WU WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Department of Information Systems and Operations Management

mipro MAY 20-24 2024

Concepts that Allow Learning the Programming Language Rexx Much Faster than Other Languages

After more than 35 years of teaching novices programming …

Vienna University of Economics and Business ▪ Welthandelsplatz 1, D2-C ▪ A-1020 Vienna

Rony G. Flatscher / Till Winkler

Developing Business Programming

- Specialisation in **"(Business) Information Systems"**
  - As customary at the time, the *most popular languages* were used to teach beginners: Pascal, BASIC, COBOL, C, PROLOG, Visual Basic Script (VBS) / Applications (VBA), Java, ...
- **Surprise** when experimenting with the Rexx programming language
  - Novices learn **much faster and more in-depth** than with popular languages
  - Analysing the **critical success** factors showed that the most important aspect was **the programming language**
- 35 years of **participant observation** (two lectures per semester)
  - Observed difficulties yielded changes in: content, slides, nutshell examples, infrastructure, presentation, ...

Rony G. Flatscher / Till Winkler

Some Historical Bits on Rexx

- Created for IBM mainframes to make programming easier compared to the rather awkward EXEC2
  - **Rexx design goals:** "human centric", "keep the language small", "easy to learn", "easy to understand hence easy to maintain"
  - Rexx is **still instrumental for IBM mainframe operating systems** today!
- Extremely successful in the 80'ies
  - Companies selling Rexx interpreters successfully, **ANSI/INCITS standard** (!)
- Object-oriented successor ("Object Rexx") in the 90'ies
  - **Open-sourced** in 2005 by RexxLA.org – "open object Rexx" (ooRexx)
    - Available for **all major operating systems**
    - Possible to programme even MS Windows applications via OLE ...

Rony G. Flatscher / Till Winkler

Fundamental Rexx Concepts, 1

- "Everything is a string"
  - If a string represents a number, one can carry out arithmetic
- Three instruction types
  1) Assignment
     - Variable name followed by the assignment operator (=) and an expression
  2) Keyword instruction
     - Keywords are English words conveying the intent of the keyword instruction, e.g. SAY, DO, IF, LOOP, CALL, PARSE, SELECT, ITERATE, LEAVE, INTERPRET, ...
     - Makes Rexx code legible as if it was pseudocode
  3) Commands
     - A string passed to the operating system for execution (as if typed in a window)

Fundamental Rexx Concepts, 2

- White space can be freely used to format code for better legibility
  - Space around operators gets removed
  - White space between symbols will be reduced to a single space serving as concatenation operator
  - Hence indentations with white space not significant
- Case of symbols irrelevant
  - Rexx uppercases everything outside of quoted strings
  - No (frustrating) casing errors for novices
- Rexx nutshell examples to stress fundamental concepts
  - Illustrate the language
  - Same examples in the popular Python language to allow direct comparisons

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 command (could be typed into a command line window)          */
"echo Hello World 2" /* execute command                               */
   /* variable RC contains the command's return code, 0 means success */
if rc=0 then say "success!"
        else say "some problem occurred, rc="rc /* show return code   */
```

Output:

```
hello world
Hello World 2
Success!
```

```
# 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("echo Hello World 2", shell=True) # run
rc=completedProcess.returncode  # fetch return code, an int
if rc==0:
    print("found!") # indentation mandatory (forcing a block)
else: # must use + (concatenation operator) with str() function
    print("some problem occurred, rc="+str(rc)) # turn rc into a string
```

Output:

```
hello world
Hello World 2
Success!
```


Rony G. Flatscher / Till Winkler

Blocks, Selection, Multiple Selections

THE Rexx LANGUAGE

```
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:

```
1: first round
2: second round
3: third round
(a=4)
(a=5)
-> a=max
-> last round!
-> loop will end
```

python™

```
max=5       # number of repetitions
for a in range(1,max+1): # loop with range() function, must add 1 to max
            # must use str() function with + (concatenation operator)
    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, must use == instead of =
        print("-> a==max")          # nested block # 2
        print("-> last round!")     # nested block # 2
        print("-> loop will end")   # nested block # 2
```

Output:

```
1: first round
2: second round
3: third round
(a=4)
(a=5)
-> a==max
-> last round!
-> loop will end
```

Nutshell Example, 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:

```
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:

