Practice & Transfer of Learning in the Teaching of Software Testing

CSEET July 2007

Dublin

Cem Kaner, J.D., Ph.D.

Sowmya Padmanabhan, B.Eng., M.Sc.

Florida Institute of Technology

http://www.kaner.com/pdfs/CSEETdomain2007.pdf

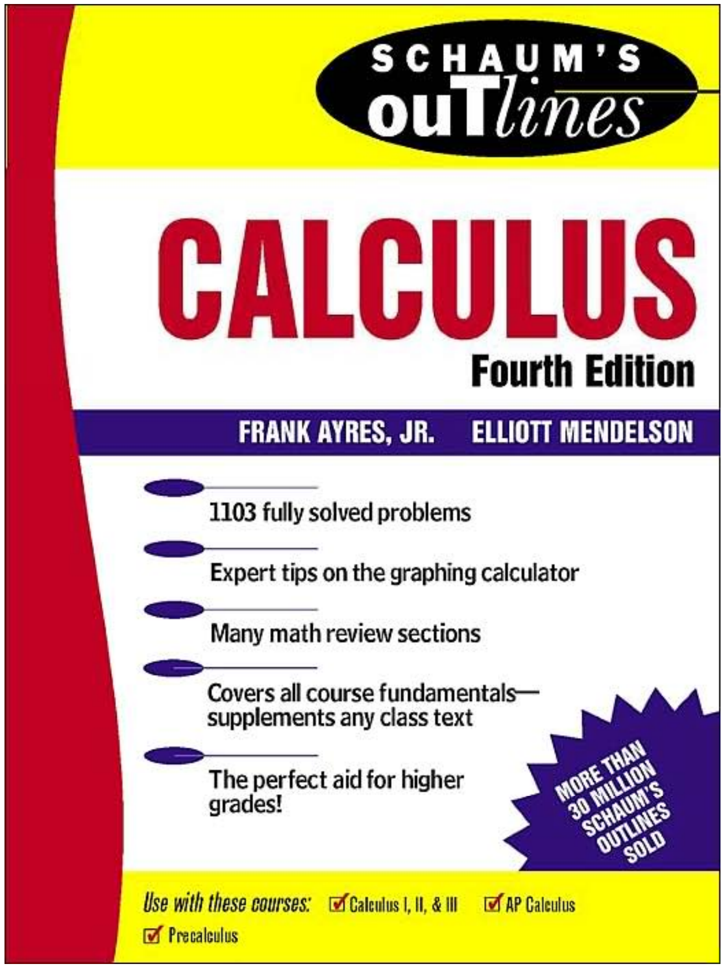
Overview

Overview: Teaching Domain Testing

In my courses:

- Students generally felt they understood domain testing after a brief lecture and one or two clear examples
- Diagnostic exercises presented a different picture:
  - Students **could** give back the definition / description / rationale
  - Students **could not** reliably apply the technique

Overview: Analogy to studying mathematics



SCHAUM'S ouTlines

CALCULUS

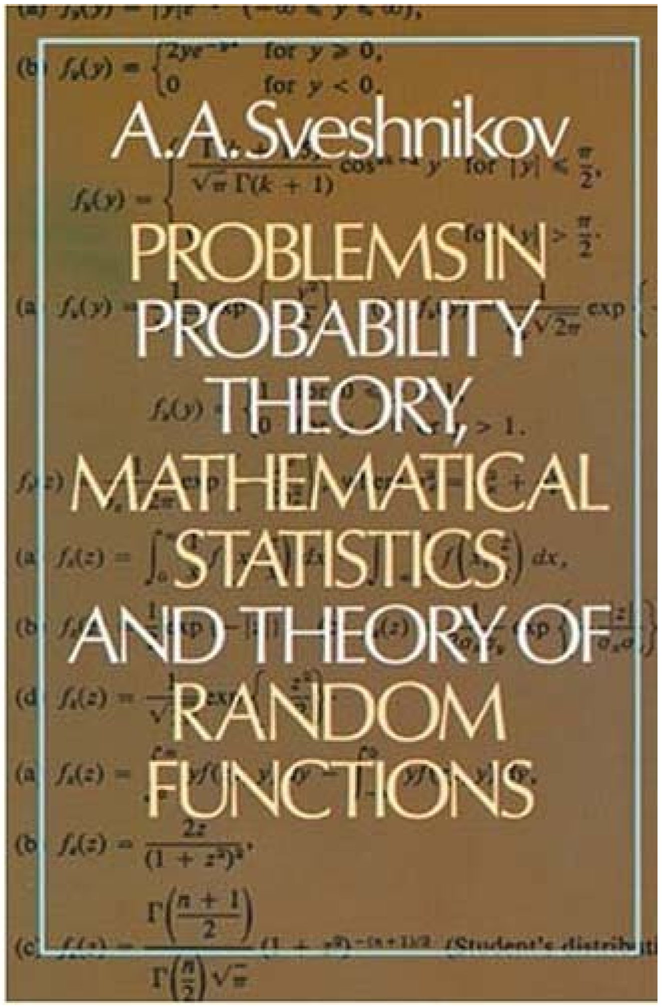
Fourth Edition

FRANK AYRES, JR. ELLIOTT MENDELSON

- 1103 fully solved problems
- Expert tips on the graphing calculator
- Many math review sections
- Covers all course fundamentals—supplements any class text
- The perfect aid for higher grades!

MORE THAN 30 MILLION SCHAUM'S OUTLINES SOLD

Use with these courses: ☑ Calculus I, II, & III ☑ AP Calculus ☑ Precalculus



A. A. Sveshnikov

PROBLEMS IN PROBABILITY THEORY, MATHEMATICAL STATISTICS AND THEORY OF RANDOM FUNCTIONS

Overview: Instructional approach

We tried a teaching style that combined:

- Lecture on the theory
- Procedural suggestions for approaching problems of this class
- Worked examples
- Many exercises, with feedback: Lots of practice

Our long-term goal:

- If this works, we can create large pool of test technique exercises and facilitate a strong improvement in the state of software testing education.

Overview: Result

- Test students using questions somewhat similar to the ones they practiced with:
  - Excellent results
- Test students using questions that require a slight stretch (apply the same principles but in a slightly different way)
  - Poor results
- This is shocking to many commercial trainers (my home community before coming back to university), who struggle to find time to add a few simple exercises.
- In retrospect, similar to traditional mathematics education
  - Students do well on exams
  - But cannot apply knowledge in later courses

Overview: My current puzzle

Should we abandon lecture+problem style in favor of a more constructivist approach?

1st year Java (co-teaching with other faculty):

- When we try emphasis on larger, meaningful assignments
  - Motivates some students, but dismays many others.
  - Serious dropout rate.
- Alternate style: Emphasis on narrowly-focused examples (many smaller exercises)
  - Seems easier
  - Seems to foster self-confidence & faster connection with basics
  - But as with domain testing, indicators of weak transfer.
- I think I'm looking for a balance, rather than abandonment of procedural instruction. ….

The Details

Software testing ...

... is a technical investigation

to provide stakeholders

with quality-related information

about a software product or service

Testers use empirical methods to learn about quality

Information objectives

- Find important bugs, to get them fixed
- Assess conformance to specifications
- Assess initial (e.g. first-day-of-use) customer experience
- Help managers make release decisions
- Block premature product releases
- Help predict and control costs of product support
- Check interoperability with other products
- Find safe scenarios for use of the product
- Certify the product meets a particular standard
- Ensure the testing process meets accountability standards
- Minimize the risk of safety-related lawsuits
- Help clients improve product quality & testability
- Help clients improve their processes
- Evaluate the product for a third party

Different objectives drive you toward different:

- Project mgmt styles
- Results reporting methods
- Politics on the project
- **Testing techniques**

Test techniques

A test technique is a heuristic for generating tests

"A heuristic is anything that provides a plausible aid or direction in the solution of a problem but is in the final analysis unjustified, incapable of justification, and fallible. It is used to guide, to discover and to reveal. … [A] heuristic has four signatures that make it easy to recognize:

- A heuristic does not guarantee a solution;
- It may contradict other heuristics;
- It reduces the search time in solving a problem; and
- Its acceptance depends on the immediate context instead of on an absolute standard."
  - Billy Vaughn Koen,
    Definition of the Engineering Method,
    (p. 5, 16-17), 1985.

A heuristic is a fallible but useful method for attempting to solve a problem or reach a decision.

Domain testing: The field's most popular technique

Select test cases through stratified sampling:

- **Identify the variable** of interest
- **Identify the set of imaginable values** of the variable (including "invalid" but not inconceivable)
- **Partition the set** into subsets that are in some way(s) comparable or equivalent
- **Select one (or a few) "best representatives" of each set**.
  - Most commonly used "best representative" is the boundary case, which catches inequality errors in specification of numeric ranges

Teaching domain testing

Teach students:

- the general principle
- how to partition
- how to select a best representative
- how to conduct the test
  - enter the value
  - look for bad results
    - ➢ immediate failure (problem with filter)
    - ➢ failure on use of the value

Common student errors

Consider an integer that can take on values from -999 to 999 inclusively

- ***Doesn't spot a boundary.***
- ***Offers excess values.*** Students offer 998 as well as the appropriate 999 and 1000.
- ***Doesn't spot a dimension.*** (a) how many characters should this field handle? Same for positive and negative numbers? (b) if you delay after entering the first character, is there a risk of time-out? What delay durations should you test? Boundaries?
- ***Doesn't articulate a risk.*** Suppose we explicitly ask students to identify a risk and then identify relevant variable(s) and a powerful test appropriate to the risk. Rather than describe how the program might fail, the student might reiterate the test or make vague statements, like "fail to process this value correctly."
- ***Doesn't explain how a test case relates to a stated risk.*** When an assignment calls for such an explanation, students may respond inarticulately or irrelevantly.
- ***Doesn't consider a consequence.*** In real life (and in some of our test questions), the tester can determine more information than the bare range of an input field. The program will do something with the data entered. It is important, for each of those uses, to check whether the bounds imposed by the input filter are appropriate to the later use, and what consequence will result if they are not.
- ***Poor generalization.*** In more complex questions than the integer example here, students often pick inappropriate variables for analysis, such as treating each value of a binary variable as the best representative of its own 1-member class.

