

# Proposing ooRexx and BSF4ooRexx for Teaching Programming and Fundamental Programming Concepts

(ISECON23, 2023-03-31)

## "Business Programming"



Basics,  
Commands,  
Parsing, ...

Messages,  
Directives,  
Classes, ...

Windows (COM/OLE),  
MS Office, ...

GUIs (awt, swing, JavaFX),  
Client/Server (Sockets),  
AOO/LO, Jsoup, XML SAX,  
JDOR Rexx commands, ...

# Overview

---

- WU (acronym from "Wirtschaftsuniversität")
- Background of developing "Business Programming" for decades
- Introducing important critical success factor (programming language)
  - [REXX](#): concepts and nutshell example
  - [ooRexx](#): concepts and nutshell examples
  - [BSF4ooRexx](#): concepts and nutshell examples
- Roundup
- Links
- Addendum ([Rexx vis-à-vis Python](#))

# WU (Business Administration University)

- Based in Vienna, Austria
  - One of the few imperial cities in Europe (located at the "heart of Europe")
- Founded 1898 as a "World Trade High School" (celebrating 125 years)
- Appr. 20,000 students
  - One of the largest universities of its kind
  - Appr. 15,000 Bachelor, 4,200 Master, and 800 Doctoral/PhD
- Information Systems (IS) Department
  - One of eleven departments at WU
  - Currently seven institutes, in alphabetic order
    - "Data, Process and Knowledge Management", "Digital Ecosystems", "Distributed Ledgers and Token Economy", "Information Management and Control", "Information Systems and New Media", "Information Systems and Society", "Production Management"

# Background: "Business Programming"

- Personal challenge of more than 35 years
  - Question: "is it possible to teach interested novice students programming in a single semester such that the students become able to program MS Office?"
  - Evolved over appr. 120 lectures (two lectures each semester)
    - Each lecture's installment got systematically analyzed
    - Observing and analyzing student's problems understanding taught concepts
    - Constantly reworking focus areas, slides, nutshell examples accordingly
    - Experimenting with various programming languages (VBA, VBS, Java, REXX/ooRexx)
  - As of 2023
    - BA students learn in a *four hour lecture (8 ECTS) in a single semester (four months)*
      - Fundamentals of programming
      - Windows and MS Office programming via COM/OLE
      - Platform independent programming of GUIs, client/server, OpenOffice/LibreOffice, ...
  - Key success factor: programming language **ooRexx** and **BSF4ooRexx**

- Mike F. Cowlshaw (IBM)
- IBM released REXX 1979 as a product
  - Became IBM's SAA strategic procedural language in the 80's
- Design of REXX
  - Explicitly human oriented as opposed to the cryptic EXEC 2 it should replace
    - Goal: easy to learn and easy to maintain
  - Principles
    - Typeless language (everything is a string, including numbers)
    - Caseless (everything outside of quotes will be uppcased before processing)
    - No reserved keywords
    - Whitespace can be freely used for formatting instructions for better legibility and better comprehension
      - Multiple whitespace characters between symbols will be reduced to a single whitespace

- Only three instruction types
  - Assignment instruction
    - Variable name, followed by the equal sign (=), followed by an expression
  - Keyword instruction
    - Keywords are English words that convey their meaning
      - Makes REXX programs look like pseudo code
    - Starts with one of the defined keyword instructions like `call`, `if`, `loop`, ...
  - Command instruction
    - Anything else (an expression evaluating to a string)
      - Or explicitly using the `address` keyword instruction which allows one to target the environment the command should get sent to
    - By default the command gets sent to the operating system for execution and the command's `return code` is made directly available to REXX via the variable named **RC**

- A REXX program demonstrating the three instruction types

```
a="Hello, world"      /* assignment      */
do i=1 to 3          /* a loop          */
  say "... round #" i:" a
End

/* command, will have a return code */
"copy file1.txt file1.txt.bkp"
if rc<>0 then /* variable RC set by REXX */
  SAY "Command's return code:" rc
```

Assuming that the file *file1.txt* does not exist such that the copy command will issue the error message "The system cannot find the file specified." in the command line window

## Output:

```
... round # 1: Hello, world
... round # 2: Hello, world
... round # 3: Hello, world
The system cannot find the file specified.
Command's return code: 1
```

