(values2)
```

LOOPS: FOR AND WHILE

```
values = [1, 2, 3, 4]
print(values)
```

```
[1, 2, 3, 4]
```

```
for v in values:
    print("v is", v)
```

```
v is 1
v is 2
v is 3
v is 4
```

```
values2 = [ v*2 for v in values ]
print(values2)
```

```
[2, 4, 6, 8]
```

```
while len(values2) > 0:
    print(values2.pop())
```

```
8
6
4
2
```

```
print(values2)
```

```
[]
```

DATA TYPES

DATA TYPES

- True, False and Null

True, False, None

DATA TYPES

- True, False and Null `True, False, None`
- Numbers (int and float) `12, 12.5, 12e3, -20`

DATA TYPES

- True, False and Null

True, False, None

- Numbers (int and float)

12, 12.5, 12e3, -20

- Strings

"Hello World", 'Hello World'

DATA TYPES

- True, False and Null

True, False, None

- Numbers (int and float)

12, 12.5, 12e3, -20

- Strings

"Hello World", 'Hello World'

- Tuples

(1, 2, 3, 4), ("A", 2, "C", None), tuple("ABCD")

DATA TYPES

- True, False and Null

True, False, None

- Numbers (int and float)

12, 12.5, 12e3, -20

- Strings

"Hello World", 'Hello World'

- Tuples

(1, 2, 3, 4), ("A", 2, "C", None), tuple("ABCD")

- Lists

[1, 2, 3, 4], ["A", 2, "C", None], list((1, 2, 3))

DATA TYPES

- True, False and Null

True, False, None

- Numbers (int and float)

12, 12.5, 12e3, -20

- Strings

"Hello World", 'Hello World'

- Tuples

(1, 2, 3, 4), ("A", 2, "C", None), tuple("ABCD")

- Lists

[1, 2, 3, 4], ["A", 2, "C", None], list((1, 2, 3))

- Sets

{"a", "b"}, {"a", "a", "b"}, set(("a", "b", "b"))

DATA TYPES

- True, False and Null

`True, False, None`

- Numbers (int and float)

`12, 12.5, 12e3, -20`

- Strings

`"Hello World", 'Hello World'`

- Tuples

`(1, 2, 3, 4), ("A", 2, "C", None), tuple("ABCD")`

- Lists

`[1, 2, 3, 4], ["A", 2, "C", None], list((1, 2, 3))`

- Sets

`{"a", "b"}, {"a", "a", "b"}, set(("a", "b", "b"))`

- Dictionaries

`{"a" : 1, "b" : 2}, dict((( "a", 1 ), ( "b", 2 )))`

STRINGS

```
s = "Hello World "  
  
print(s[0])  
print(s[:-2])  
print(s[1:3])  
  
print(s.strip() + "!")  
print(s.lower())  
print(s.replace("ll", "j").replace("o", "").replace("W", "w"))  
  
print(s == 'Hello World ')  
  
s += "!"  
print(s * 2)  
print("World" in s)  
  
print(s.split())  
print('-'.join(["Hello", "World!"]))  
  
print("Hello {you}, my name is {me}".format(you="A", me="M"))
```

STRINGS

```
s = "Hello World "
```

```
print(s[0])      H
print(s[:-2])    Hello Worl
print(s[1:3])    el
```

```
print(s.strip() + "!")
print(s.lower())
print(s.replace("ll", "j").replace("o", "").replace("W", "w"))
```

```
print(s == 'Hello World ')
```

```
s += "!"
print(s * 2)
print("World" in s)
```

```
print(s.split())
print('-'.join(["Hello", "World!"]))
```

```
print("Hello {you}, my name is {me}".format(you="A", me="M"))
```

STRINGS

```
s = "Hello World "
```

```
print(s[0])  
print(s[:-2])  
print(s[1:3])
```

H
Hello Worl
el

```
print(s.strip() + "!")  
print(s.lower())  
print(s.replace("ll", "j").replace("o", "").replace("W", "w"))
```

Hello World!
hello world
Hej wrld

```
print(s == 'Hello World ')
```

```
s += "!"  
print(s * 2)  
print("World" in s)
```

```
print(s.split())  
print('-'.join(["Hello", "World!"]))
```

```
print("Hello {you}, my name is {me}".format(you="A", me="M"))
```

STRINGS

```
s = "Hello World "
```

```
print(s[0])  
print(s[:-2])  
print(s[1:3])
```

H
Hello Worl
el

```
print(s.strip() + "!")  
print(s.lower())  
print(s.replace("ll", "j").replace("o", "").replace("W", "w"))
```

Hello World!
hello world
Hej wrld

```
print(s == 'Hello World ')
```

True

```
s += "!"  
print(s * 2)  
print("World" in s)
```

Hello World !Hello World !
True

```
print(s.split())  
print('-'.join(["Hello", "World!"]))
```

['Hello', 'World', '!']
Hello-World!