Common errors

Students have learned the basic idea

- Bloom's taxonomy lower levels: know / explain

Students don't have a higher-level understanding

- apply / analyze / think through what they are learning

How can we increase their depth of understanding?

Our teaching strategy: Extensive practice

- Teach the basic principle by lecture
- Procedural tips for solving the problem
- Applied to different types of variables, such as:
  - Integers
  - Dollars
  - Floats
  - Strings
  - Records (non-primitive data types)
- Students work through exercises, from simple explicit cases to “word problems.”
- Expectation: Diversified practice will lead to skill and to transfer of learning to real application

Details of the experiment

http://www.testingeducation.org/a/DTD&C.pdf

Sowmya Padmanabhan's M.Sc. Thesis

Domain Testing: Divide and Conquer

661 pages, including:

- Instructional materials
- Practice exercises / examples
- Final exam
- Assessments by practitioners (James Bach, Pat McGee, Cem Kaner)

The course

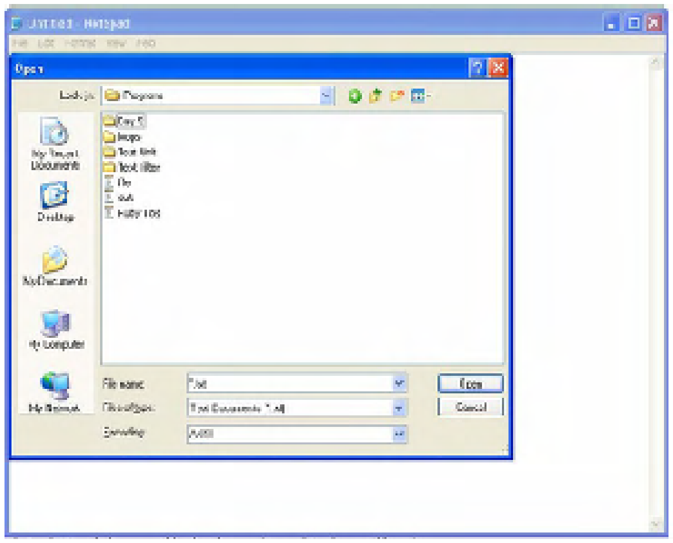
- 18 classroom hours (5 days)
- 23 paid *learners* (undergrad and grad) who had not taken the testing course but had completed the course prerequisites (discrete math; 2 semesters of programming).
- Taught in 5 replications, 4-5 learners per replication
- 90-minute pre-test (Pre-test A or Pre-test B)
- Classes with many exercises and tests
- 90-minute post-test (Pre-test B or Pre-test A)
- Performance test

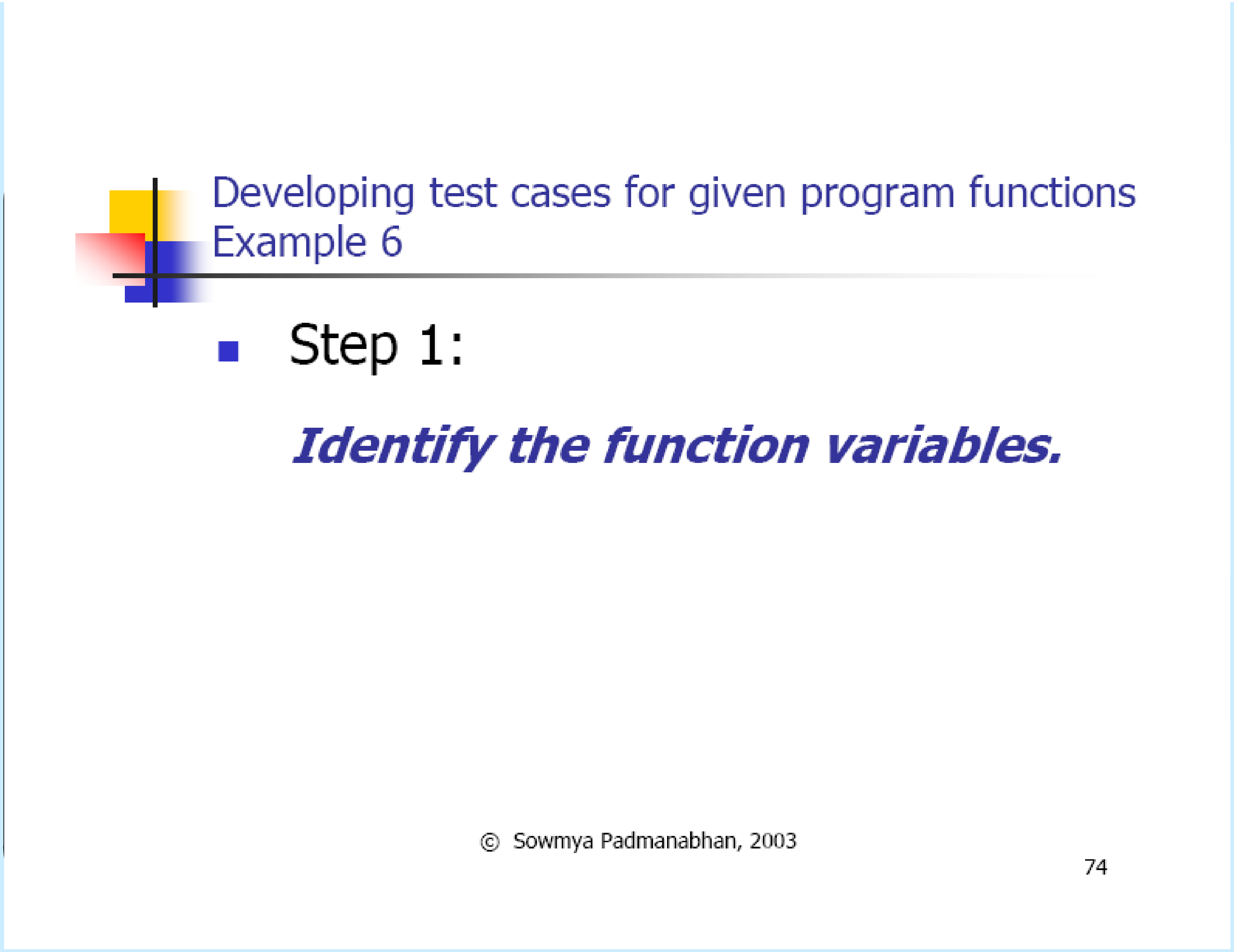
Sample slides
(Pardon my rush ...)

Developing test cases for given program functions

- Example 6:

For the 'File Open' function of the notepad program, identify its variables, the dimensions, data type along each dimension. Develop a series of test cases by performing equivalence class analysis and boundary-value analysis on each of the variables. A screenshot of the function is provided on the right hand side.

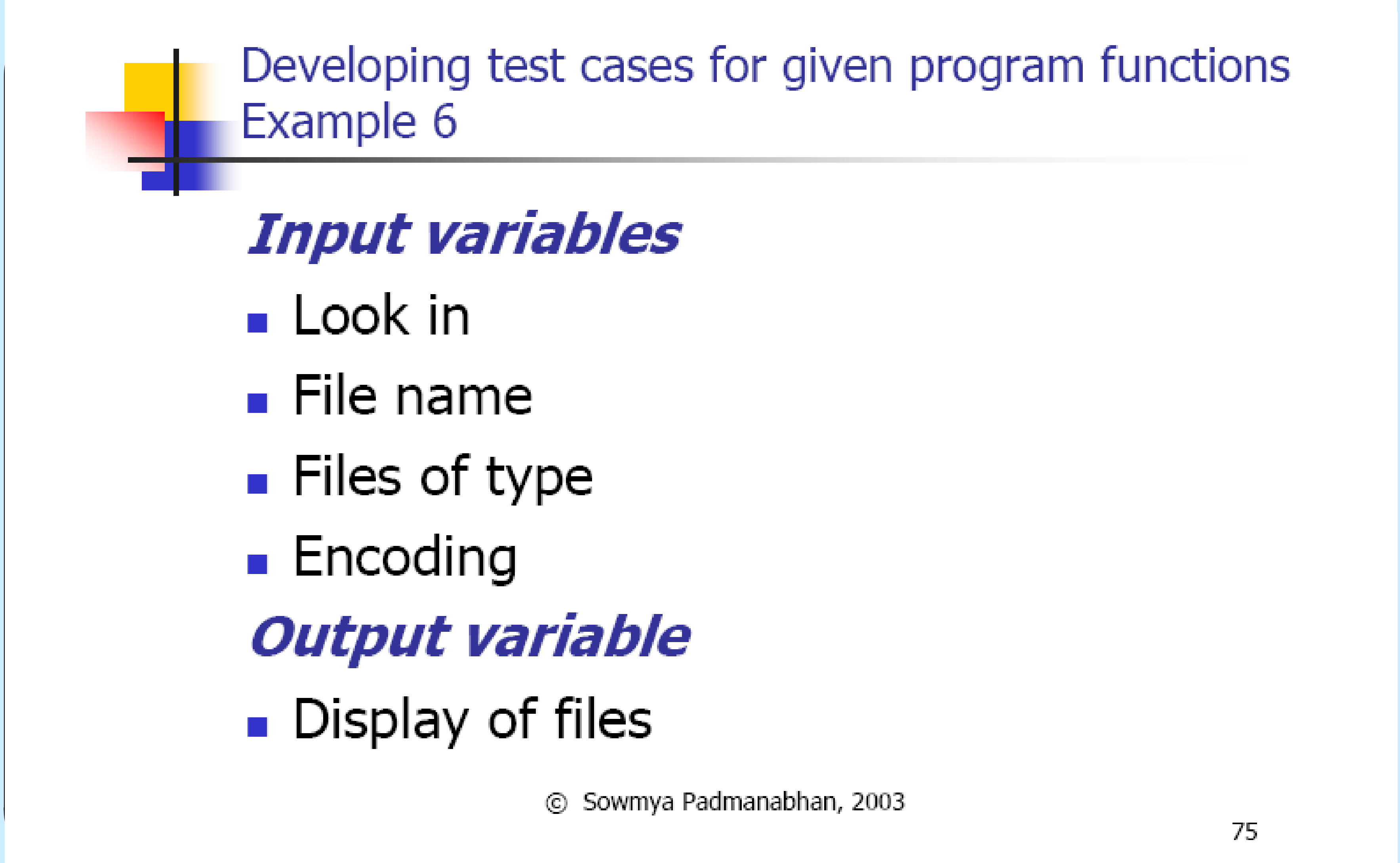




Developing test cases for given program functions Example 6

- Step 1:

  Identify the function variables.