- Object-oriented successor for REXX developed by IBM
  - IBM released "Object REXX" 1994 with the operating system "OS/2 Warp"
  - 2004 source code handed over to the non-profit SIG "Rexx Language Assoc."
  - [RexxLA.org](http://RexxLA.org) released "open object Rexx (ooRexx) version 3.0" in 2005
- Design of ooRexx
  - Goals
    - Keep human oriented design principle
    - Run REXX programs unchanged
  - Influenced by SmallTalk
    - Message paradigm (the tilde character ~ is an explicit message operator in ooRexx)
      - Alan Kay (Wikipedia): *I'm sorry that I long ago coined the term "objects" for this topic because it gets many people to focus on the lesser idea. The big idea is "messaging".*
      - Simplifies programming as object's implementation is encapsulated (and becomes irrelevant)



- The message paradigm abstracts from the implementation
  - A programmer conceptually communicates with an object (as if it was a living thing) by sending it a message
    - No need to have any knowledge about the implementation of a method routine
  - The object will search for a method routine by the name of the received message, invokes it (supplying any arguments received with the message) and returns any result to the caller
    - If a method routine is not found in the object's class it will search its superclass up to the root class (thereby realizing inheritance): the first found method routine will be executed by the object
    - Should the object not be able to find the method routine the error message "Object does not understand message" gets raised
- Introduces the *directive* instruction type
  - Placed at the end of a program, led in with two colons :: followed by one of
    - **ANNOTATE, ATTRIBUTE, CLASS, CONSTANT, METHOD, OPTIONS, REQUIRES, RESOURCE, ROUTINE**
  - Processed by the interpreter after the *syntax checking phase*, thereby setting up the program's environment (*setup phase*) before it gets executed (*execution phase*)

# ooRexx, 3

## Directives and Messages



```
p1=.person~new("Albert Einstein", 45000) -- create a new person: person1
say "p1:" p1~name p1~salary             -- show person1's attribute values

p2=.person~new("Mary Withanyname", 35000) -- create a new person: person2
say "p2:" p2~name p2~salary             -- show person2's attribute values

p1~increaseSalary(10000)                 -- increase salary of person1
say "p1:" p1~name p1~salary             -- show person1's attribute values

p2~name="Mary Withaspecificname"         -- change the name of person2
p2~salary=45500                           -- change the salary of person2
say "p2:" p2~name p2~salary             -- show person2's attribute values
say "total of salaries:" p1~salary + p2~salary
```

### Output:

```
p1: Albert Einstein 45000
p2: Mary Withanyname 35000
p1: Albert Einstein 55000
p2: Mary Withaspecificname 45500
total of salaries: 100500
```

```
::class Person -- define name of class
::attribute name -- define attribute "name"
::attribute salary -- define attribute "salary"

::method init -- define constructor (a method routine)
  expose name salary -- establish direct access to attributes
  use arg name, salary -- fetch and store arguments in attributes

::method increaseSalary -- define method routine
  expose salary -- establish direct access to attribute "salary"
  use arg increase -- fetch argument
  salary=salary+increase -- increase value of salary
```

## Messages to MS Excel (Windows)



```

excApp = .OLEObject~new("Excel.Application")
excApp~visible = .true           -- make Excel visible
sheet = excApp~Workbooks~Add~Worksheets[1] -- add and get sheet
      -- set titles from an ooRexx array
titleRange=sheet~range("A1:C1")  -- get title cell range
titleRange~value = .array~of("Austria", "Belgium", "Croatia")
titleRange~font~bold = .true     -- use bold font for titles
sheet~range("A2:C5")~value = createRows(4) -- create and assign array
excApp~displayAlerts = .false    -- no alerts (should file exists already)
fileName=directory()"\test.xlsx" -- save in current directory
Say 'fileName:' fileName        -- show fully qualified file name
sheet~SaveAs(fileName)         -- save file (no alerts, see above)
excApp~quit                     -- quit (end) Excel

```

```

::routine createRows -- create two-dimensional array with arbitrary data
  use arg items=5    -- fetch argument, default, if omitted: 5
  arr=.array~new    -- create Rexx array
  do i=1 to items   -- create random(min,max) numbers
    arr[i,1] = random( 0 ,100 ) -- Austria
    arr[i,2] = random(101,200 ) -- Belgium
    arr[i,3] = random(201,300)  -- Croatia
  end
  return arr        -- return two-dimensional Rexx array

