2) Basic components

clauses,
statements,
comments,
literal strings,
numbers,
symbols,
variables,
labels

arithmetic and concatenation operators.

Instructions: Assignment, DROP, UPPER, NUMERIC

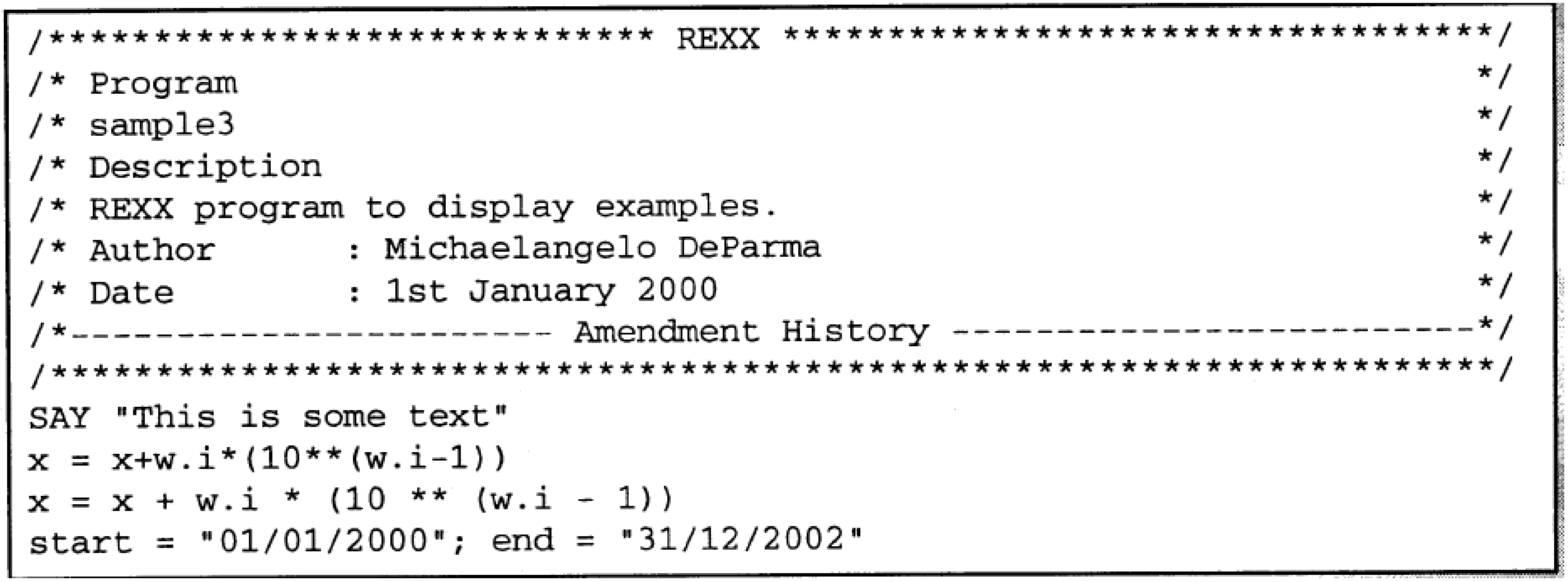
Resources: TSO/E REXX Reference

This course has been prepared by Milos Forman for MCoE needs only!

Clauses

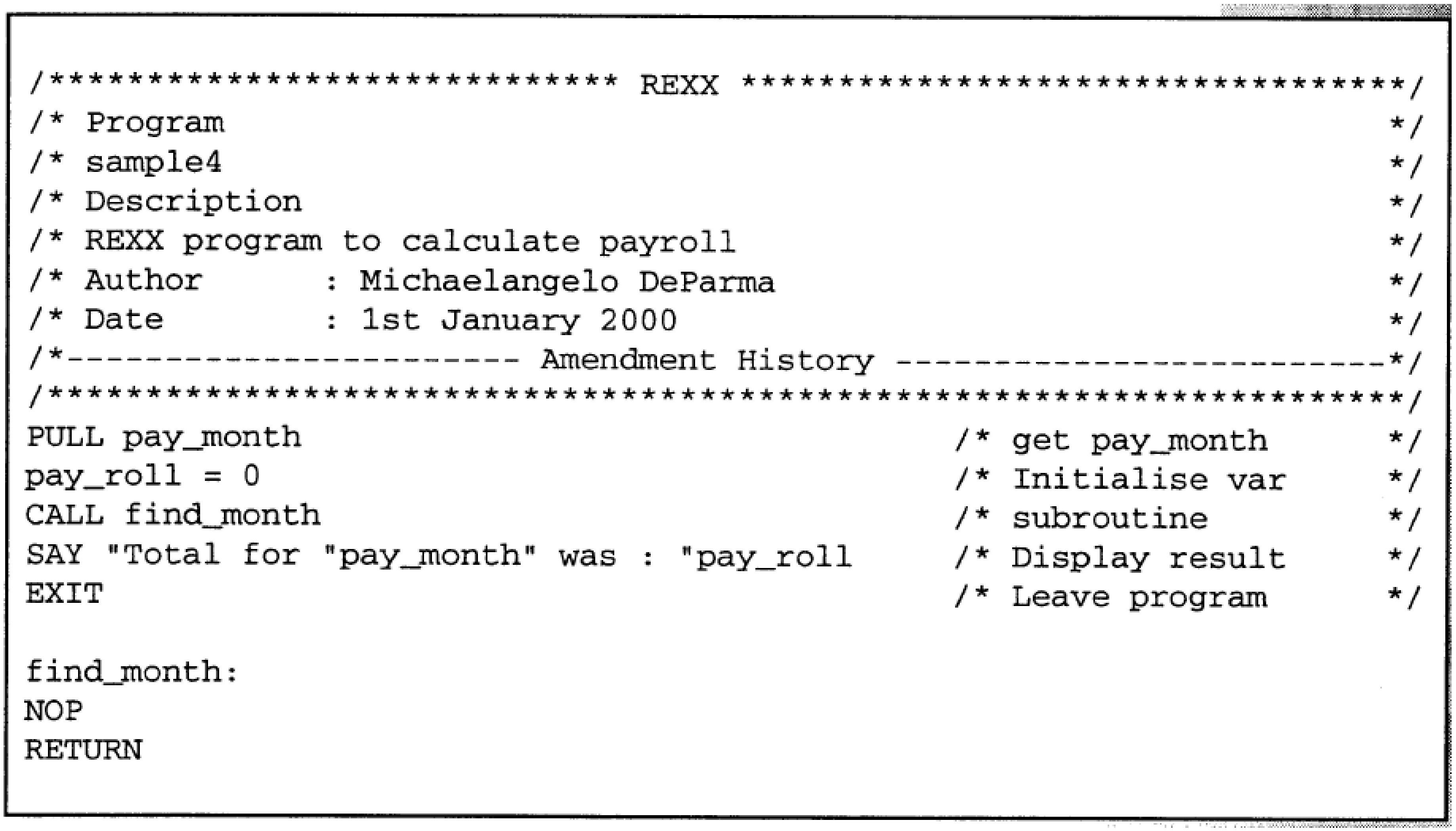
- REXX Instructions
- Tokens
  - symbols
  - operators
  - literal
  - special characters
- Blanks

Clauses example



```
/****************************** REXX **********************************/
/* Program                                                            */
/* sample3                                                            */
/* Description                                                        */
/* REXX program to display examples.                                  */
/* Author        : Michaelangelo DeParma                              */
/* Date          : 1st January 2000                                   */
/*----------------------- Amendment History --------------------------*/
/**********************************************************************/
SAY "This is some text"
x = x+w.i*(10**(w.i-1))
x = x + w.i * (10 ** (w.i - 1))
start = "01/01/2000"; end = "31/12/2002"
```

Sample Program



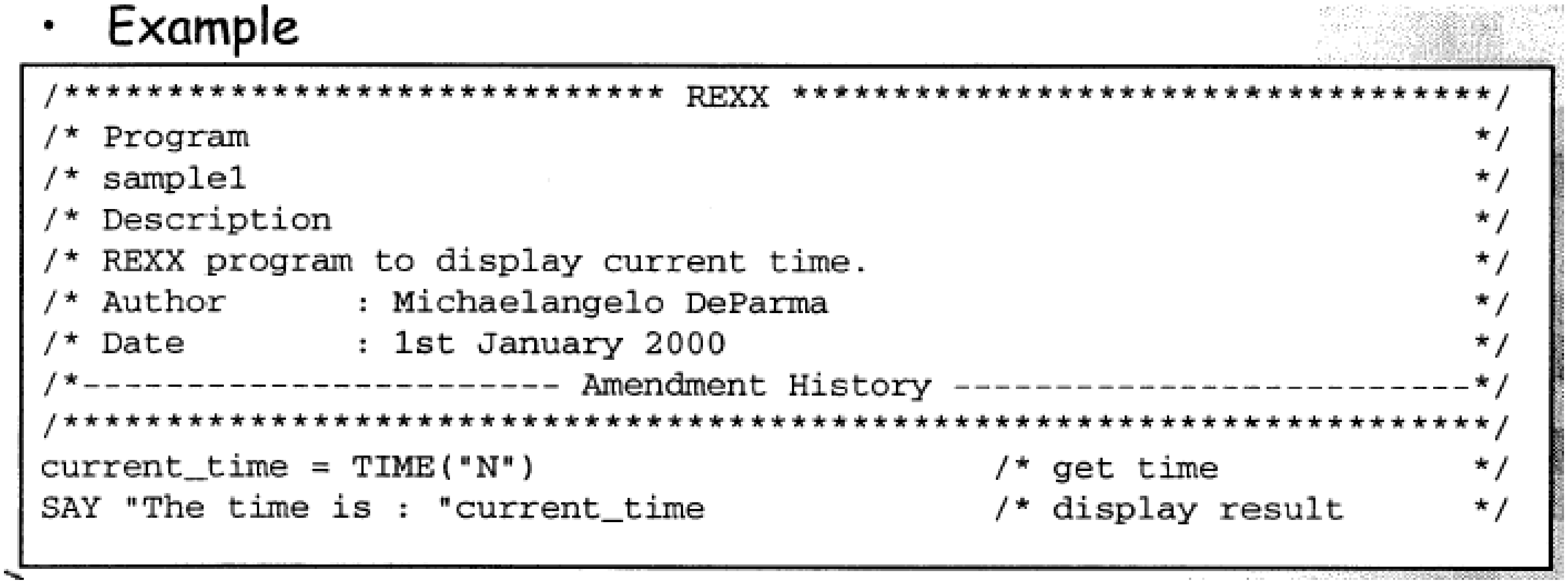
```
/***************************** REXX ***********************************/
/* Program                                                            */
/* sample4                                                            */
/* Description                                                        */
/* REXX program to calculate payroll                                  */
/* Author        : Michaelangelo DeParma                              */
/* Date          : 1st January 2000                                   */
/*----------------------- Amendment History --------------------------*/
/**********************************************************************/
PULL pay_month                                /* get pay_month        */
pay_roll = 0                                  /* Initialise var       */
CALL find_month                               /* subroutine           */
SAY "Total for "pay_month" was : "pay_roll    /* Display result       */
EXIT                                          /* Leave program        */

find_month:
NOP
RETURN
```