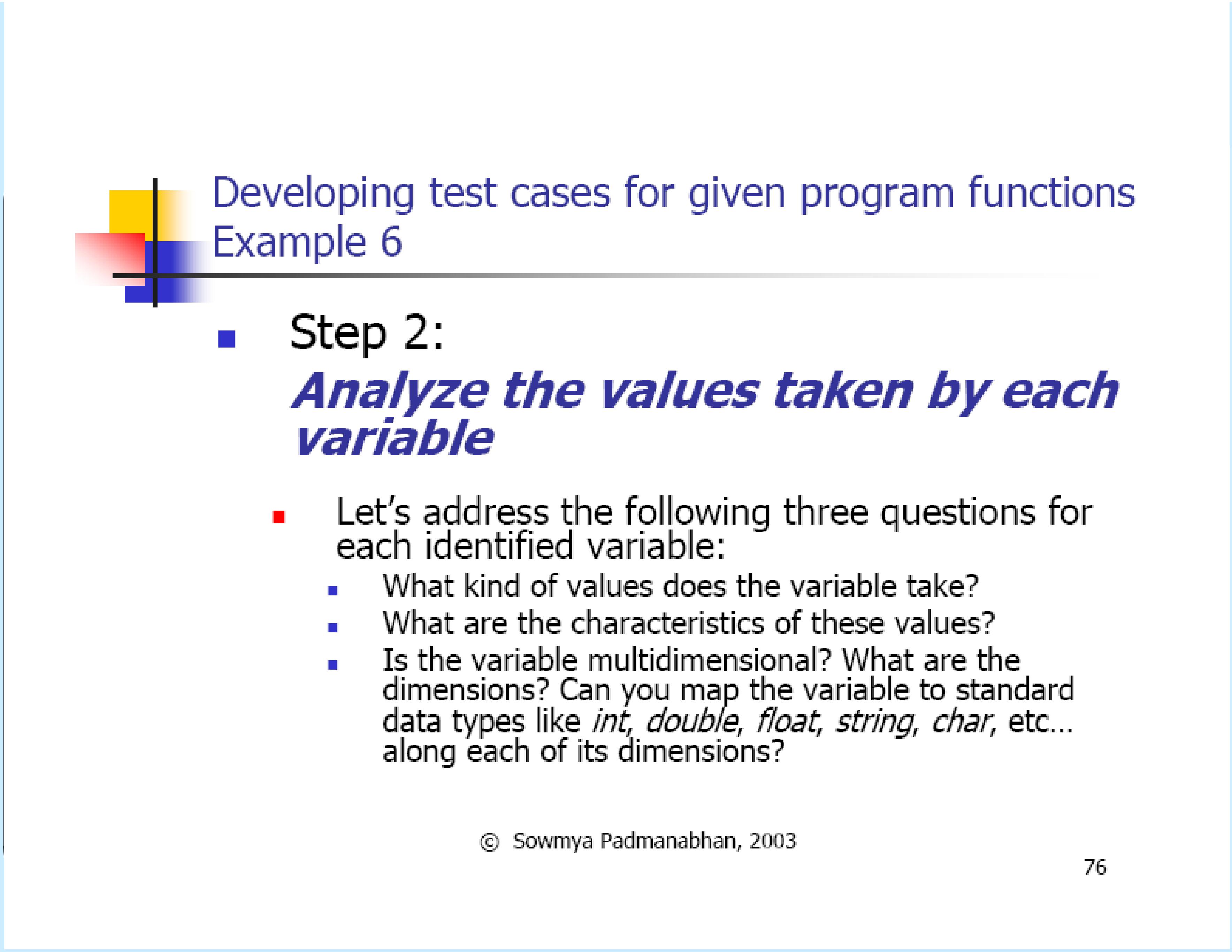
Developing test cases for given program functions Example 6

Input variables

- Look in
- File name
- Files of type
- Encoding

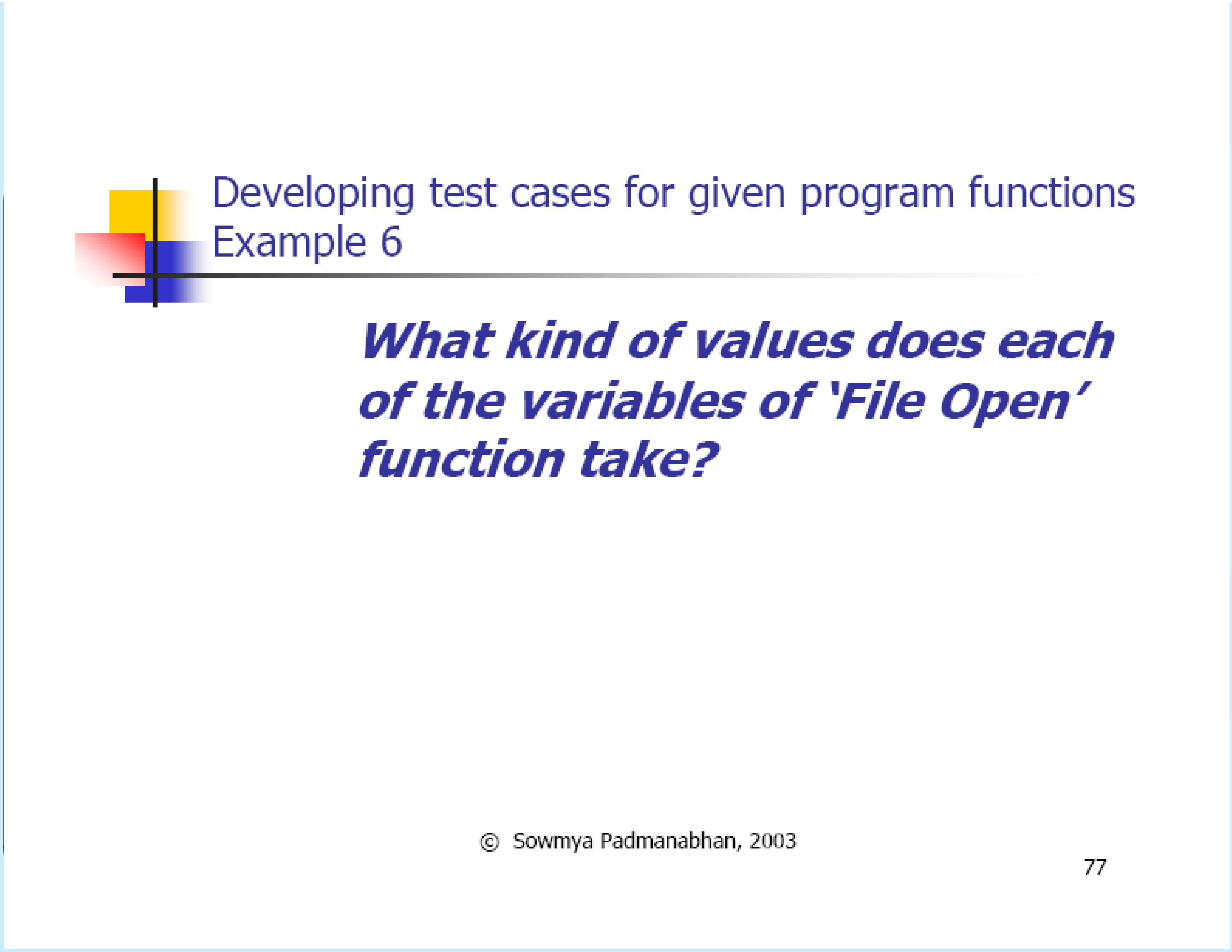
Output variable

- Display of files



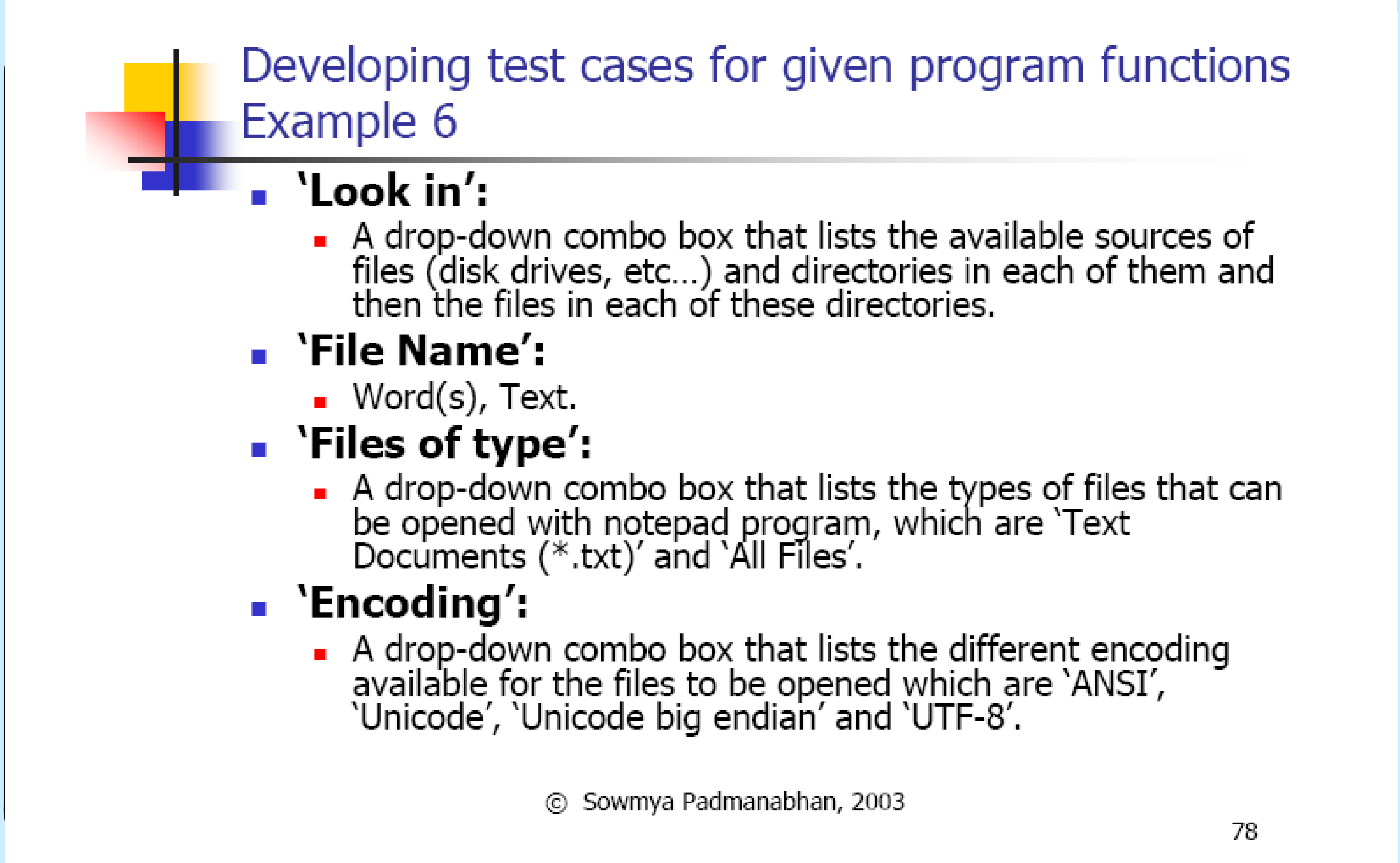
Developing test cases for given program functions Example 6