```
print("Hello {you}, my name is {me}".format(you="A", me="M"))
```

STRINGS

```
s = "Hello World "
```

```
print(s[0])  
print(s[:-2])  
print(s[1:3])
```

H
Hello Worl
el

```
print(s.strip() + "!")  
print(s.lower())  
print(s.replace("ll", "j").replace("o", "").replace("W", "w"))
```

Hello World!
hello world
Hej wrld

```
print(s == 'Hello World ')
```

True

```
s += "!"  
print(s * 2)  
print("World" in s)
```

Hello World !Hello World !
True

```
print(s.split())  
print('-'.join(["Hello", "World!"]))
```

['Hello', 'World', '!']
Hello-World!

```
print("Hello {you}, my name is {me}".format(you="A", me="M"))
```

Hello A, my name is M

Use for any kind of texts and unknown input.

STRINGS

```
s = "Hello World "
```

```
print(s[0])  
print(s[:-2])  
print(s[1:3])
```

H
Hello Worl
el

```
print(s.strip() + "!")  
print(s.lower())  
print(s.replace("ll", "j").replace("o", "").replace("W", "w"))
```

Hello World!
hello world
Hej wrld

```
print(s == 'Hello World ')
```

True

```
s += "!"  
print(s * 2)  
print("World" in s)
```

Hello World !Hello World !
True

```
print(s.split())  
print('-'.join(["Hello", "World!"]))
```

['Hello', 'World', '!']
Hello-World!

```
print("Hello {you}, my name is {me}".format(you="A", me="M"))
```

Hello A, my name is M

LISTS

```
lis1 = list((1, 2, 3))
lis2 = [5, 6, 7]

print(lis1[:-1])
print(lis1 + lis2)

lis1.append(False)
lis1.extend(lis2)
print(lis1)

print(sorted(lis1),
      lis1.sort(), lis1)

for i, v in enumerate(lis2):
    print(i, v)

lis3 = [i for i in range(10)]
print(lis3)
```

LISTS

```
lis1 = list((1, 2, 3))  
lis2 = [5, 6, 7]
```

```
print(lis1[:-1])  
print(lis1 + lis2)
```

```
[1, 2]  
[1, 2, 3, 5, 6, 7]
```

```
lis1.append(False)  
lis1.extend(lis2)  
print(lis1)
```

```
[1, 2, 3, False, 5, 6, 7]
```

```
print(sorted(lis1),  
      lis1.sort(), lis1)
```

```
[False, 1, 2, 3, 5, 6, 7]  
None [False, 1, 2, 3, 5, 6, 7]
```

```
for i, v in enumerate(lis2):  
    print(i, v)
```

```
0 5  
1 6  
2 7
```

```
lis3 = [i for i in range(10)]  
print(lis3)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

LISTS

```
lis1 = list((1, 2, 3))  
lis2 = [5, 6, 7]
```

```
print(lis1[:-1])  
print(lis1 + lis2)
```

```
lis1.append(False)  
lis1.extend(lis2)  
print(lis1)
```

```
print(sorted(lis1),  
      lis1.sort(), lis1)
```

```
for i, v in enumerate(lis2):  
    print(i, v)
```

```
lis3 = [i for i in range(10)]  
print(lis3)
```

```
[1, 2]  
[1, 2, 3, 5, 6, 7]
```

```
[1, 2, 3, False, 5, 6, 7]
```

```
[False, 1, 2, 3, 5, 6, 7]  
None [False, 1, 2, 3, 5, 6, 7]
```

```
0 5  
1 6  
2 7
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Use for ordered collections of data items, do not use for fast inclusion checks or unique data → use set or dictionary instead.

DICTIONARIES