Types of statements

- Null
  - blank or comments
- Label
  - label_name:
- Assignments
  - literal, numbers, symbols, operators, variables
- Keyword instructions
- Commands

Comments

- Begin with /* and end with */
- Can contain any sequence of characters on one or more lines.
- REXX program should start with comment line.
- **This first comment should contain string REXX**



- Example

```
/******************************* REXX ***********************************/
/* Program                                                              */
/* sample1                                                              */
/* Description                                                          */
/* REXX program to display current time.                                */
/* Author       : Michaelangelo DeParma                                 */
/* Date         : 1st January 2000                                      */
/*----------------------- Amendment History ---------------------------*/
/***********************************************************************/
current_time = TIME("N")                        /* get time             */
SAY "The time is : "current_time                /* display result       */
```

Literal Strings

- A series of characters enclosed in single or double quotes
- A literal string can contain ANY characters
- Maximum size may vary
- Continued by placing a comma after the last character on the line continuing in column 1 on next line

Literal Strings

- If a literal contains a ' use a "
- If a literal contains a " use a '
- Strings starting with quotes must also end with quotes
- Any string with no characters, "", is called a null string and has a length of 0

Literal Strings examples

```
SAY "12345"
SAY 'abc'
SAY "ABC"
SAY "Please enter a number"
SAY "Your name is 'Bob Flemming'"
```

```
12345
abc
ABC
Please enter a number
Your name is 'Bob Flemming'
***
```

This course has been prepared by Milos Forman for MCoE needs only!

Hexadecimal and Binary strings

Hexadecimal string:
- Contain characters 0-9 A-F or a-f
- Delimited by quotes
- Followed by x or X

```
num_1 = 'ABCD'X
num_2 = "1d ec f8"X
num_3 = '123 45'x
```

Binary string:
- Contain characters 0-1
- Delimited by quotes
- Followed by b or B

```
'11110000'b        /* == 'f0'x                  */
"101 1101"b        /* == '5d'x                  */
'1'b               /* == '00000001'b and '01'x  */
'10000 10101010'b  /* == '0001 0000 1010 1010'b */
''b                /* == ''                     */
```

Numbers

- Character strings that consist of one or more decimal digits
- Optionally prefixed by a plus or minus
- Can include decimal point
- can be exponential notation

```
num_1 = 29
num_2 = -94.3
num_3 = 53.312E23
num_4 = +0.5
num_5 = 0.73E-7
```

Test Exercise 21

Write a REXX program using the following code. Test and identify the type of statements. Which statement is :

- Comment
- Literal
- Hexadecimal
- number
- error

```
/* REXX EXEC */
a = "REXX EXEC"
c = 94 4F
d = '94 4F'
e = 32
f = '32'
g = "32"x
b = "REXX EXEC"x
REXX EXEC
```

Symbols

- A group of any characters selected from the English alphabetic and numeric characters (A-Z, a-z, 0-9) and/or from the characters .!? And _
- Symbols are used to name variables, functions, instructions, labels etc.

```
a = ALLEN
b = this is a symbol!
c = hello?
test_routine:
```

This course has been prepared by Milos Forman for MCoE needs only!

Simple Variables

- A variable is a name that represents a value.
- First character cannot be a number or a period
- Variable names must be less than 250 characters

```
salary = pay + benefit + bonus
job_class = "A"
msg_class = "X"
operator = current_operator_for_this_task
```

This course has been prepared by Milos Forman for MCoE needs only!

Simple Variables

- Variable names can only consist of the following :
  - A-Z, 0-9, a-z, @#$?!._

- Variable names are converted to upper case for comparison

- Variables are initialised to their own name in upper case.

```
SAY pay_roll
```

```
PAY_ROLL
***
```

This course has been prepared by Milos Forman for MCoE needs only!

Test Exercise 22

Write a REXX program to identify which are valid statements.

- Try and fix where possible.

```
Answer = "yes"
The-Answer = "no"
Msg.Text = "IST510I"
1ST_Class = "never"
the_worlds_largest_variable = "test"
REPLLY? = "Z NET,QUICK"
SAY reply
```

This course has been prepared by Milos Forman for MCoE needs only!

Compound Variables

- A technique for grouping variables
- Start with a stem which is a valid variable name followed by a period (.)
- Characters after the first period are called tail

```
CPUTIME.ASID
Stem    Tail

new_rate.shift.type.weekly
Stem      Tail  Tail Tail
```

Compound Variables

The name begins with a **stem** (that part of the symbol up to and including the first period). This is followed by a **tail**, parts of the name (delimited by periods) that are constant symbols, simple symbols, or null. The **derived name** of a compound symbol is the stem of the symbol, in uppercase, followed by the tail, in which all simple symbols have been replaced with their values. A tail itself can be comprised of the characters A–Z, a–z, 0–9, and @ # $ ¢ . ! ? and underscore. The value of a tail can be any character string, including the null string and strings containing blanks. For example:

At execution time, each variable is replaced by its current value to generate the derived name.

```
/* REXX ***********************************************************/
/* Program    - RX201218                                          */
/******************************************************************/
INDEX = 2
DAY.1  = "Mon"
DAY.2  = "Tue"
DAY.3  = "Wed"
say  DAY.INDEX
```

Assigning Values to the Stem

If a value is assigned to a stem, ALL POSSIBLE compound variables that begin with that stem also are assigned to this value.

```
name.  = "Nobody"
a = 2
b = 3
name.a = "Jed"
name.b = "Jim"
SAY name.1 name.2 name.3 a b
```

```
Nobody Jed Jim 2 3
***
```

This course has been prepared by Milos Forman for MCoE needs only!

Arithmetic Operators

- Prefix operators
  - +, -
- Power Operators
  - **
- Multiplication and division
  - *, /, %, //
- Addition and subtraction
  - +, -
- Grouping
  - ()

Test Exercise 23

- What are the answers to the following:
  - 2+3*4
  - (2+3)*4
  - 9**2/10
  - 23%5//2**3
- Now check how REXX calculates these numbers.

Concatenation Operators

- Concatenation
  - blank
  - ||
  - abutted

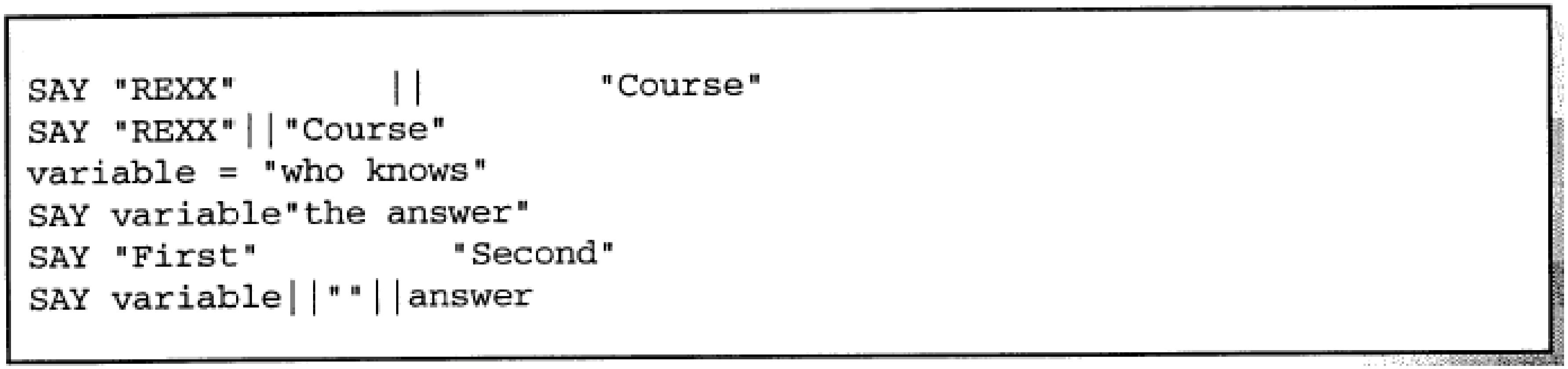
Priority:
Aritmetic operators
Concatenation operators

```
a = "one"; b = "two"
total = "75"
SAY a b
SAY ab
SAY a||b
SAY total "%"
SAY total"%"
```

```
one two
AB
onetwo
75 %
75%
***
```

Test Exercise 24

- Test the following :



```
SAY "REXX"       ||        "Course"
SAY "REXX"||"Course"
variable = "who knows"
SAY variable"the answer"
SAY "First"         "Second"
SAY variable||""||answer
```

Keyword Instructions

- One or more clauses, first of which starts with a keyword that identifies the instruction
- Keyword instructions control
  - external and internal interfaces
  - affect the flow of control
- Some keywords can included nested instructions
- Keywords are not case dependant.