```
first name: John, last name: Doe, city: Vienna
first name: Mary, last name: Doe, city: Tokyo
```

Roundup

- Popular languages are not the best choice for teaching programming!
- Rexx' human centric design allows novices to learn programming much faster
  - **Intrinsic load much lower**
    - Less syntax rules
    - Keywords imply intent (looks like pseudo code)
    - Simply instrument one own's computer with commands
  - **Lower cognitive load is a critical success factor** for teaching novices programming with almost no drop-outs
- Learned programming concepts can be applied to any other programming language (Java, Python, ...)

Experience and Analyse for Yourself!

- Attend CSEE&T 2024, Würzburg, Germany, July 29th through August 1st 2024
  - Four hour workshop *"Teaching Novices Programming and Important Applications in a Single Semester - Critical Factors from Zero to Portable GUI Programming in Four Hours"*, <https://conf.researchr.org/track/cseet-2024/cseet-2024-workshops>
- Free Zoom course in English for interested employees of WU, August 2024
  - From August 5th (Monday) through August 23rd (Friday) 2024
    - Daily from 9:00-13:00 MEST (middle European summer time)
  - You can join for free, if interested :-)
    - Open for yourself, your PhD students, or for interested colleagues of yours
      - Please, if you do, stick to the three weeks
    - There will be a proper MailMan e-mail list for interested IT professional instructors
    - To register every person simply sends an e-mail by clicking the following link:
      - <mailto:Rony.Flatscher@wu.ac.at?subject=[WU_BP_2024s_extern]%20Registration>
      - You will get further information sent after registering
    - There will be optional sessions for IT lecturers to discuss the pedagogics

Some References

- **Open and free slides**
  - R. G. Flatscher, “Introduction to Programming with ooRexx and BSF4ooRexx 1. 1-7.” [PDF slides]:
    - <https://wi.wu.ac.at/rgf/wu/lehre/autowin/material/foils/>
  - R. G. Flatscher, “Introduction to Programming with ooRexx and BSF4ooRexx 2. 8-14.” [PDF slides]:
    - <https://wi.wu.ac.at/rgf/wu/lehre/autojava/material/foils/>
- T. Winkler, "Collection of Rexx References". <https://wi.wu.ac.at/rgf/rexx/rexxref/searchref.html>
  - Maintained at: <https://gitlab.com/dylwi/rexx-references>
- R. G. Flatscher and G. Müller, “'Business Programming' – Critical Factors from Zero to Portable GUI Programming in Four Hours,” in 6th BEE-Conference, Plitvice Lakes, Croatia, 2021, pp. 76-82.
  - <https://research.wu.ac.at/files/32933925/2021_BusinessProgramming_BEE2021_accordingToGuidelines.pdf>
- R. G. Flatscher, "Proposing ooRexx and BSF4ooRexx for Teaching Programming and Fundamental Programming Concepts", in 2023 Program Guide ISECON: Information Systems Education Conference, Dallas/Plano, Tx, 2023, pp. 89-102.
  - <https://research.wu.ac.at/files/41301564/ISECON23_Flatscher_Proposing_ooRexx_article.pdf>
- T. Winkler and R. G. Flatscher, “Cognitive Load in Programming Education: Easing the Burden on Beginners with REXX.” In Central European Conference on Information and Intelligent Systems. 2023, pp. 171-178.
  - <https://research.wu.ac.at/files/46150789/CECIIS_CLT_REXX.pdf>
- R. G. Flatscher, "Introduction to Rexx and ooRexx – From Rexx to Open Object Rexx (ooRexx)", text book, ISBN 9789403 739298

Some Links

- Portable zip archives (no installation needed): ooRexx 5.1, oorexxshell, dbusoorexx, bsf4oorexx
  - <https://www.ronyrexx.net/xfer/portable>
    - Note: bsf4oorexx (ooRexx-Java bridge) needs Java/OpenJDK installed
- Installation packages
  - ooRexx 5.1:
    - <https://sourceforge.net/projects/oorexx/files/oorexx/5.1.0beta/>
  - BSF4ooRexx (ooRexx-Java bridge, needs Java/OpenJDK preinstalled):
    - <https://sourceforge.net/projects/bsf4oorexx/files/GA/BSF4ooRexx-850.20240304-GA/>
- Selected seminar papers, Bachelor and Master thesis with ooRexx, BSF4ooRexx, dbusoorexx
  - <https://wi.wu.ac.at/rgf/diplomarbeiten/>
- Non-profit Rexx Language Association (owner of ooRexx):
  - <https://www.RexxLA.org>
- Web page with Rexx related resources maintained by R.G. Flatscher:
  - <https://ronyrexx.net>