```
dic = {"a" : 1, "b" : 2}
```

```
print(dic["a"])
```

```
dic["c"] = 3
```

```
print(dic)
```

```
del dic["b"]
```

```
print("b" in dic, "b" not in dic)
```

```
for k in dic: # dic.keys()
```

```
    print(k)
```

```
for v in dic.values():
```

```
    print(v)
```

```
for k, v in dic.items():
```

```
    print(k, v)
```

DICTIONARIES

```
dic = {"a" : 1, "b" : 2}
```

```
print(dic["a"])  
dic["c"] = 3  
print(dic)
```

```
del dic["b"]  
print("b" in dic, "b" not in dic)
```

```
for k in dic: # dic.keys()  
    print(k)
```

```
for v in dic.values():  
    print(v)
```

```
for k, v in dic.items():  
    print(k, v)
```

```
1
```

```
{'a': 1, 'b': 2, 'c': 3}
```

```
False True
```

```
a  
c
```

```
1  
3
```

```
a 1  
c 3
```

DICTIONARIES

```
dic = {"a" : 1, "b" : 2}
```

```
print(dic["a"])  
dic["c"] = 3  
print(dic)
```

```
del dic["b"]  
print("b" in dic, "b" not in dic)
```

```
for k in dic: # dic.keys()  
    print(k)
```

```
for v in dic.values():  
    print(v)
```

```
for k, v in dic.items():  
    print(k, v)
```

```
1
```

```
{'a': 1, 'b': 2, 'c': 3}
```

```
False True
```

```
a  
c
```

```
1  
3
```

```
a 1  
c 3
```

Use for key => value data items.
The keys are unique per dictionary and the values can be any type of data.
Fast check if key contained, fast access of values via key.

TUPLES

```
tup1 = (1, 2, 3)
tup2 = 5, 6, 7
```

```
print(tup1[0])
print(tup2)
```

```
a, b = "A", "B"
print(a, b)
```

```
for (i, j) in ((1, "a"), (2, "b"), (3, "c")):
    print(i, j)
```

TUPLES

```
tup1 = (1, 2, 3)
tup2 = 5, 6, 7
```

```
print(tup1[0])
print(tup2)
```

```
a, b = "A", "B"
print(a, b)
```

```
for (i, j) in ((1, "a"), (2, "b"), (3, "c")):
    print(i, j)
```

```
1
(5, 6, 7)
```

```
A B
```

```
1 a
2 b
3 c
```

TUPLES

```
tup1 = (1, 2, 3)
tup2 = 5, 6, 7
```

```
print(tup1[0])
print(tup2)
```

```
a, b = "A", "B"
print(a, b)
```

```
for (i, j) in ((1, "a"), (2, "b"), (3, "c")):
    print(i, j)
```

```
1
(5, 6, 7)
```

```
A B
```

```
1 a
2 b
3 c
```

Tuples cannot be modified or extended! Otherwise, similar to list.

FUNCTION DEFINITIONS

```
def add_or_multiply(x, y, add=True):
```

```
    if add:
```

```
        return x + y
```

```
    else:
```

```
        return x * y
```

```
print(add_or_multiply(1, 2))
```

```
print(add_or_multiply(1, 2, False))
```

```
print(add_or_multiply(x=5, y=6, add=True))
```

```
print(add_or_multiply(x=5, add=False, y=6))
```

```
add_or_multiply = "Hallo"
```

```
add_or_multiply(1, 2)
```

FUNCTION DEFINITIONS

```
def add_or_multiply(x, y, add=True):
```

```
    if add:
        return x + y
    else:
        return x * y
```

```
print(add_or_multiply(1, 2))
print(add_or_multiply(1, 2, False))
```

3
2

```
print(add_or_multiply(x=5, y=6, add=True))
print(add_or_multiply(x=5, add=False, y=6))
```

```
add_or_multiply = "Hallo"
add_or_multiply(1, 2)
```

FUNCTION DEFINITIONS

```
def add_or_multiply(x, y, add=True):
```

```
    if add:
        return x + y
    else:
        return x * y
```

```
print(add_or_multiply(1, 2))
print(add_or_multiply(1, 2, False))
```

3
2

```
print(add_or_multiply(x=5, y=6, add=True))
print(add_or_multiply(x=5, add=False, y=6))
```

11
30

```
add_or_multiply = "Hallo"
add_or_multiply(1, 2)
```

FUNCTION DEFINITIONS

```
def add_or_multiply(x, y, add=True):
```

```
    if add:
        return x + y
    else:
        return x * y
```

```
print(add_or_multiply(1, 2))
print(add_or_multiply(1, 2, False))
```

3
2

```
print(add_or_multiply(x=5, y=6, add=True))
print(add_or_multiply(x=5, add=False, y=6))
```

11
30