Keyword Instructions

```
IF title = "Yes" THEN DO
  SAY "Hello"
END
DO WHILE title = "Yes"
   SAY "Hi"
   LEAVE
END
```

Work Section 2.1

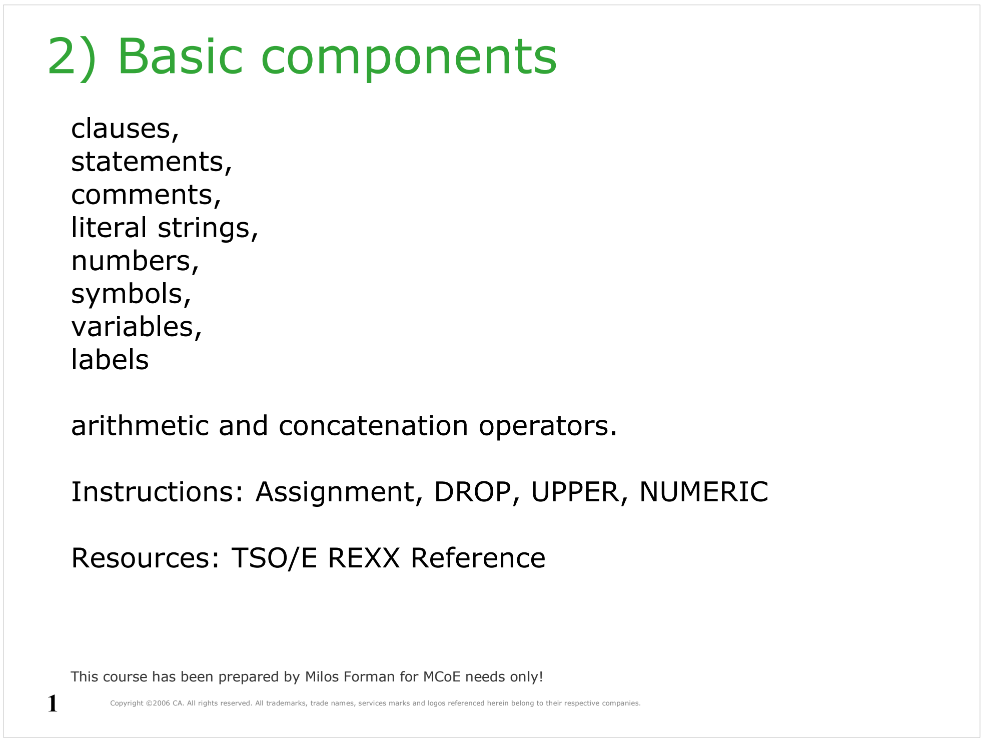
- Write a REXX program to manipulate information entered from the terminal.
- Prompt the user to enter their first name. Then prompt the user to enter their last name.
- After they do this display the first and last names in the following formats on the terminal

```
first_name last_name
last_name, first_name
last_namefirst_name
first_namelast_name
```

Work Section 2.2

- Write a REXX program to prompt a user to enter their name.
- Then prompt for their gross salary.
- Then display their name and the tax they have paid at 30%

```
 Please enter your first name.
Mike
 Please enter your salary.
20000
 Your tax paid is :6000.0
 ***
```



2) Basic components

clauses,
statements,
comments,
literal strings,
numbers,
symbols,
variables,
labels

arithmetic and concatenation operators.

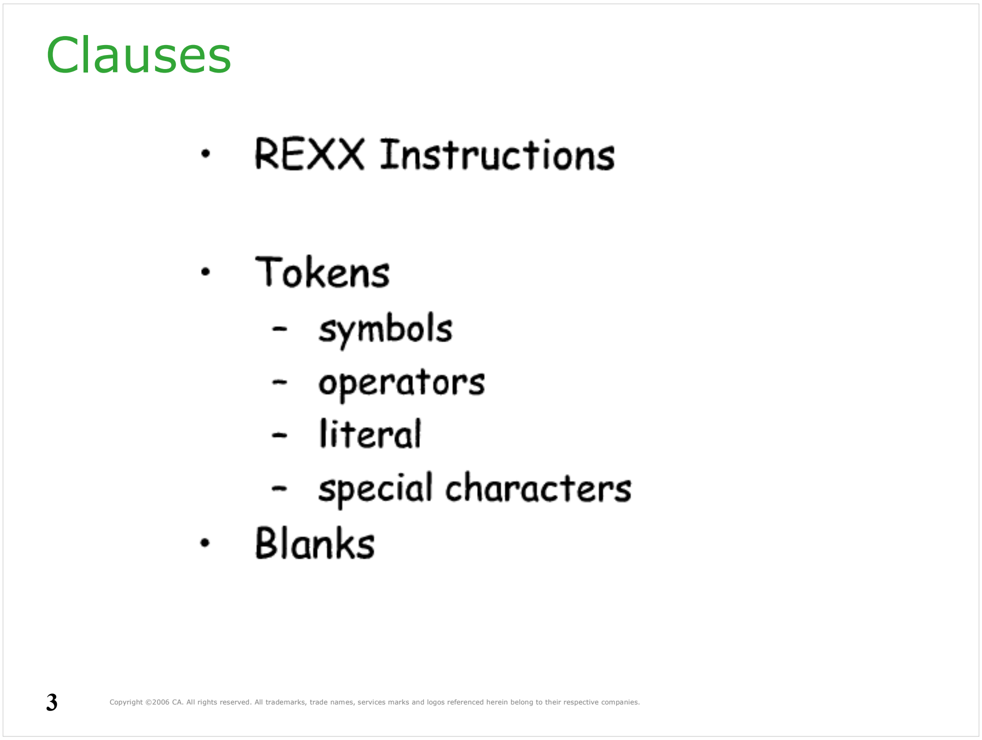
Instructions: Assignment, DROP, UPPER, NUMERIC

Resources: TSO/E REXX Reference

This course has been prepared by Milos Forman for MCoE needs only!

1

See TSO/E REXX Reference



Clauses

- REXX Instructions
- Tokens
  - symbols
  - operators
  - literal
  - special characters
- Blanks

3

Just the terminology – clause is created by:

•Null clause

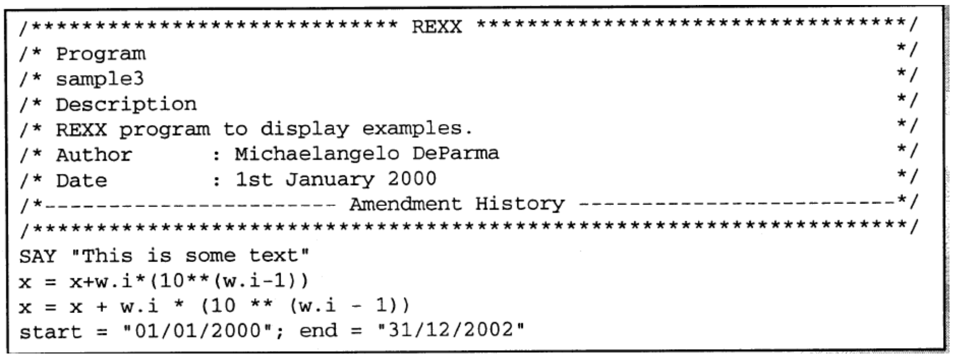
•Label

•Assignment

•Keyword instruction

•Command

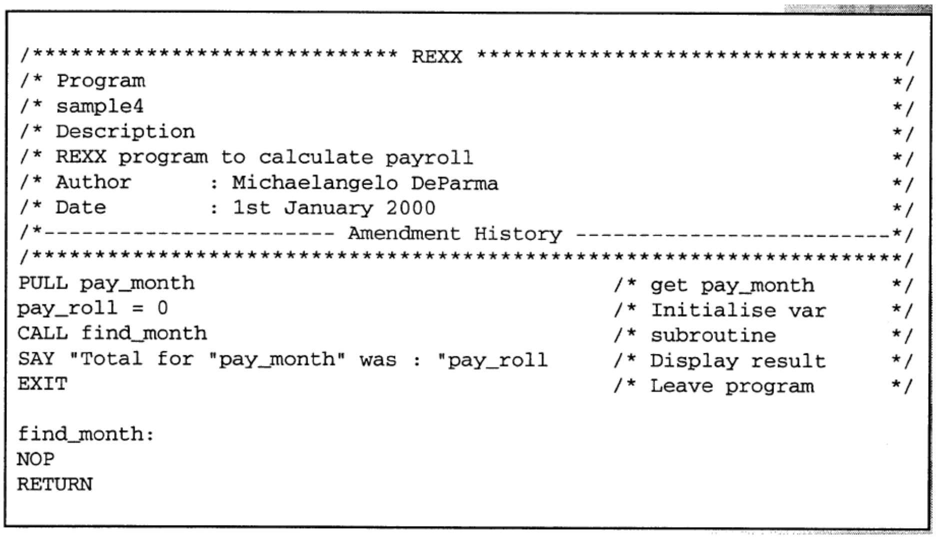
Clauses example



```
/****************************** REXX ***********************************/
/* Program                                                             */
/* sample3                                                             */
/* Description                                                         */
/* REXX program to display examples.                                   */
/* Author       : Michaelangelo DeParma                                */
/* Date         : 1st January 2000                                     */
/*----------------------- Amendment History --------------------------*/
/**********************************************************************/
SAY "This is some text"
x = x+w.i*(10**(w.i-1))
x = x + w.i * (10 ** (w.i - 1))
start = "01/01/2000"; end = "31/12/2002"
```

We can consider clause as a sentenece. There are four clauses in the example.