- Step 2:
  Analyze the values taken by each variable
  - Let's address the following three questions for each identified variable:
    - What kind of values does the variable take?
    - What are the characteristics of these values?
    - Is the variable multidimensional? What are the dimensions? Can you map the variable to standard data types like *int*, *double*, *float*, *string*, *char*, etc… along each of its dimensions?



Developing test cases for given program functions Example 6

What kind of values does each of the variables of 'File Open' function take?



Developing test cases for given program functions Example 6

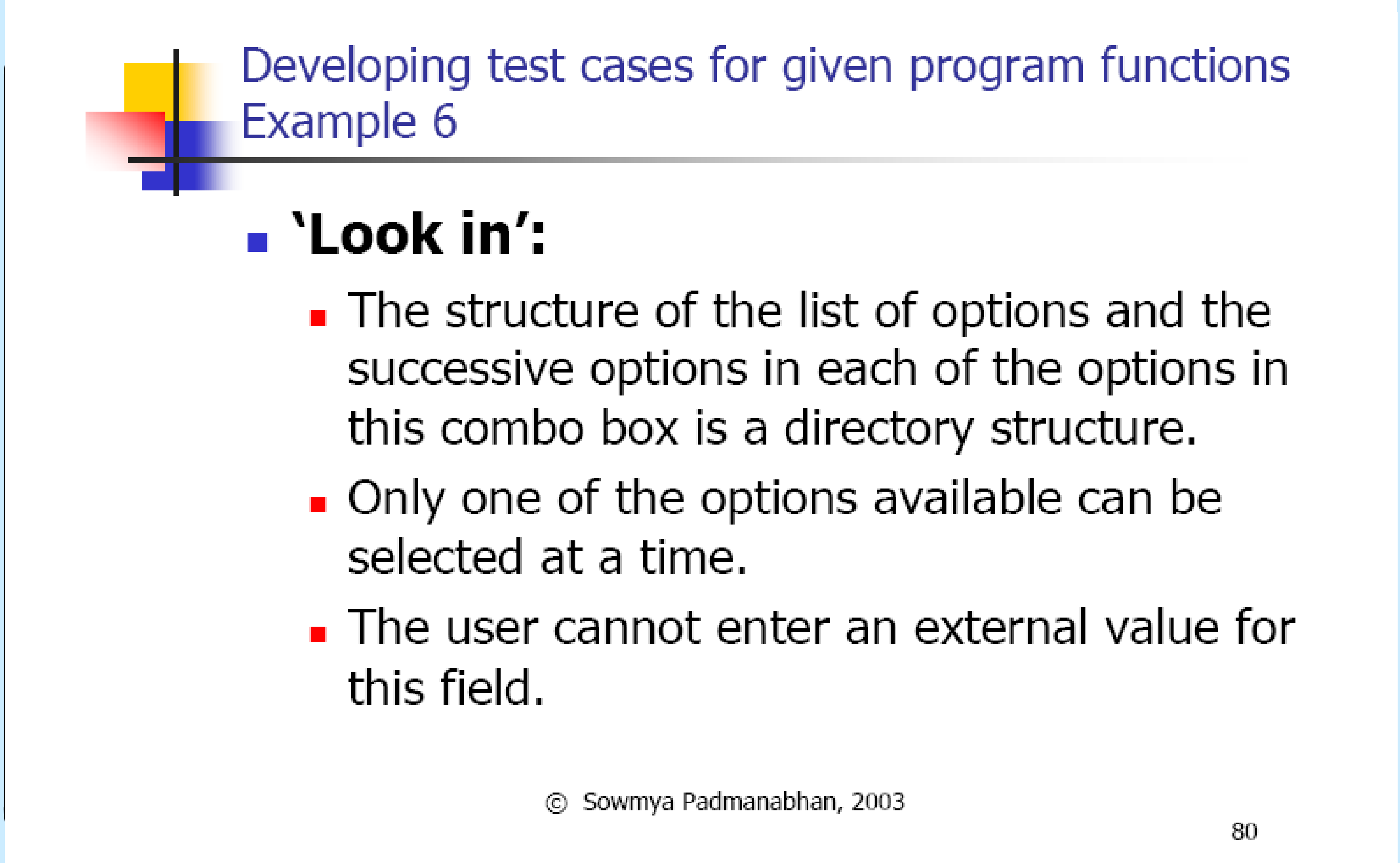
- **'Look in':**
  - A drop-down combo box that lists the available sources of files (disk drives, etc...) and directories in each of them and then the files in each of these directories.
- **'File Name':**
  - Word(s), Text.
- **'Files of type':**
  - A drop-down combo box that lists the types of files that can be opened with notepad program, which are 'Text Documents (*.txt)' and 'All Files'.
- **'Encoding':**
  - A drop-down combo box that lists the different encoding available for the files to be opened which are 'ANSI', 'Unicode', 'Unicode big endian' and 'UTF-8'.

© Sowmya Padmanabhan, 2003

78

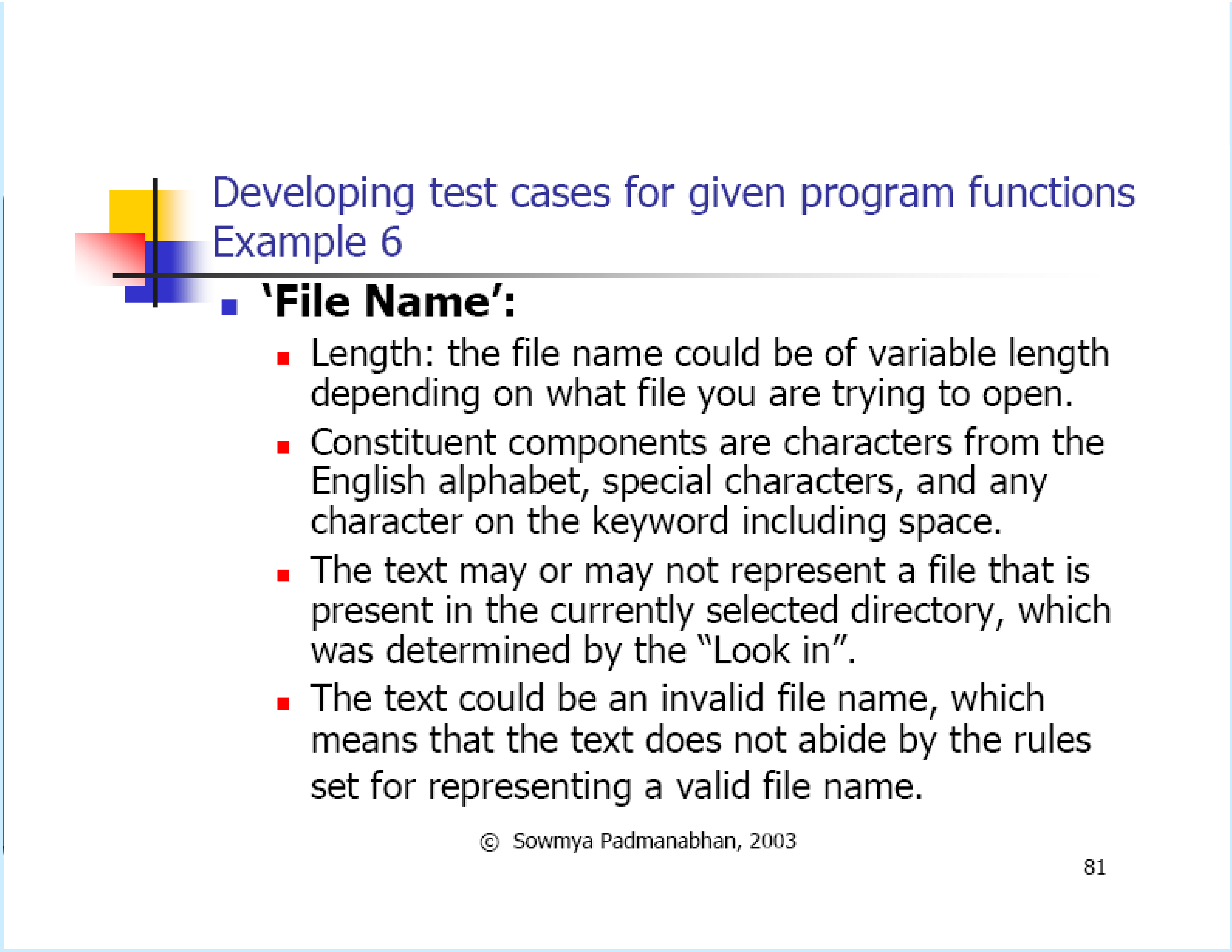
Developing test cases for given program functions Example 6

What are the characteristics of these values?