```
add_or_multiply = "Hallo"
add_or_multiply(1, 2)
```

```
Traceback:
  add_or_multiply(1,2)
TypeError: 'str' object is not callable
```

HELPFUL FUNCTIONS

Strings	<code>s.strip()</code>	Removes whitespace (spaces) at the beginning and end of a string.
	<code>s.lower()</code>	Converts all characters in a string to their lowercase version.
	<code>s.replace(x, y)</code>	Replaces all occurrences of x with y in a string.
	<code>s.split(x)</code>	Splits a string at each occurrence of x and creates a list.
	<code>s.join(x)</code>	Joins the elements of the list x into a string with s as the separator.
Lists	<code>l.append(x)</code>	Adds a new element x to a list.
Dictionaries	<code>d.items()</code>	Iterates over all elements of a dictionary as tuples of key and value.
	<code>d.values()</code>	Iterates over all values of a dictionary.
Iteration	<code>enumerate(l)</code>	Enumerates all elements of a list, outputting tuples consisting of a run index and value.
	<code>zip(l1, l2)</code>	Iterates over two lists simultaneously and outputs the values with the same index together.
	<code>range(x)</code>	Allows iteration from 0 to x-1.
Types	<code>str(x)</code>	Converts x to a string.
	<code>int(x)</code>	Converts x to an integer (rounding down).
	<code>float(x)</code>	Converts x to a floating point number.
	<code>type(x)</code>	Determines the type of x.
General	<code>print(x)</code>	Outputs x.
	<code>open(f, r)</code>	Opens a file f with permission r ("r" for read, "w" for write).
	<code>len(x)</code>	Determines the length of x.

Import CSV file, filter out all numbers line by line, and export only the numbers as CSV.

AN EXAMPLE

name.csv

A,	Otto,	12,	2045
B,	Heinz,	13,	5689
C,	Franz,	89,	38594
D,	Ernst,	09,	2830

```
def extract_numbers(l):  
    l = l.strip()  
    numbers = []  
    for p in l.split(","):  
        if p.strip().isnumeric():  
            numbers.append(int(p))  
    return numbers  
  
def build_csv(nl):  
    csv = ""  
    for line in nl:  
        csv += ','.join([  
            str(n) for n in line  
        ]) + "\n"  
    return csv
```

```
f = open("name.csv", "r")  
lines = f.readlines()  
f.close()
```

```
new_lines = []  
for line in lines:  
    numbers = extract_numbers(line)  
    new_lines.append([n ** 2 for n in numbers])
```

```
print(build_csv(new_lines))
```

```
144,4182025  
169,32364721  
7921,1489496836  
81,8008900
```

JAVASCRIPT OBJECT NOTATION (JSON)

```
import json
```

```
d = {  
    "a" : 1,  
    "b" : "B",  
    "c" : [1.2, 1.3, 1.4],  
    "e" : None  
}
```

```
json_str = json.dumps(d, indent=2)  
print(json_str)
```

```
d_ = json.loads(json_str)  
print(d_)
```

JAVASCRIPT OBJECT NOTATION (JSON)

```
import json
```

```
d = {  
    "a" : 1,  
    "b" : "B",  
    "c" : [1.2, 1.3, 1.4],  
    "e" : None  
}
```

```
json_str = json.dumps(d, indent=2)  
print(json_str)
```

```
d_ = json.loads(json_str)  
print(d_)
```

```
{  
  "a": 1,  
  "b": "B",  
  "c": [  
    1.2,  
    1.3,  
    1.4  
  ],  
  "e": null  
}
```

JAVASCRIPT OBJECT NOTATION (JSON)

```
import json
```

```
d = {  
    "a" : 1,  
    "b" : "B",  
    "c" : [1.2, 1.3, 1.4],  
    "e" : None  
}
```

```
json_str = json.dumps(d, indent=2)  
print(json_str)
```

```
{  
  "a": 1,  
  "b": "B",  
  "c": [  
    1.2,  
    1.3,  
    1.4  
  ],  
  "e": null  
}
```

```
d_ = json.loads(json_str)  
print(d_)
```

```
{'a': 1, 'b': 'B', 'c': [1.2, 1.3, 1.4], 'e': None}
```