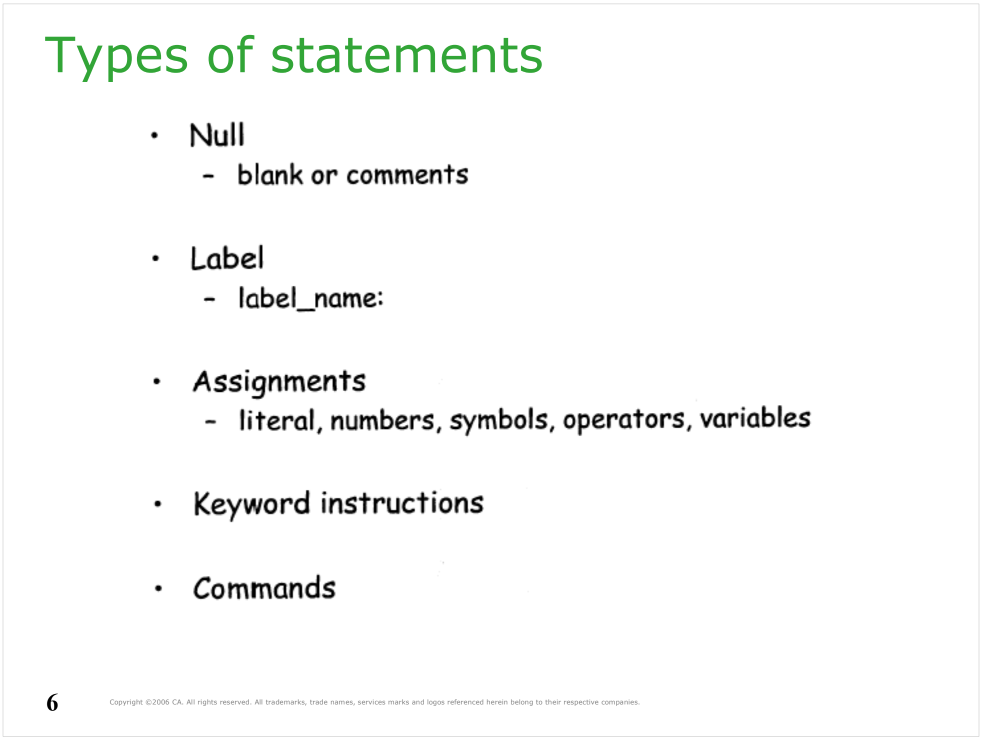
Sample Program



```
/***************************** REXX ***********************************/
/* Program                                                            */
/* sample4                                                            */
/* Description                                                        */
/* REXX program to calculate payroll                                  */
/* Author       : Michaelangelo DeParma                               */
/* Date         : 1st January 2000                                    */
/*------------------------ Amendment History -------------------------*/
/**********************************************************************/
PULL pay_month                                 /* get pay_month      */
pay_roll = 0                                   /* Initialise var     */
CALL find_month                                /* subroutine         */
SAY "Total for "pay_month" was : "pay_roll     /* Display result     */
EXIT                                           /* Leave program      */

find_month:
NOP
RETURN
```

5

What is on the picture? REXX sample program with couple of clauses...



Types of statements

- Null
  - blank or comments
- Label
  - label_name:
- Assignments
  - literal, numbers, symbols, operators, variables
- Keyword instructions
- Commands

6

Clauses can be subdivided into the following types:

Null Clauses A clause consisting only of blanks or comments or both.

Labels A clause that consists of a single symbol followed by a colon.

Assignments A single clause of the form *symbol=expression* is an instruction known as an **assignment**.

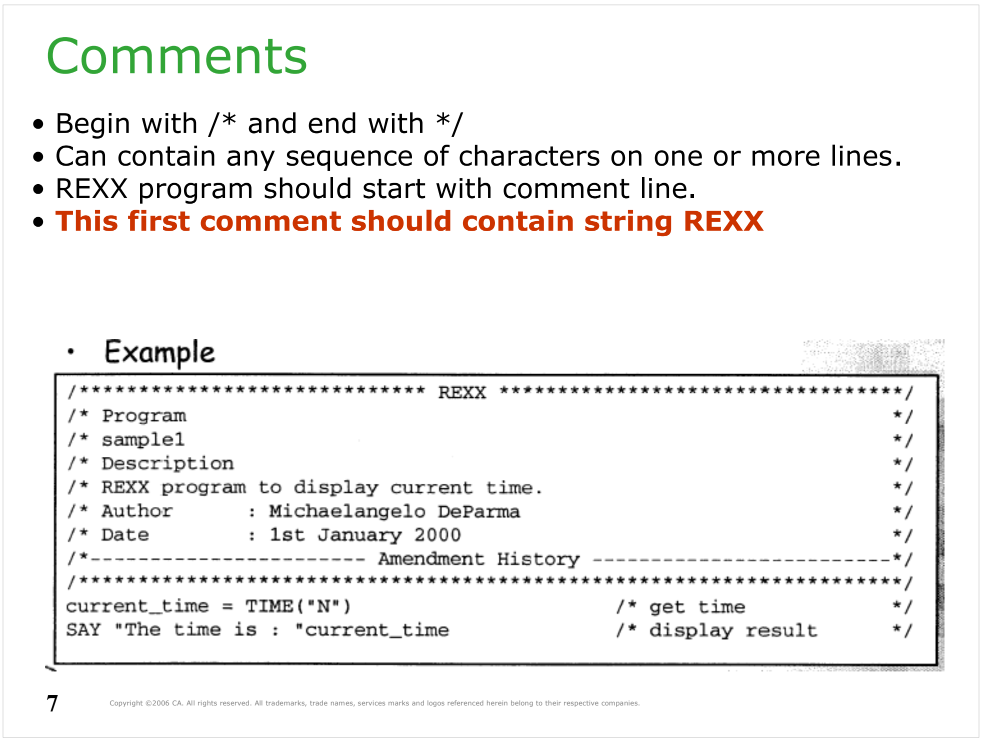
Instructions An **instruction** consists of one or more clauses describing some action to take. Instructions can be: assignments, keyword instructions, or commands.

Keyword Instructions A **keyword instruction** is one or more clauses, the first of which starts with a keyword that identifies the instruction.

Commands A **command** is a clause consisting of only an expression. The expression is evaluated and the result is passed as a command string to some external environment.

If REXX does not understand the command, the execution ends with:

```
COMMAND DAY NOT FOUND
    3 *-* day date()  say 'Hello World'
      +++ RC(-3) +++
```



Comments

- Begin with /* and end with */
- Can contain any sequence of characters on one or more lines.
- REXX program should start with comment line.
- **This first comment should contain string REXX**

- Example

```
/***************************** REXX ************************************/
/* Program                                                             */
/* sample1                                                             */
/* Description                                                         */
/* REXX program to display current time.                               */
/* Author      : Michaelangelo DeParma                                 */
/* Date        : 1st January 2000                                      */
/*----------------------- Amendment History ---------------------------*/
/***********************************************************************/
current_time = TIME("N")                       /* get time             */
SAY "The time is : "current_time               /* display result       */
```

7

If you omit the string REXX in the first line:

COMMAND SAY NOT FOUND OR REXX IDENTIFIER IS MISSING+

SUPPLY '/* REXX */' AS THE FIRST RECORD TO EXECUTE AS A REXX EXEC...

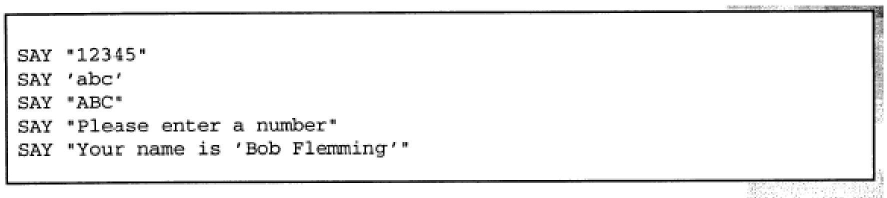
Literal Strings

- A series of characters enclosed in single or double quotes
- A literal string can contain ANY characters
- Maximum size may vary
- Continued by placing a comma after the last character on the line continuing in column 1 on next line

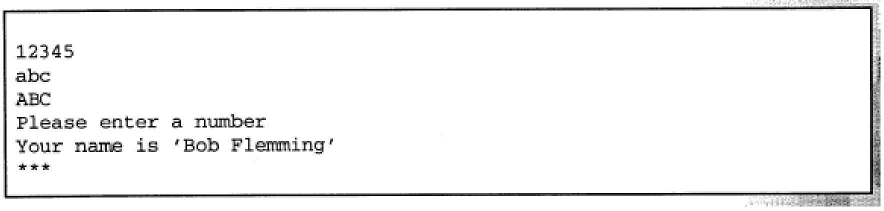
Literal Strings

- If a literal contains a ' use a "
- If a literal contains a " use a '
- Strings starting with quotes must also end with quotes
- Any string with no characters, "", is called a null string and has a length of 0