Developing test cases for given program functions Example 6

- **'Look in':**
  - The structure of the list of options and the successive options in each of the options in this combo box is a directory structure.
  - Only one of the options available can be selected at a time.
  - The user cannot enter an external value for this field.

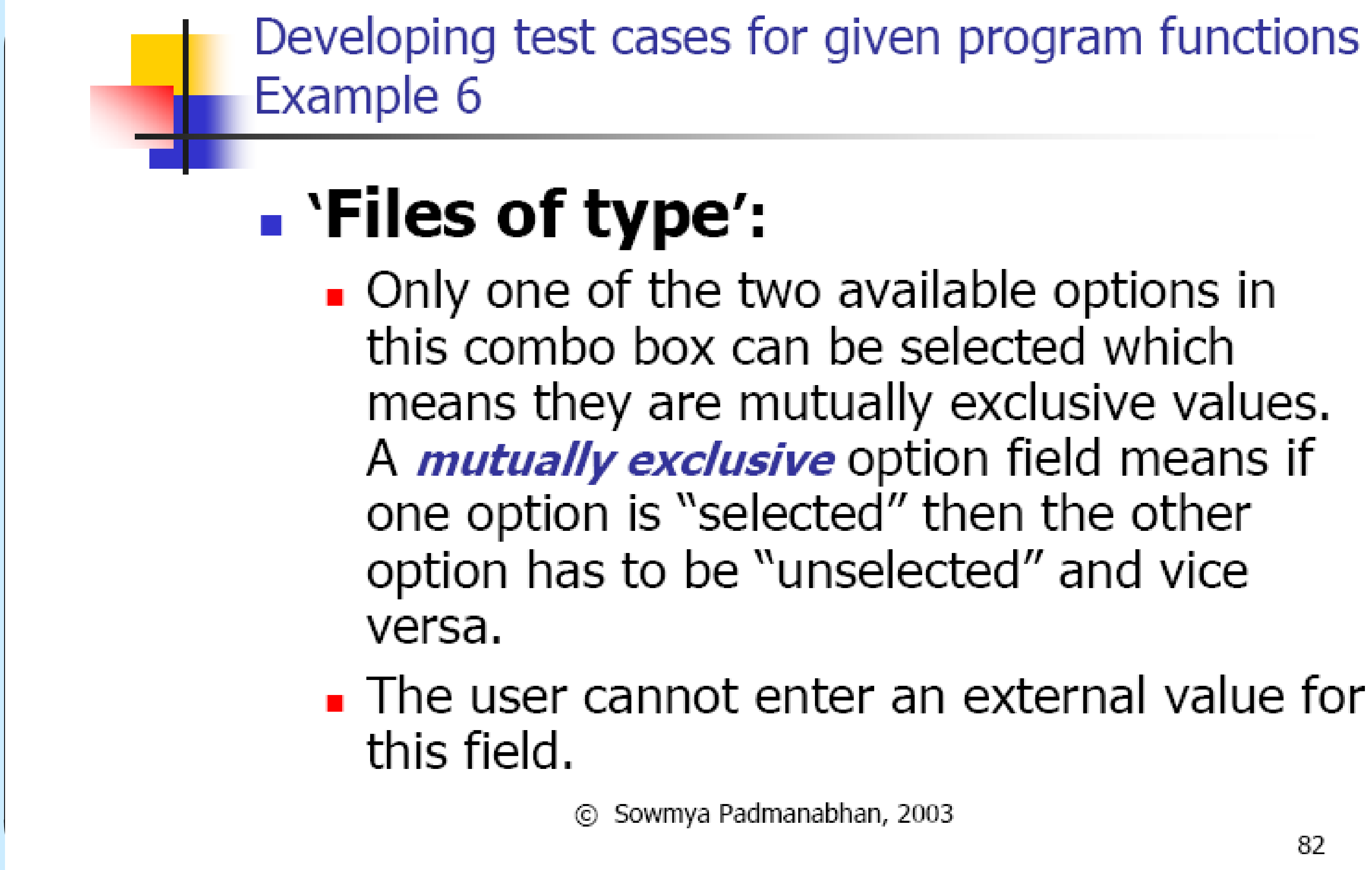


Developing test cases for given program functions Example 6

- **‘File Name’:**
  - Length: the file name could be of variable length depending on what file you are trying to open.
  - Constituent components are characters from the English alphabet, special characters, and any character on the keyword including space.
  - The text may or may not represent a file that is present in the currently selected directory, which was determined by the “Look in”.
  - The text could be an invalid file name, which means that the text does not abide by the rules set for representing a valid file name.

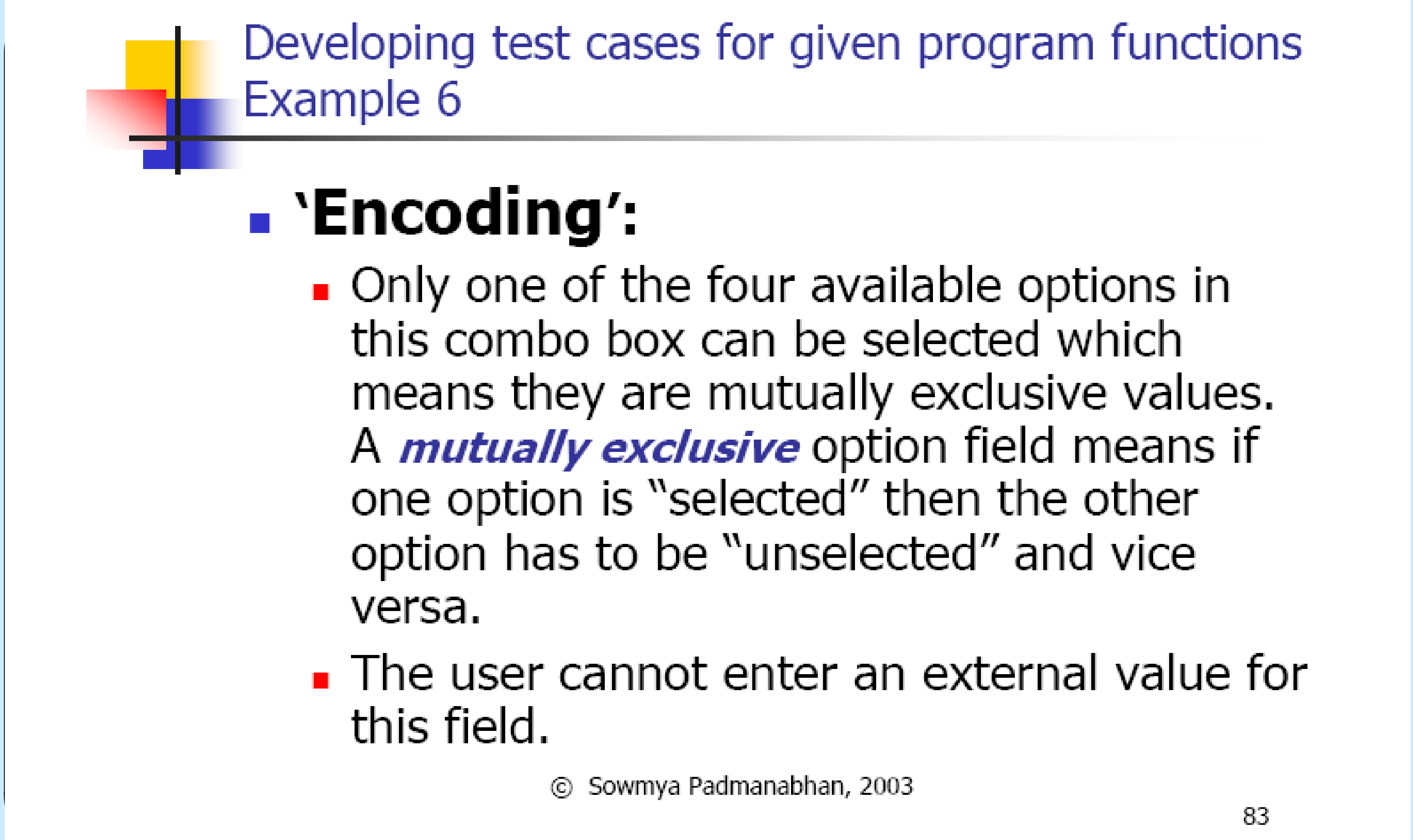
© Sowmya Padmanabhan, 2003

81



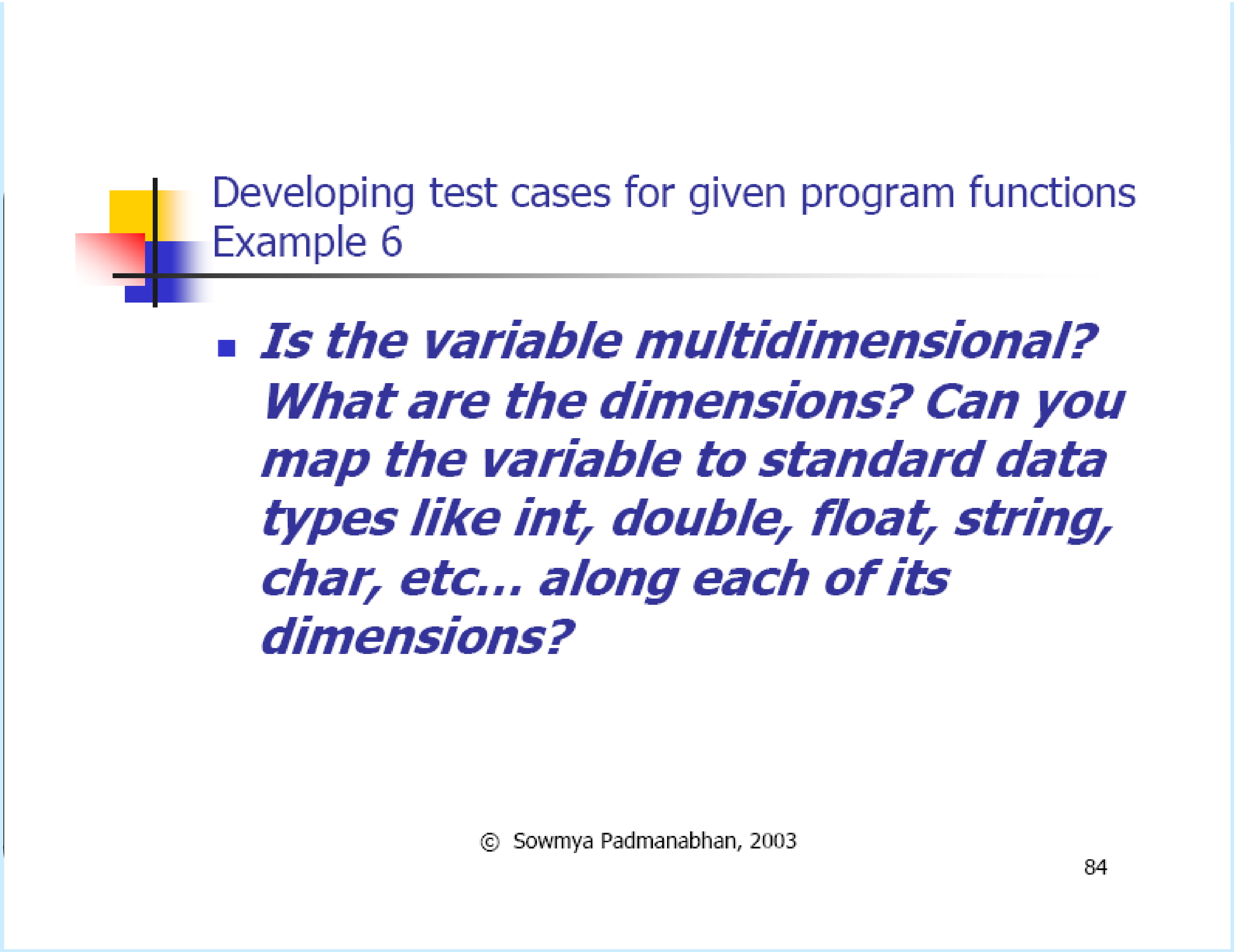
Developing test cases for given program functions Example 6