```

## Output:

	A	B	C	D	E
1	<b>Austria</b>	<b>Belgium</b>	<b>Croatia</b>		
2	88	148	261		
3	11	176	250		
4	38	124	250		
5	25	198	206		
6					
7					
8					

Ready

fileName: C:\Users\rony\test.xlsx



- Bidirectional bridge between ooRexx and Java
  - In development since 2000, latest version: BSF4ooRexx850
    - Minimum Java version: 8, minimum ooRexx version: 5.0
    - Includes a Rexx command handler for Java2D named "JDOR" (Java2D for ooRexx)
      - Simplifies using Java2D considerably using Rexx commands
- Design of BSF4ooRexx
  - Goals
    - Keep REXX' human oriented design principle
      - ooRexx programmers need not know implementation details
        - Camouflage Java objects as ooRexx objects that understand messages
      - Allow Java programmers to send ooRexx objects messages from Java
    - Make all Java functionality available to ooRexx in a platform independent manner



- Prerequisites
  - Installation
    - [Java 8](#) or later (Oracle) or [OpenJDK 8](#) (open-source version) or later
      - Hint: use the installation packages with the [JavaFX](#) GUI modules ("FX" or "full" in name)
    - [ooRexx 5.0](#) or later
    - [BSF4ooRexx](#) or [BSF4ooRexx850](#) installed
  - [ooRexx](#) programs
    - Get the camouflaging support by requiring the ooRexx package named [BSF.CLS](#)  
`::requires BSF.CLS -- get ooRexx-Java bridge`

# BSF4ooRexx, 3

## Messages to Java Objects

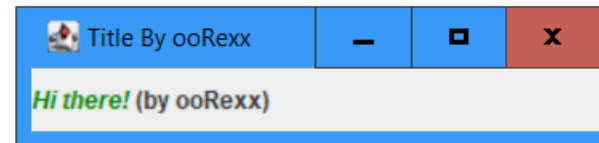


```
jf = .bsf~new("javax.swing.JFrame", "Title By ooRexx") -- create JFrame
lblText = '<html><em style="color: green;">Hi there!</em> (by ooRexx)</html>'
lbl= .bsf~new("javax.swing.JLabel", lblText) -- create JLabel
jf~add(lbl) -- add label
jf~setSize(300,70) -- set size
jf~setLocation(50,200) -- set location
jf~visible=.true -- make visible
jf~ToFront -- place frame in front of all windows
say 'Hit <enter> to proceed (end) ...'
parse pull data -- wait until user presses <enter> on the keyboard

::requires "BSF.CLS" -- get ooRexx-Java bridge
```