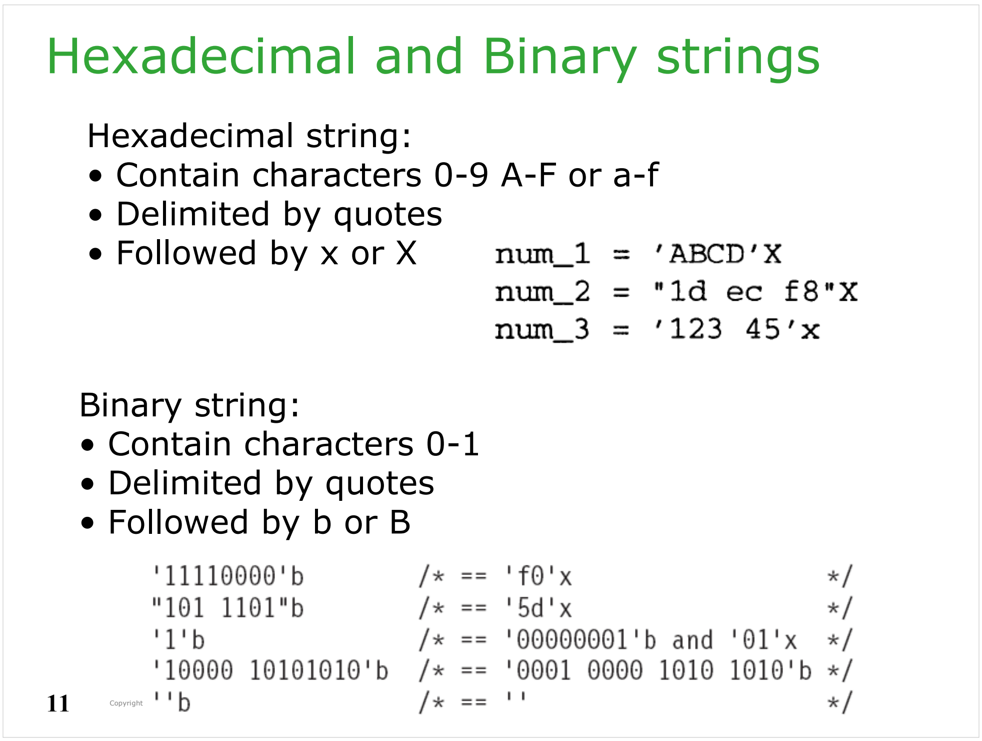
Literal Strings examples



```
SAY "12345"
SAY 'abc'
SAY "ABC"
SAY "Please enter a number"
SAY "Your name is 'Bob Flemming'"
```



```
12345
abc
ABC
Please enter a number
Your name is 'Bob Flemming'
***
```



Hexadecimal and Binary strings

Hexadecimal string:
- Contain characters 0-9 A-F or a-f
- Delimited by quotes
- Followed by x or X

```
num_1 = 'ABCD'X
num_2 = "1d ec f8"X
num_3 = '123 45'x
```

Binary string:
- Contain characters 0-1
- Delimited by quotes
- Followed by b or B

```
'11110000'b         /* == 'f0'x                      */
"101 1101"b         /* == '5d'x                      */
'1'b                /* == '00000001'b and '01'x   */
'10000 10101010'b   /* == '0001 0000 1010 1010'b */
''b                 /* == ''                         */
```

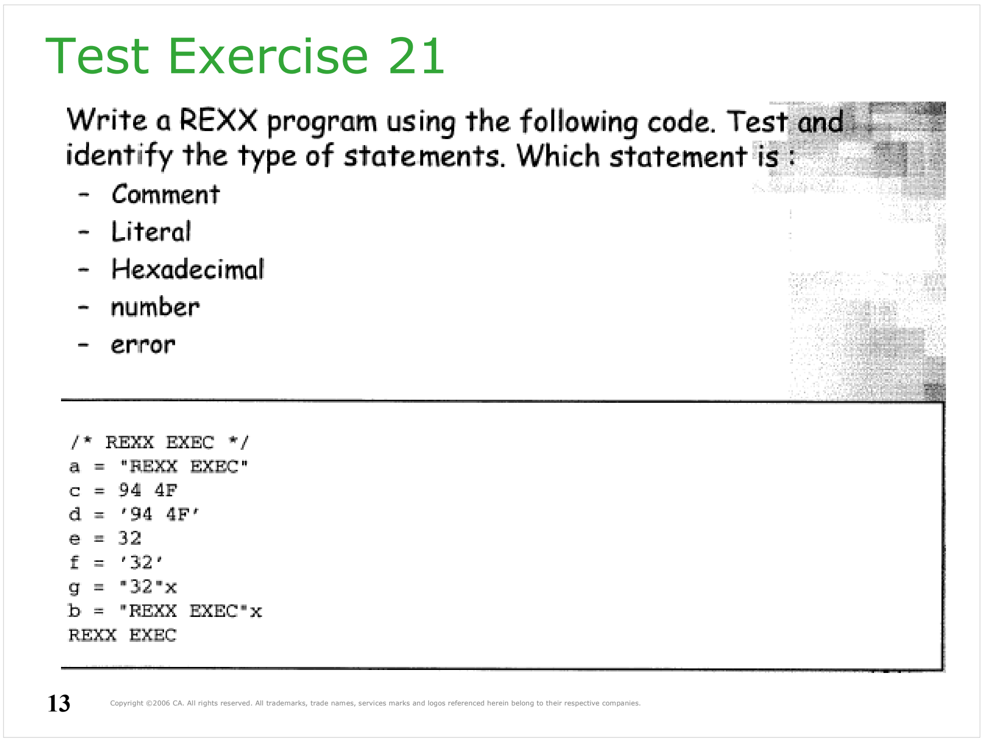
11

Hexadecimal and binary strings can be treated as literal strings.

Numbers

- Character strings that consist of one or more decimal digits
- Optionally prefixed by a plus or minus
- Can include decimal point
- can be exponential notation

```
num_1 = 29
num_2 = -94.3
num_3 = 53.312E23
num_4 = +0.5
num_5 = 0.73E-7
```



Test Exercise 21

Write a REXX program using the following code. Test and identify the type of statements. Which statement is :

- Comment
- Literal
- Hexadecimal
- number
- error

```
/* REXX EXEC */
a = "REXX EXEC"
c = 94 4F
d = '94 4F'
e = 32
f = '32'
g = "32"x
b = "REXX EXEC"x
REXX EXEC
```

13

Write and test it.

See 'MCOE.REXA.REXX(RX201212)'

Symbols

- A group of any characters selected from the English alphabetic and numeric characters (A-Z, a-z, 0-9) and/or from the characters .!? And _
- Symbols are used to name variables, functions, instructions, labels etc.

```
a = ALLEN
b = this is a symbol!
c = hello?
test_routine:
```

This course has been prepared by Milos Forman for MCoE needs only!

Simple Variables

- A variable is a name that represents a value.
- First character cannot be a number or a period
- Variable names must be less than 250 characters

```
salary = pay + benefit + bonus
job_class = "A"
msg_class = "X"
operator = current_operator_for_this_task
```

This course has been prepared by Milos Forman for MCoE needs only!

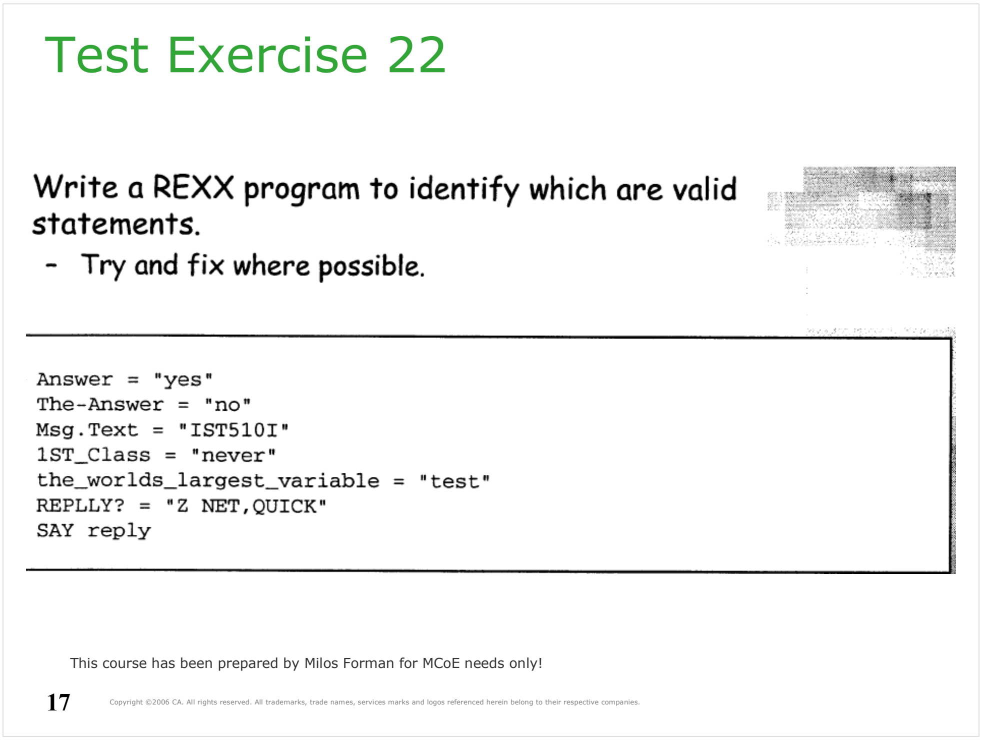
Simple Variables

- Variable names can only consist of the following :
  - A-Z, 0-9, a-z, @#$?!._
- Variable names are converted to upper case for comparison
- Variables are initialised to their own name in upper case.

```
SAY pay_roll
```

```
PAY_ROLL
***
```

This course has been prepared by Milos Forman for MCoE needs only!