- **‘Files of type’:**
  - Only one of the two available options in this combo box can be selected which means they are mutually exclusive values. A ***mutually exclusive*** option field means if one option is “selected” then the other option has to be “unselected” and vice versa.
  - The user cannot enter an external value for this field.



Developing test cases for given program functions Example 6

- **'Encoding':**
  - Only one of the four available options in this combo box can be selected which means they are mutually exclusive values. A ***mutually exclusive*** option field means if one option is "selected" then the other option has to be "unselected" and vice versa.
  - The user cannot enter an external value for this field.

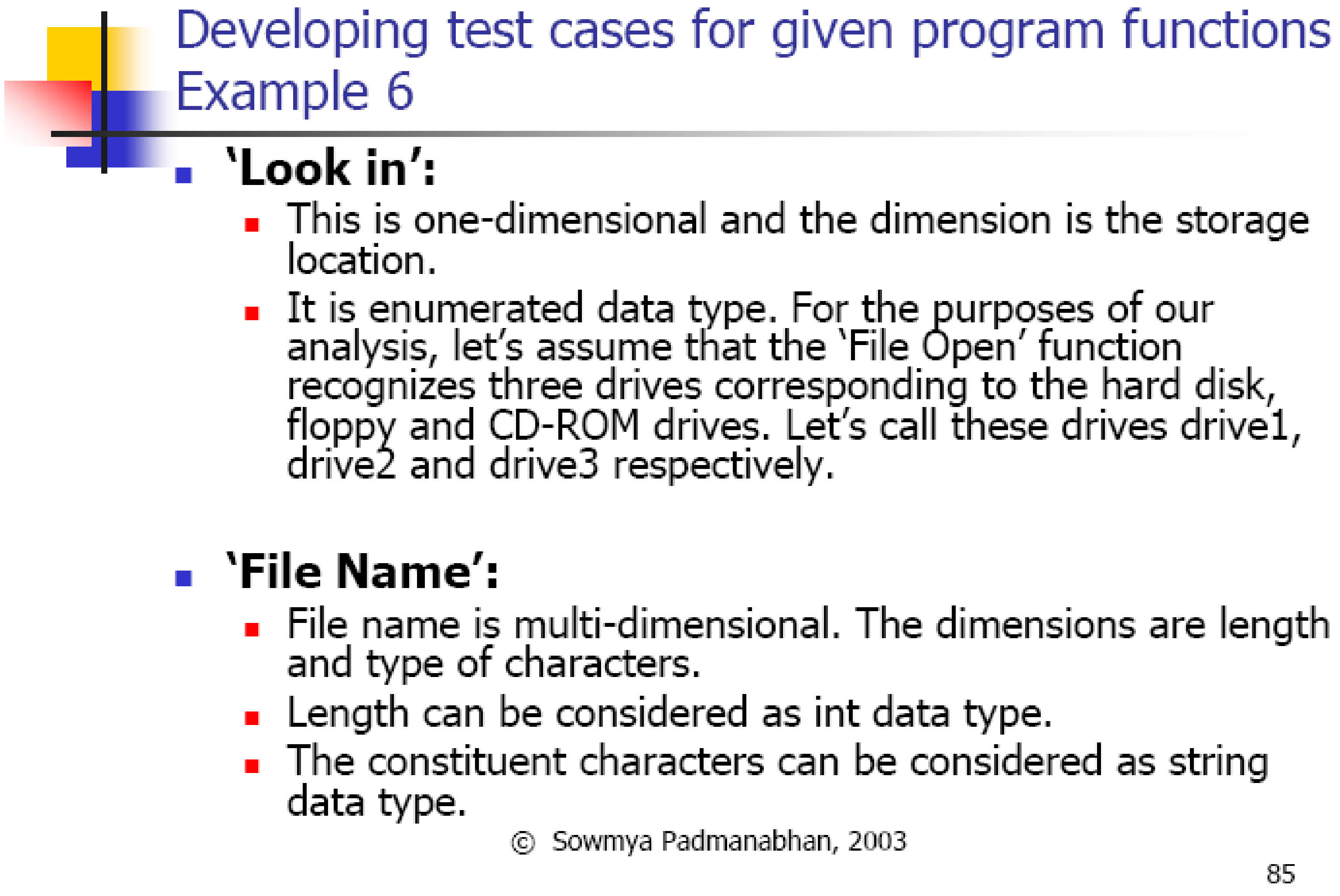


Developing test cases for given program functions Example 6

- ***Is the variable multidimensional? What are the dimensions? Can you map the variable to standard data types like int, double, float, string, char, etc... along each of its dimensions?***

© Sowmya Padmanabhan, 2003

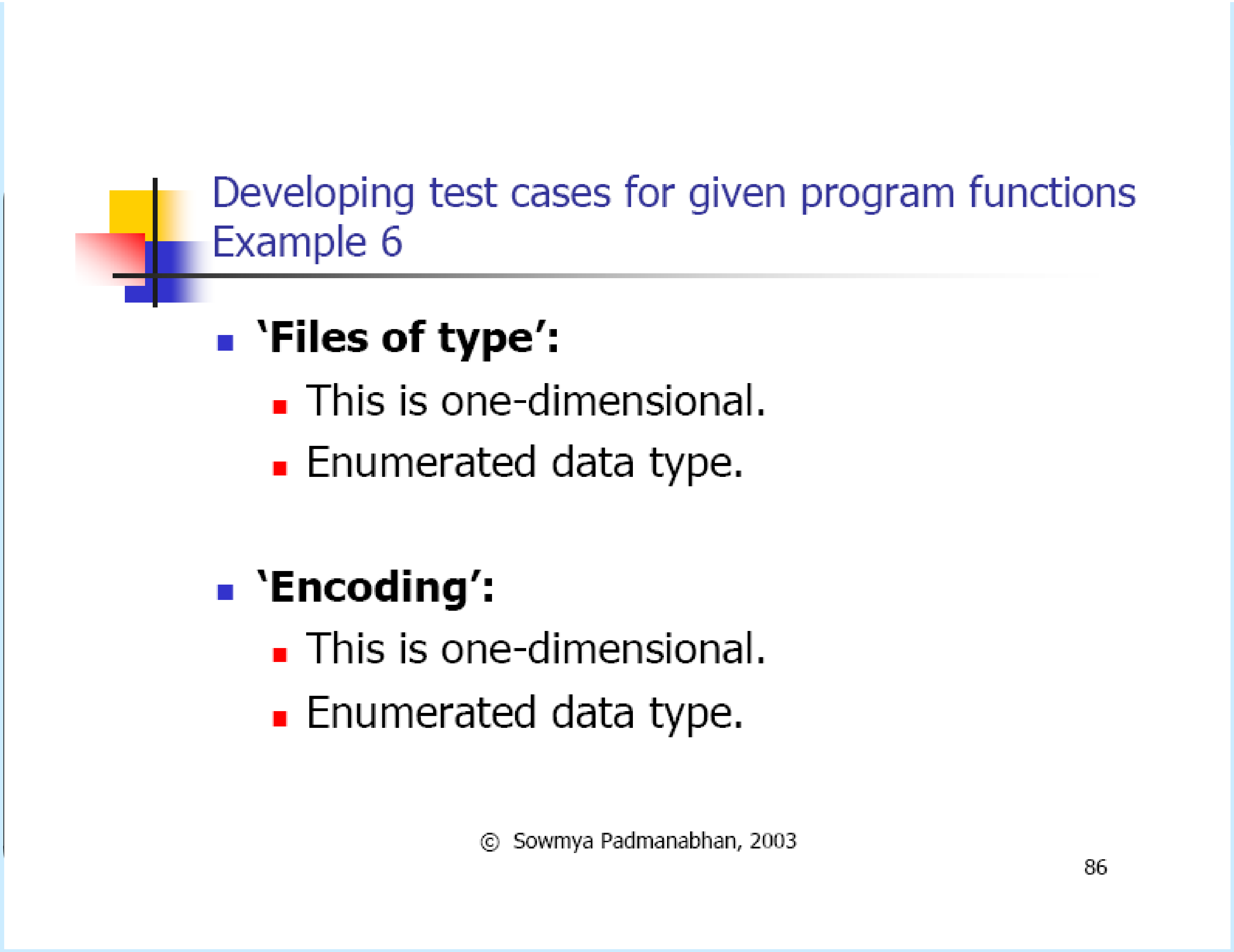
84



Developing test cases for given program functions Example 6

- **'Look in':**
  - This is one-dimensional and the dimension is the storage location.
  - It is enumerated data type. For the purposes of our analysis, let's assume that the 'File Open' function recognizes three drives corresponding to the hard disk, floppy and CD-ROM drives. Let's call these drives drive1, drive2 and drive3 respectively.