**Output:**

```
E:\rony\Vortraege\2023\isecon23\work>rexx code_4_ooRexx_1.rex
Hit <enter> to proceed (end) ...
```



- Java2D
  - Powerful 2D graphics
  - Used for drawing light-weight `javax.swing` classes
  - Used for `Java` games and business graphics of any kind
  - ...
- Example
  - `Java` code to create a `Java2D` graphic
  - Equivalent `ooRexx` code to create the same `Java2D` graphic
    - Rexx `JDOR` commands make this considerably easier
      - WU students immediately take advantage of it after the first `Java` related lecture!



```
import java.awt.*;
import java.awt.geom.AffineTransform;
import javax.swing.*;

/** Test applying affine transform on vector graphics */
@SuppressWarnings("serial")
public class AffineTransformDemo extends JPanel {
    // Named-constants for dimensions
    public static final int CANVAS_WIDTH = 640;
    public static final int CANVAS_HEIGHT = 480;
    public static final String TITLE = "Affine Transform Demo";

    // Define an arrow shape using a polygon centered at (0, 0)
    int[] polygonXs = { -20, 0, +20, 0 };
    int[] polygonYs = { 20, 10, 20, -20 };
    Shape shape = new Polygon(polygonXs, polygonYs, polygonXs.length);
    double x = 50.0, y = 50.0; // (x, y) position of this Shape

    /** Constructor to set up the GUI components */
    public AffineTransformDemo() {
        setPreferredSize(new Dimension(CANVAS_WIDTH, CANVAS_HEIGHT));
    }

    /** Custom painting codes on this JPanel */
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g); // paint background
        setBackground(Color.WHITE);
        Graphics2D g2d = (Graphics2D)g;

        // Save the current transform of the graphics contexts.
        AffineTransform saveTransform = g2d.getTransform();
        // Create a identity affine transform, and apply to the Graphics2D context
        AffineTransform identity = new AffineTransform();
        g2d.setTransform(identity);
    }
    // ... continued ...
}
```

```
// ... continued ...
    // Paint Shape (with identity transform), centered at (0, 0) as defined.
    g2d.setColor(Color.GREEN);
    g2d.fill(shape);
    // Translate to the initial (x, y) position, scale, and paint
    g2d.translate(x, y);
    g2d.scale(1.2, 1.2);
    g2d.fill(shape);

    // Try more transforms
    for (int i = 0; i < 5; ++i) {
        g2d.translate(50.0, 5.0); // translates by (50, 5)
        g2d.setColor(Color.BLUE);
        g2d.fill(shape);
        g2d.rotate(Math.toRadians(15.0)); // rotates about transformed origin
        g2d.setColor(Color.RED);
        g2d.fill(shape);
    }
    // Restore original transform before returning
    g2d.setTransform(saveTransform);
}

/** The Entry main method */
public static void main(String[] args) {
    // Run the GUI codes on the Event-Dispatching thread for thread safety
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            JFrame frame = new JFrame(TITLE);
            frame.setContentPane(new AffineTransformDemo());
            frame.pack();
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setLocationRelativeTo(null); // center the application window
            frame.setVisible(true);
        }
    });
}
}
```



```
-- create a JDOR Rexx command handler
jdh=.bsf~new("org.ooRexx.handlers.jdor.JavaDrawingHandler")
say "JDOR version:" jdh~version -- show version
call BsfCommandHandler "add", "j dor", jdh -- add as a Rexx command handler
address jdor -- set default environment from operating system to JDOR
```

```
newImage 640 480 -- create new image
winShow -- show image in a window
winTitle "Affine Transform Demo (ooRexx)" -- set window's title
```

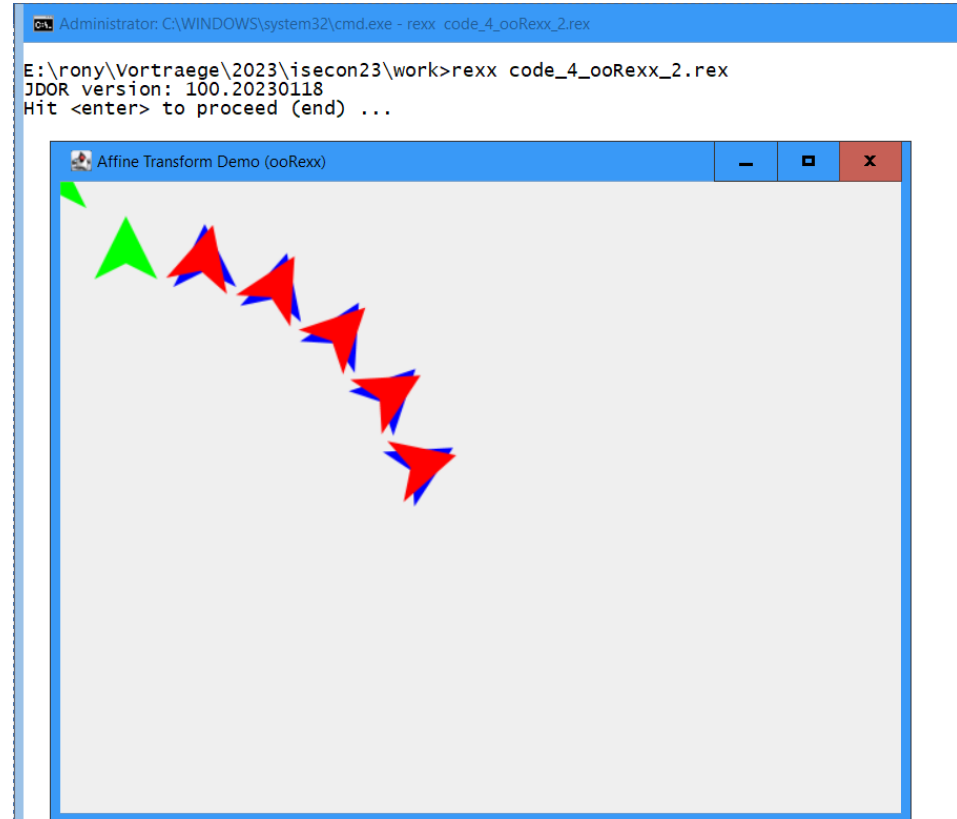
```
-- could use Rexx variables denoting the respective Java arrays instead
polygonXs="(-20,0,+20,0)" -- define four x coordinates
polygonYs="(20,10,20,-20)" -- define four y coordinates
shape myP polygon polygonXs polygonYs 4 -- create polygon shape
color green -- set color to green
fillShape myP -- fill (and show) the polygon shape
translate 50 50 -- move origin (x=x+50, y=y+50)
scale 1.2 1.2 -- increase the polygon shape sizes by 20%
fillShape myP -- fill (and show) the polygon shape
```

```
do 5 -- repeat five times
  translate 50 50 -- move origin (x=x+50, y=y+5)
  color blue -- set color to blue
  fillShape myP -- fill (and show) the polygon shape
  rotate 15 -- rotate by 15°
  color red -- set color to red
  fillShape myP -- fill (and show) the polygon shape
end
```

```
say 'Hit <enter> to proceed (end) ...'
parse pull data -- wait until user presses <enter> on the keyboard
```

```
::requires "BSF.CLS" -- get ooRexx-Java bridge
```