Test Exercise 22

Write a REXX program to identify which are valid statements.

- Try and fix where possible.

```
Answer = "yes"
The-Answer = "no"
Msg.Text = "IST510I"
1ST_Class = "never"
the_worlds_largest_variable = "test"
REPLLY? = "Z NET,QUICK"
SAY reply
```

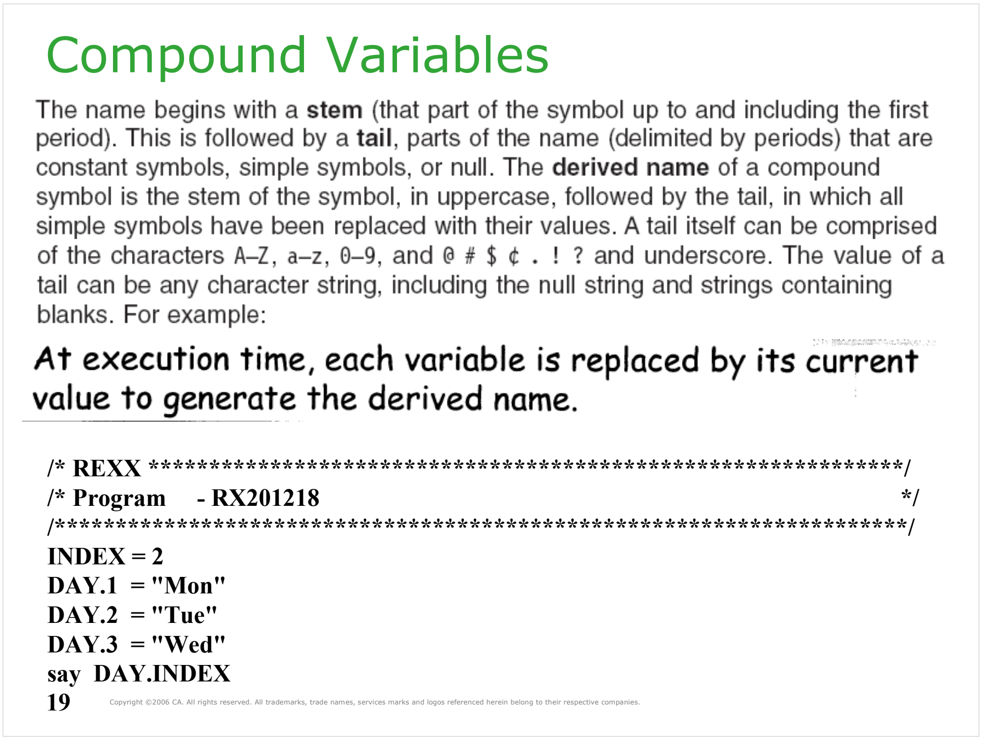
Write and test it.

See ‘MCOE.REXA.REXX(RX201216)’

Compound Variables

- A technique for grouping variables
- Start with a stem which is a valid variable name followed by a period (.)
- Characters after the first period are called tail

```
CPUTIME.ASID
Stem    Tail

new_rate.shift.type.weekly
Stem      Tail   Tail Tail
```



Compound Variables

The name begins with a **stem** (that part of the symbol up to and including the first period). This is followed by a **tail**, parts of the name (delimited by periods) that are constant symbols, simple symbols, or null. The **derived name** of a compound symbol is the stem of the symbol, in uppercase, followed by the tail, in which all simple symbols have been replaced with their values. A tail itself can be comprised of the characters `A–Z`, `a–z`, `0–9`, and `@ # $ ¢ . ! ?` and underscore. The value of a tail can be any character string, including the null string and strings containing blanks. For example:

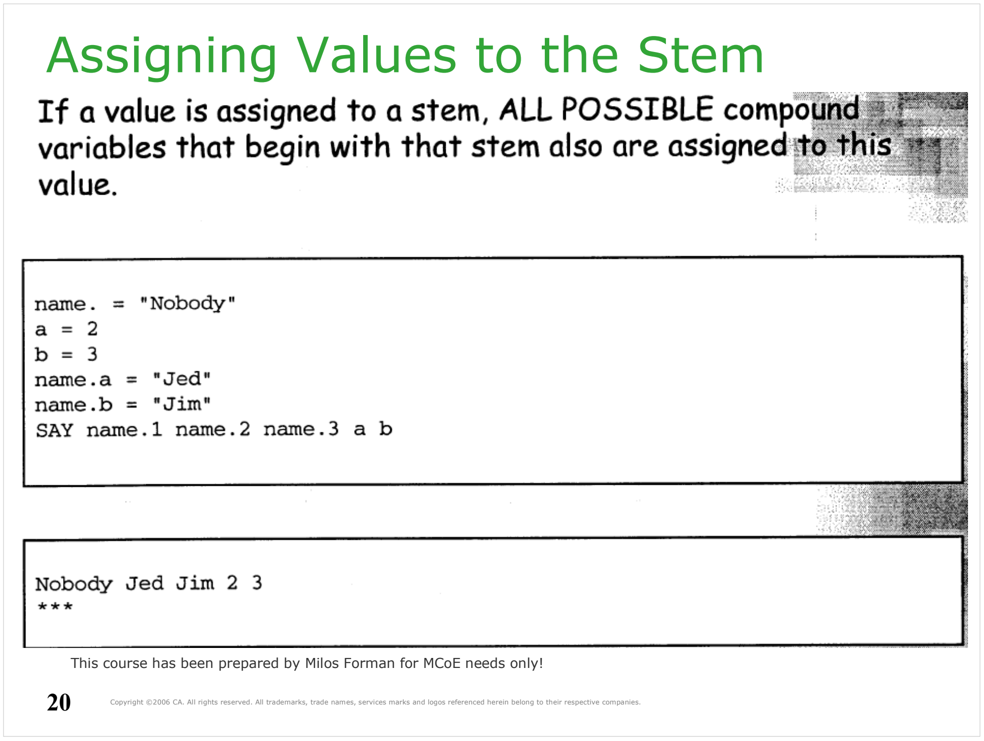
At execution time, each variable is replaced by its current value to generate the derived name.

```
/* REXX ****************************************************************/
/* Program    - RX201218                                              */
/***********************************************************************/
INDEX = 2
DAY.1  = "Mon"
DAY.2  = "Tue"
DAY.3  = "Wed"
say  DAY.INDEX
```

19

Write and test it.

See ‘MCOE.REXA.REXX(RX201218)’



Assigning Values to the Stem

If a value is assigned to a stem, ALL POSSIBLE compound variables that begin with that stem also are assigned to this value.

```
name. = "Nobody"
a = 2
b = 3
name.a = "Jed"
name.b = "Jim"
SAY name.1 name.2 name.3 a b
```

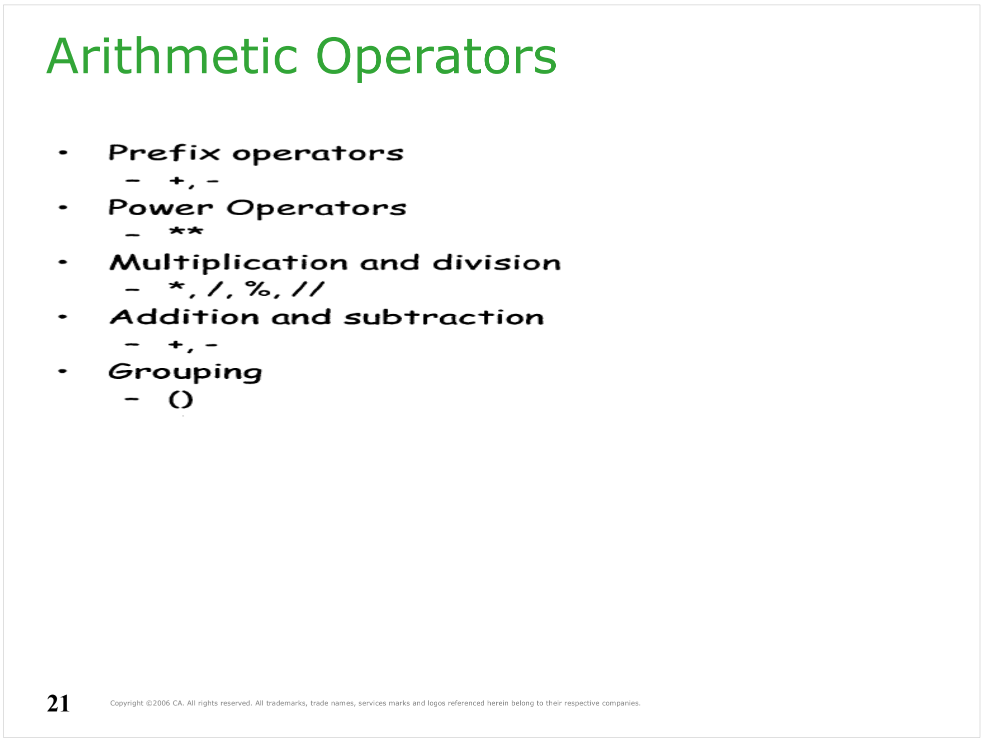
```
Nobody Jed Jim 2 3
***
```

This course has been prepared by Milos Forman for MCoE needs only!

20

Write and test it.