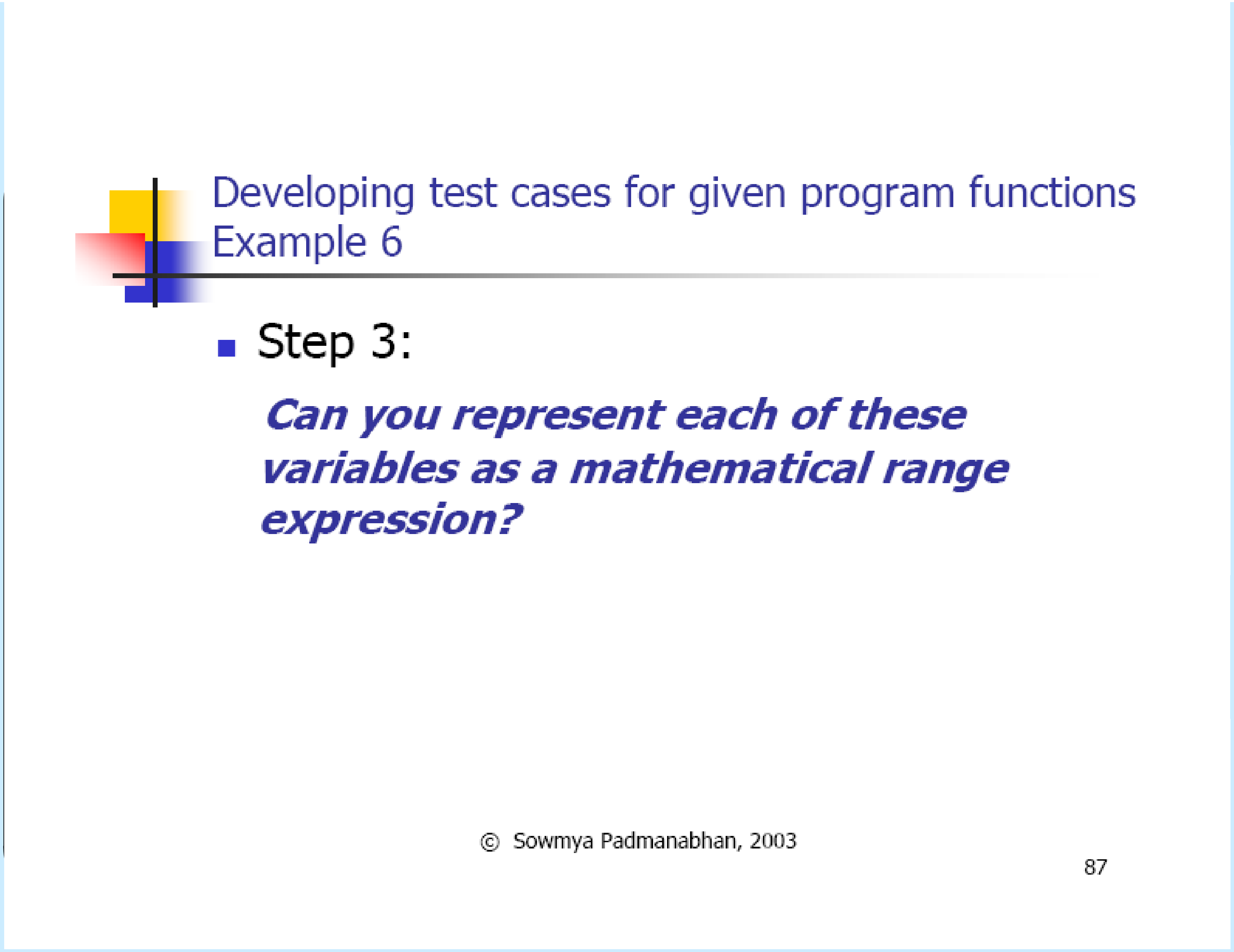
- **'File Name':**
  - File name is multi-dimensional. The dimensions are length and type of characters.
  - Length can be considered as int data type.
  - The constituent characters can be considered as string data type.



Developing test cases for given program functions Example 6

- **'Files of type':**
  - This is one-dimensional.
  - Enumerated data type.
- **'Encoding':**
  - This is one-dimensional.
  - Enumerated data type.

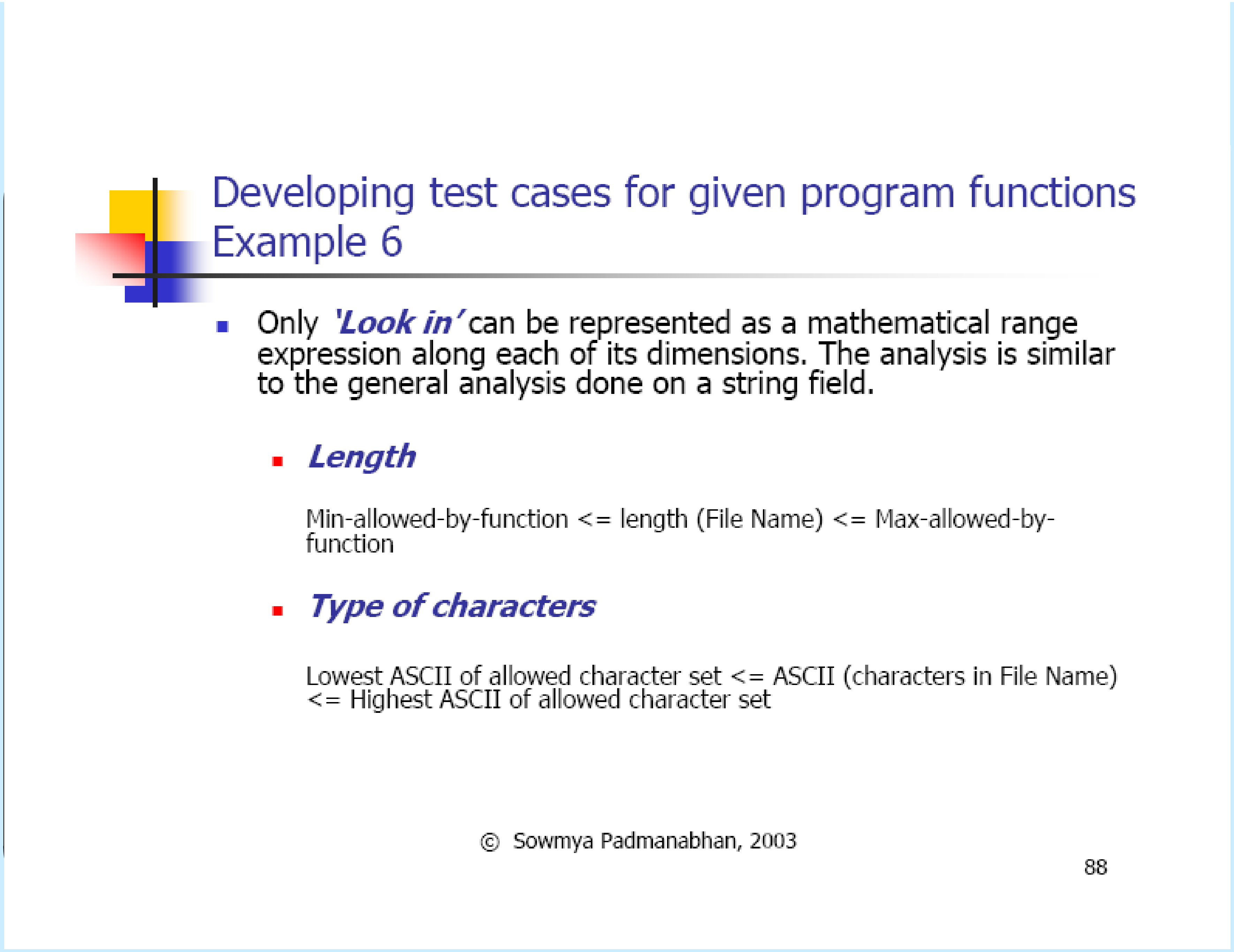
86



Developing test cases for given program functions Example 6

- Step 3:

  Can you represent each of these variables as a mathematical range expression?