### Output (AffineTransformDemo: Java and ooRexx):



# Roundup

- "Business Programming"
  - Four weekly contact hours for one semester (four months)
  - 8 ECTS, total net teaching load 200 hours
  - *Novices* get empowered by being able to learn programming
    - At the middle of the semester (after two months), after seven installments
      - Fundamental programming concepts, programming Windows (COM/OLE) and MS Office, AOO/LO
    - At the end of the semester (after four months), after seven installments
      - Programming exploiting all of Java camouflaged as ooRexx
        - GUI (awt, swing, [JavaFX](#))
        - Client/server socket programming including SSL/TLS
        - Interacting with web servers ([curl](#), [Jsoup](#))
        - Using Java APIs: Apache OpenOffice (AOO)/LibreOffice (LO)
- Critical success factor "programming language"
  - [ooRexx](#) with [BSF4ooRexx](#) (making all of [Java/OpenJDK](#) available, camouflaged as [ooRexx](#))
  - All needed software is free and open-source

# Links (As of 2023-03-20)

- **WU (English):** <https://www.wu.ac.at/en/the-university/about-wu/facts-figures/studierende/>
  - **Business Programming 1 (BP1):** first half of semester (two months)
    - Syllabus (German use e.g. Google translate, deepl.com) 2023: <http://wi.wu.ac.at/rgf/wu/lehre/autowin/2023sBP1/BP1-autowin-2023s-uebersicht.pdf>
    - Slides (English): <https://wi.wu.ac.at/rgf/wu/lehre/autowin/material/foils/>
  - **Business Programming 2 (BP2):** second half of semester (two months)
    - Syllabus (German use e.g. Google translate, deepl.com) 2023: <http://wi.wu.ac.at/rgf/wu/lehre/autojava/2023sBP2/BP2-autojava-2023s-uebersicht.pdf>
    - Slides (English): <https://wi.wu.ac.at/rgf/wu/lehre/autojava/material/foils/>
  - **Some seminar papers, Bachelor and Master thesis with ooRexx, BSF4ooRexx:** <https://wi.wu.ac.at/rgf/diplomarbeiten/>
- **Software**
  - **ooRexx 5.1:** <https://sourceforge.net/projects/oorexx/files/oorexx/5.1.0beta/>
  - **Java/OpenJDK with JavaFX** modules, e.g. <https://www.azul.com/downloads/?package=jdk#zulu>
  - **BSF4ooRexx850:** <https://sourceforge.net/projects/bsf4oorexx/files/beta/20221004/>
- Hock-Chuan, Chua: *"Java Game Programming: 2D Graphics, Java2D and Images"*; *AffineTransformDemo*: [https://www3.ntu.edu.sg/home/ehchua/programming/java/J8b\\_Game\\_2DGraphics.html#zz-2.2](https://www3.ntu.edu.sg/home/ehchua/programming/java/J8b_Game_2DGraphics.html#zz-2.2)
- REXX history (initial specification): <https://speleotrope.com/rexxhist/REXinitspec-1979.pdf>