See ‘MCOE.REXA.REXX(RX201219)’



Arithmetic Operators

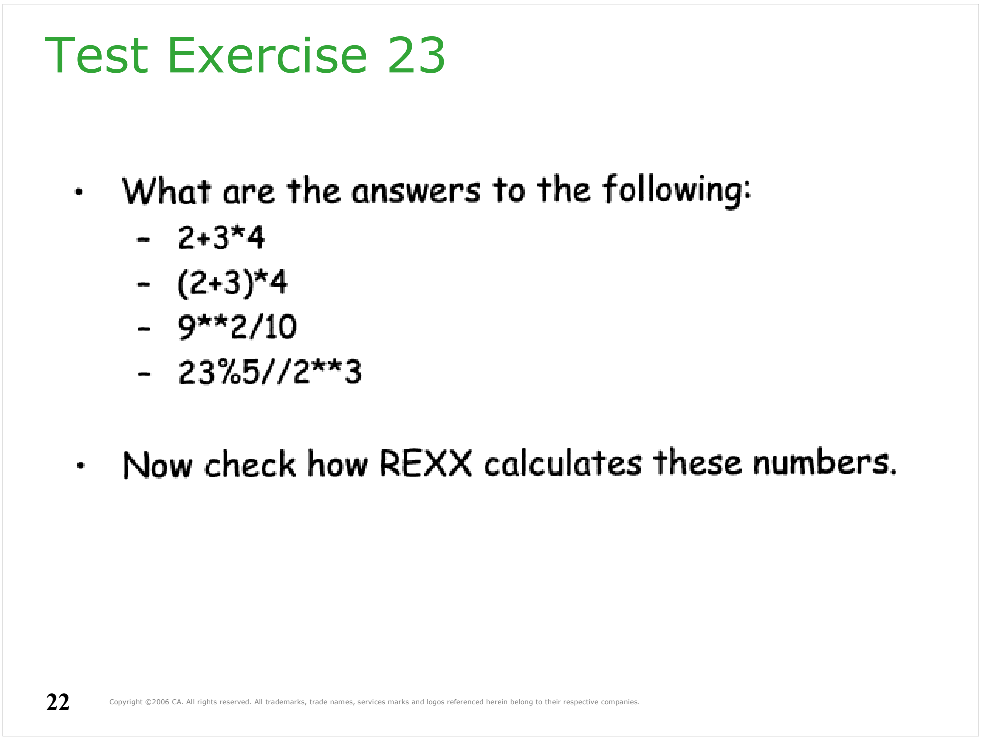
- Prefix operators
  - +, -
- Power Operators
  - **
- Multiplication and division
  - *, /, %, //
- Addition and subtraction
  - +, -
- Grouping
  - ()

21

The order of evaluation:

1. Expressions in parenthesis are evaluated first
2. Prefix operators – and + if present for numbers in expression
3. Exponentiation **
4. Multiplication and division in this order * / % //
5. Addition and subtraction + and -

You can test it using ‘MCOE.REXA.REXX(KALK)’



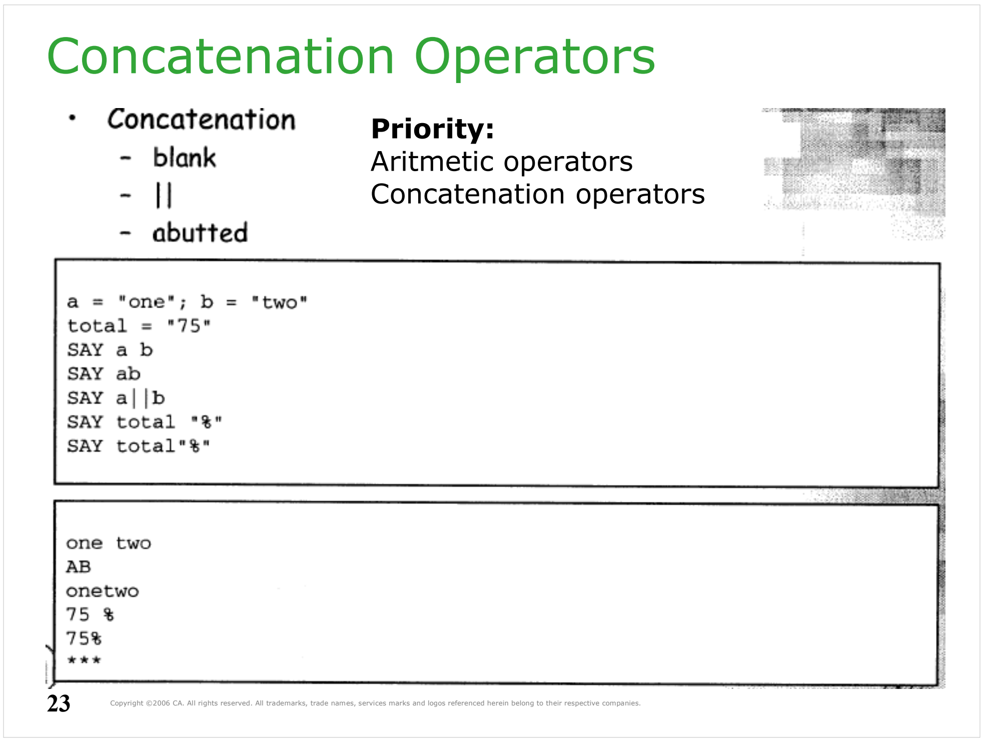
Test Exercise 23

- What are the answers to the following:
  - 2+3*4
  - (2+3)*4
  - 9**2/10
  - 23%5//2**3
- Now check how REXX calculates these numbers.

22

**	exponentiation
%	whole number division
//	rest after whole number division

Check it using KALK: 14   20   8,1   4



Concatenation Operators

- Concatenation
  - blank
  - ||
  - abutted

Priority:
Aritmetic operators
Concatenation operators

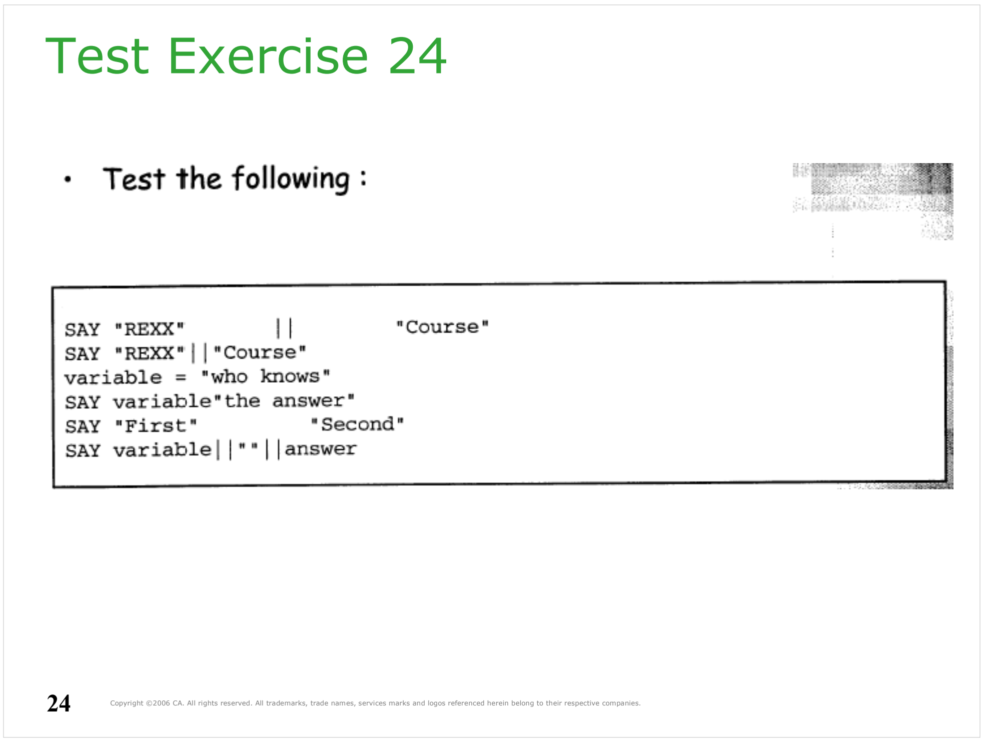
```
a = "one"; b = "two"
total = "75"
SAY a b
SAY ab
SAY a||b
SAY total "%"
SAY total"%"
```

```
one two
AB
onetwo
75 %
75%
***
```

23

Write it and test it.

Abutted – přiléhající, hraničící.

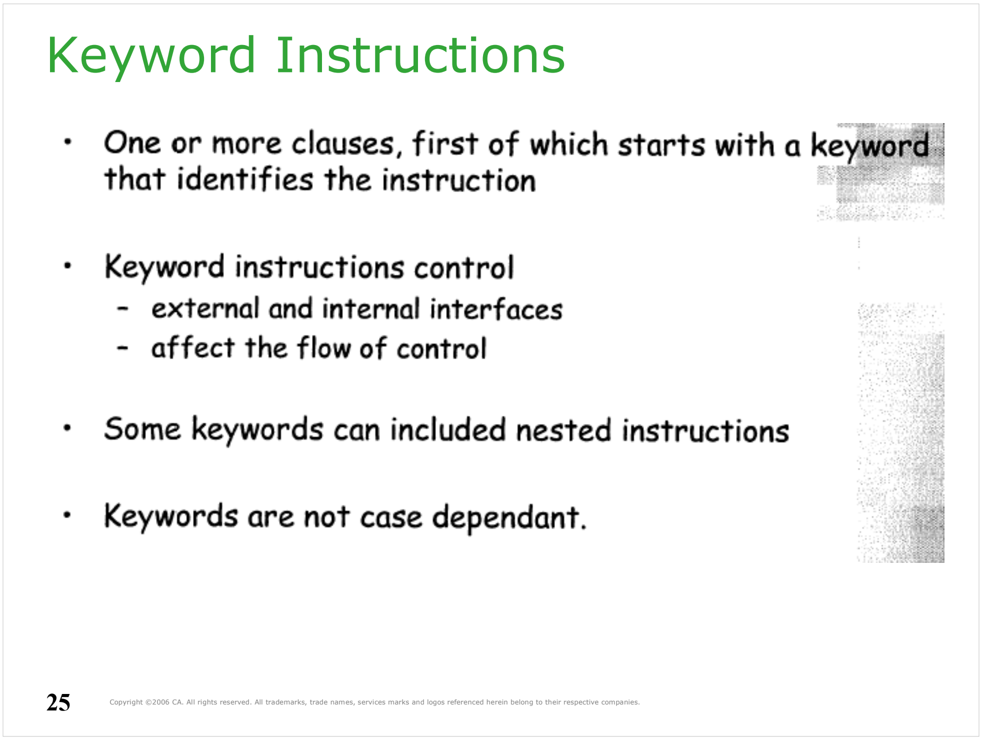


Test Exercise 24

- Test the following :

```
SAY "REXX"       ||         "Course"
SAY "REXX"||"Course"
variable = "who knows"
SAY variable"the answer"
SAY "First"         "Second"
SAY variable||""||answer
```

See ‘MCOE.REXA.REXX(RX201223)’

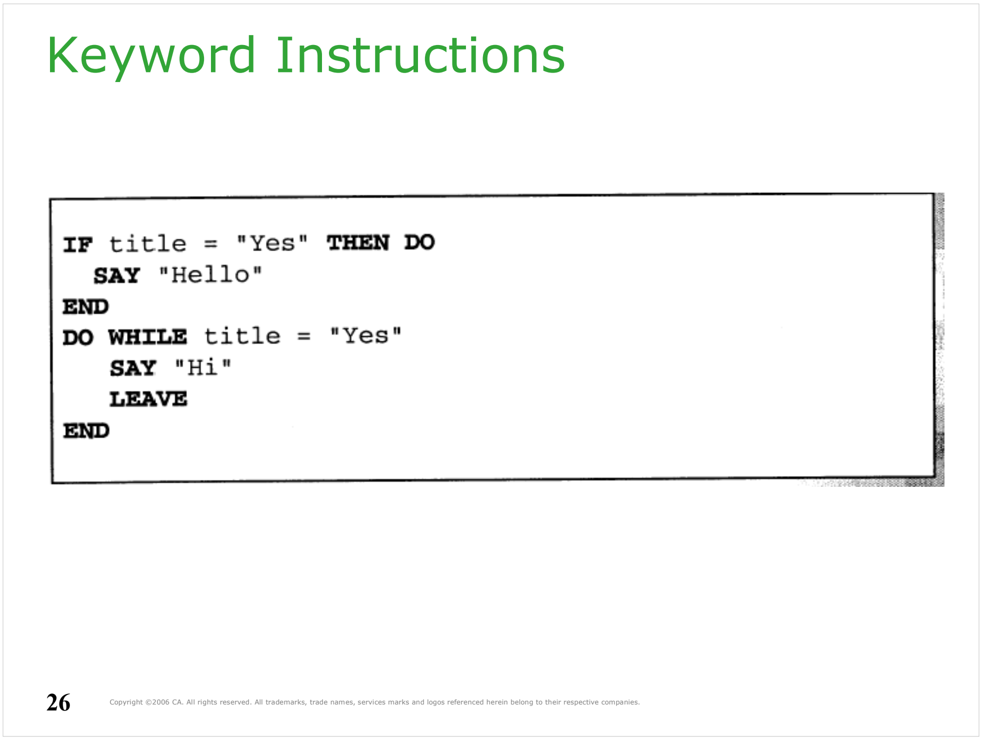


Keyword Instructions

- One or more clauses, first of which starts with a keyword that identifies the instruction
- Keyword instructions control
  - external and internal interfaces
  - affect the flow of control
- Some keywords can included nested instructions
- Keywords are not case dependant.

25

I am not an expert in a theory, let’s skip the slide to practice...

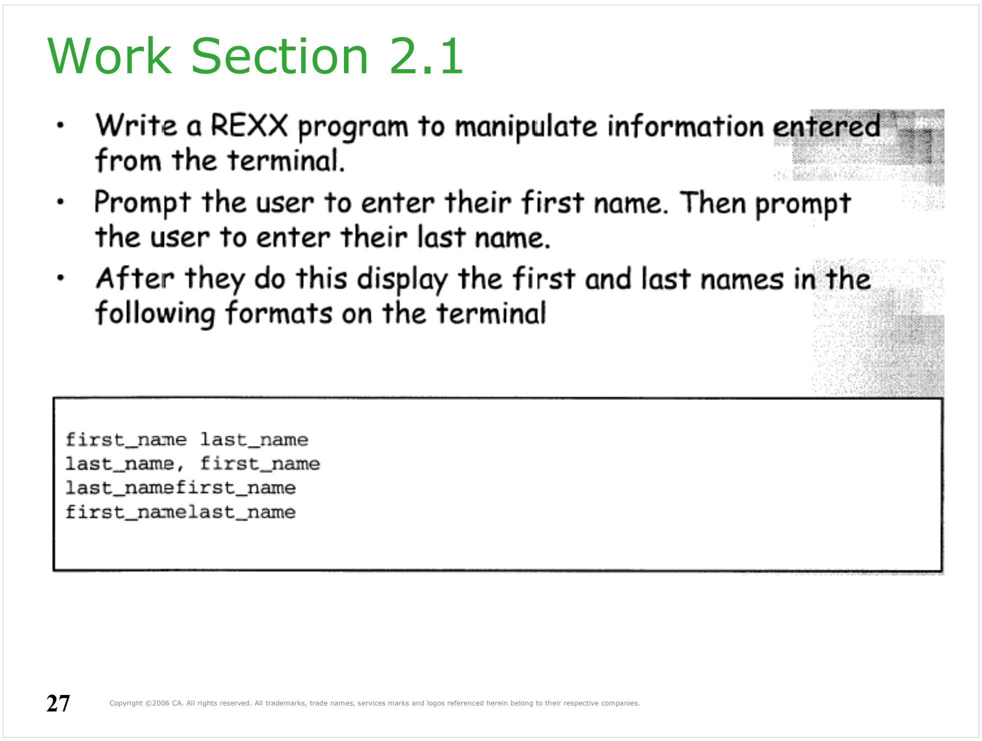


Keyword Instructions

```
IF title = "Yes" THEN DO
  SAY "Hello"
END
DO WHILE title = "Yes"
   SAY "Hi"
   LEAVE
END
```

26

Keyword instructions are bold.



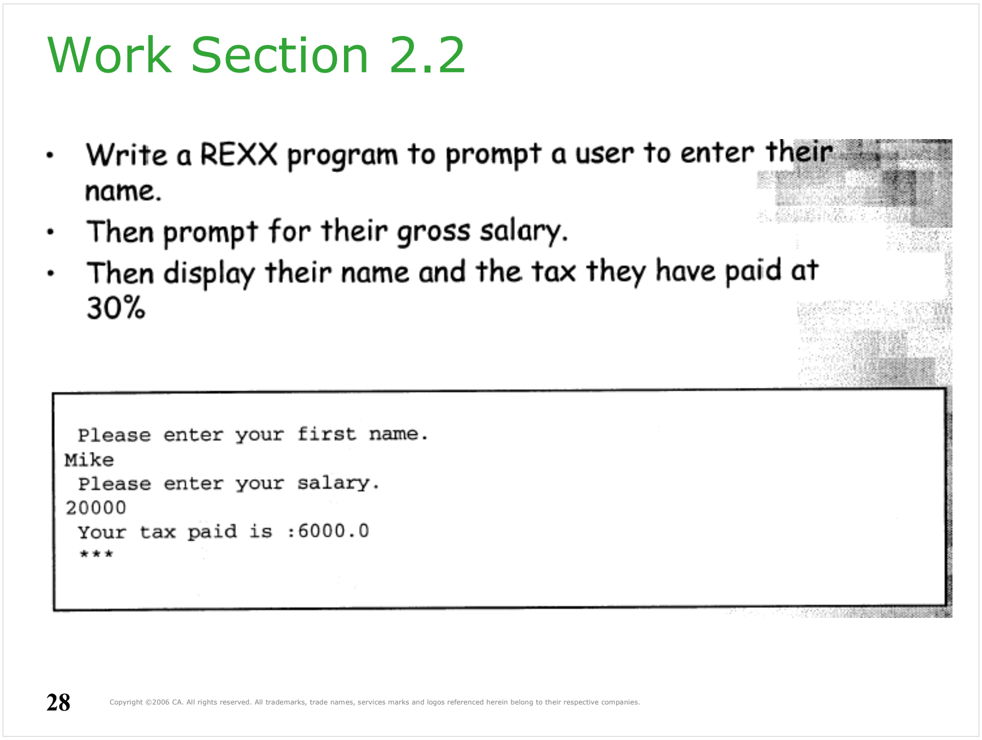
Work Section 2.1

- Write a REXX program to manipulate information entered from the terminal.
- Prompt the user to enter their first name. Then prompt the user to enter their last name.
- After they do this display the first and last names in the following formats on the terminal

```
first_name last_name
last_name, first_name
last_namefirst_name
first_namelast_name
```

27

What is the result from this?



Work Section 2.2

- Write a REXX program to prompt a user to enter their name.
- Then prompt for their gross salary.
- Then display their name and the tax they have paid at 30%

```
 Please enter your first name.
Mike
 Please enter your salary.
20000
 Your tax paid is :6000.0
 ***
```

28

What is the format of the result? Notice the format of **say** instruction.