# Addendum (**Rexx** vis-à-vis **Python**)

- **Rexx** and **Python** programs
  - Instructions
  - Block, selections, multiple selections
  - Parsing strings
- Possible assessment question
  - What concepts need to be explained and understood (by novices) for the **Rexx** solution and for the feature equivalent **Python** solution?

# Rexx and Python, 1

## (Instructions)



```
/* an assignment instruction: */
a="hello world" /* assigns "hello world" to a variable named a */

/* a keyword instruction: */
say a /* output: hello world */

/* a command instruction: */
/* a Windows command (could be typed into a command line window) */
"dir a.txt" /* command: list the file a.txt */
/* variable RC contains the command's return code, 0 means success */
if rc=0 then say "found!"
else say "some problem occurred, rc="+rc /* show return code */
```

```
# an assignment instruction
a="hello world" # assigns "hello world" to a variable named a

# no keyword instruction for output, using built-in function print()
print(a)

# no command instruction using module subprocess instead
import subprocess # import subprocess module
# execute command
completedProcess=subprocess.run("dir a.txt", shell=True) # run command
rc=completedProcess.returncode # fetch return code, an int
if rc==0:
    print("found!") # indentation mandatory (forcing a block)
else:
    print("some problem occurred, rc="+str(rc)) # turn rc into a string
```

# Rexx and Python, 2

## (Blocks, Selection, Multiple Selections)



```
max=5          /* number of repetitions */
loop a=1 to max /* loop block */
  select      /* nested block # 1 */
    when a=1 then say a": first round"
    when a=2 then say a": second round"
    when a=3 then say a": third round"
    otherwise say "(a="a")"
  end
if a=max then
do          /* nested block # 2 */
  say "-> a=max"
  say "-> last round!"
  say "-> loop will end"
end
end
/* output of the above program will be:
1: first round
2: second round
3: third round
(a=4)
(a=5)
-> a=max
-> last round!
-> loop will end
*/
```

```
max=5          # number of repetitions
for a in range(1,max+1): # loop with range() function, must add 1 to max
  match a:      # must be indented, "match" needs Python 3.10 or higher
    case 1: print(str(a)+": first round") # nested block # 1
    case 2: print(str(a)+": second round") # nested block # 1
    case 3: print(str(a)+": third round") # nested block # 1
    case _: print("(a="+str(a)+")") # default, nested block # 1

  if a==max: # must be indented
    print("-> a==max") # nested block # 2
    print("-> last round!") # nested block # 2
    print("-> loop will end") # nested block # 2

""" output of the above program will be:
1: first round
2: second round
3: third round
(a=4)
(a=5)
-> a==max
-> last round!
-> loop will end
"""
```

# Rexx and Python, 3

## (Parsing Strings)



```
text = " John      Doe   Vienna Austria"
parse var text firstName lastName city country
say "first name:" firstName", last name:" lastName", city:" city

text = "Mary Doe Tokyo Japan"
parse var text firstName lastName city . /* ignore country */
say "first name:" firstName", last name:" lastName", city:" city

/* output of the above program will be:
   first name: John, last name: Doe, city: Vienna
   first name: Mary, last name: Doe, city: Tokyo
*/
```

```
text      = " John      Doe   Vienna Austria"
words     = text.split()      # create list of words
firstName = words[0]         # assign to variable
lastName  = words[1]         # assign to variable
city      = words[2]         # assign to variable
print("first name:",firstName+",", "last name:",lastName+",", "city:",city)

text      = "Mary Doe Tokyo Japan"
words     = text.split()     # create list of words
# assign multiple elements in a single statement
firstName, lastName, city = [words[i] for i in (0, 1, 2)]
print("first name:",firstName+",", "last name:",lastName+",", "city:",city)

""" output of the above program will be:
   first name: John, last name: Doe, city: Vienna
   first name: Mary, last name: Doe, city: Tokyo
"""
```