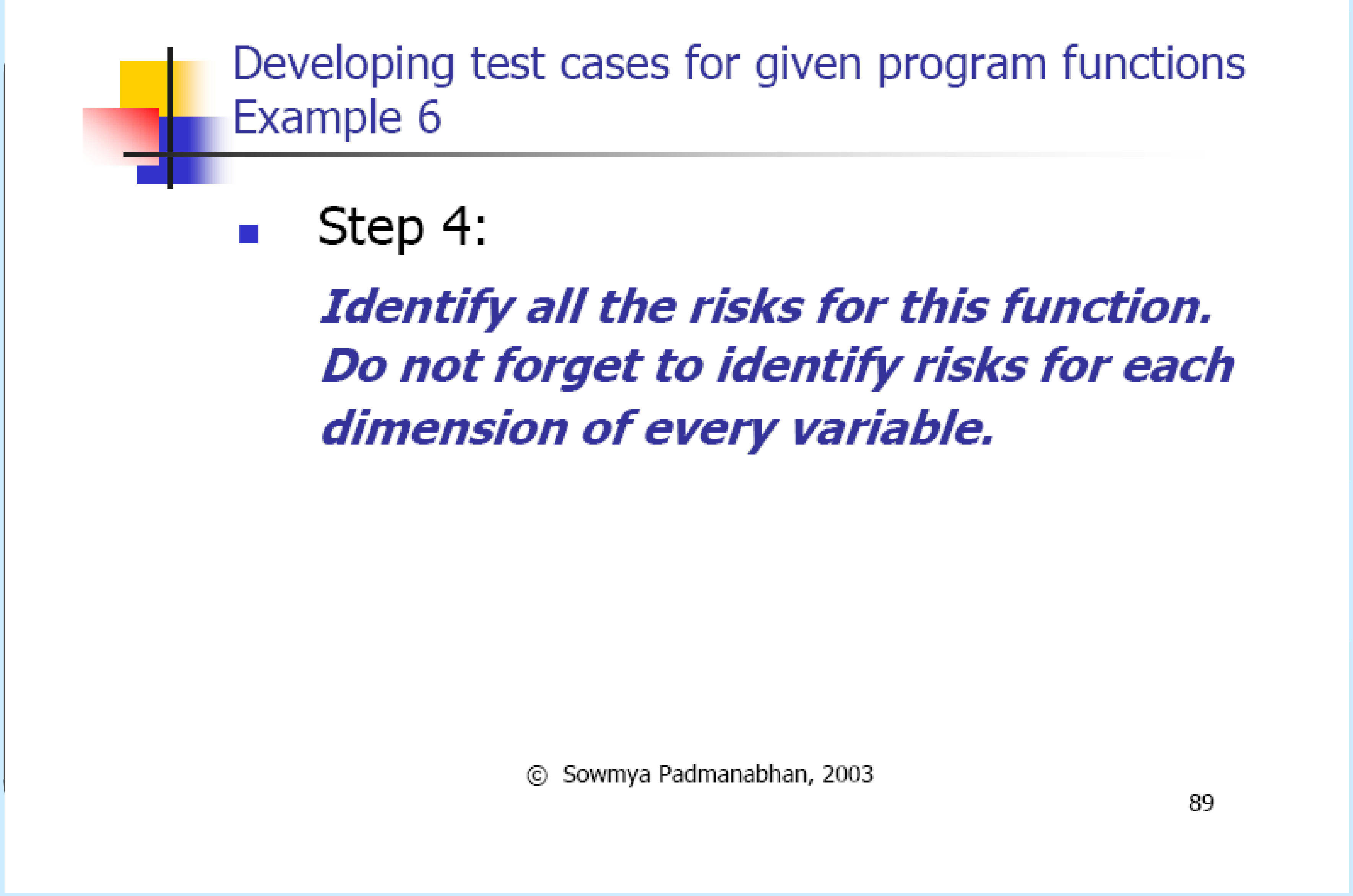


Developing test cases for given program functions Example 6

- Only ***'Look in'*** can be represented as a mathematical range expression along each of its dimensions. The analysis is similar to the general analysis done on a string field.
  - ***Length***

    Min-allowed-by-function <= length (File Name) <= Max-allowed-by-function
  - ***Type of characters***

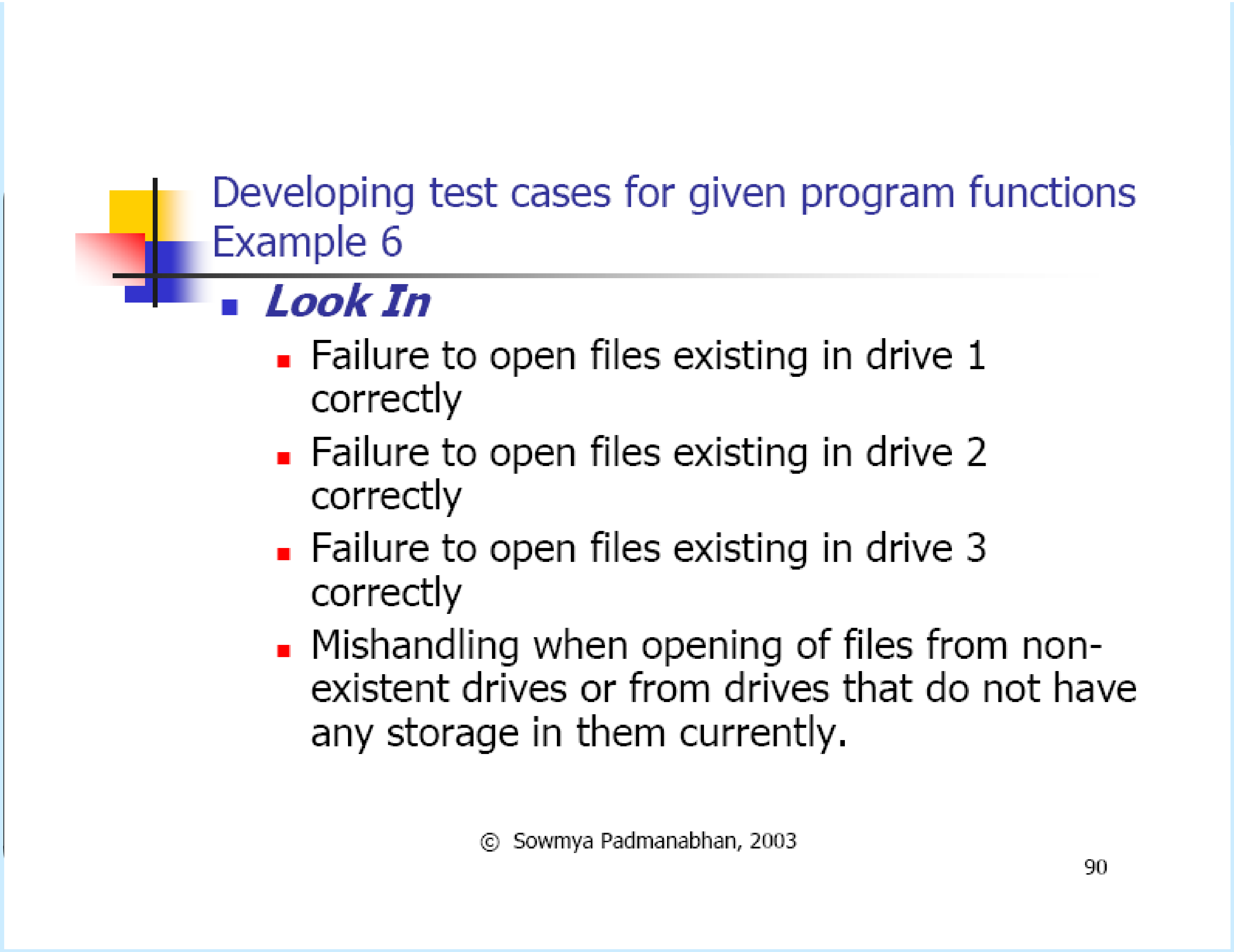
    Lowest ASCII of allowed character set <= ASCII (characters in File Name) <= Highest ASCII of allowed character set



Developing test cases for given program functions Example 6

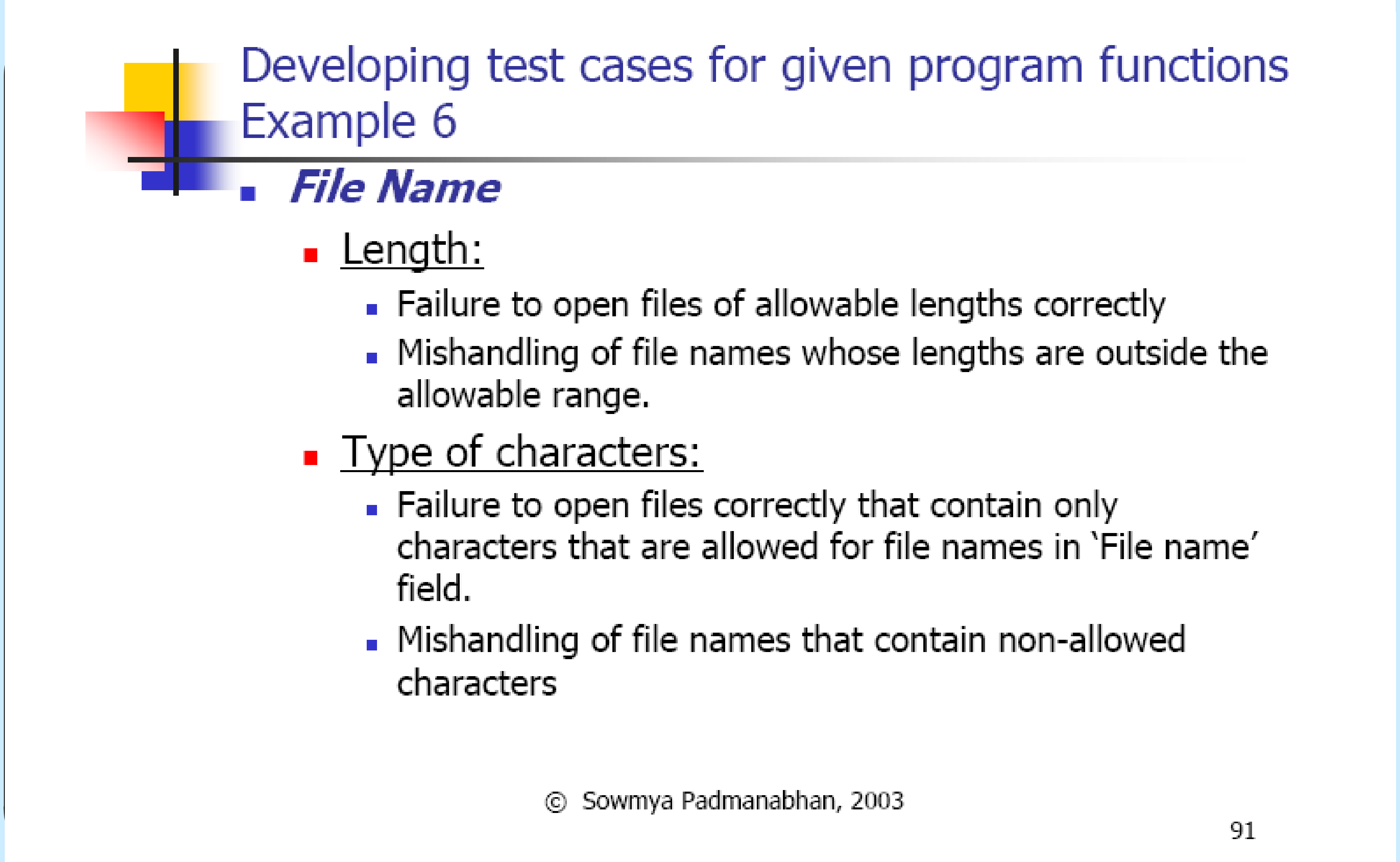
- Step 4:

  Identify all the risks for this function. Do not forget to identify risks for each dimension of every variable.



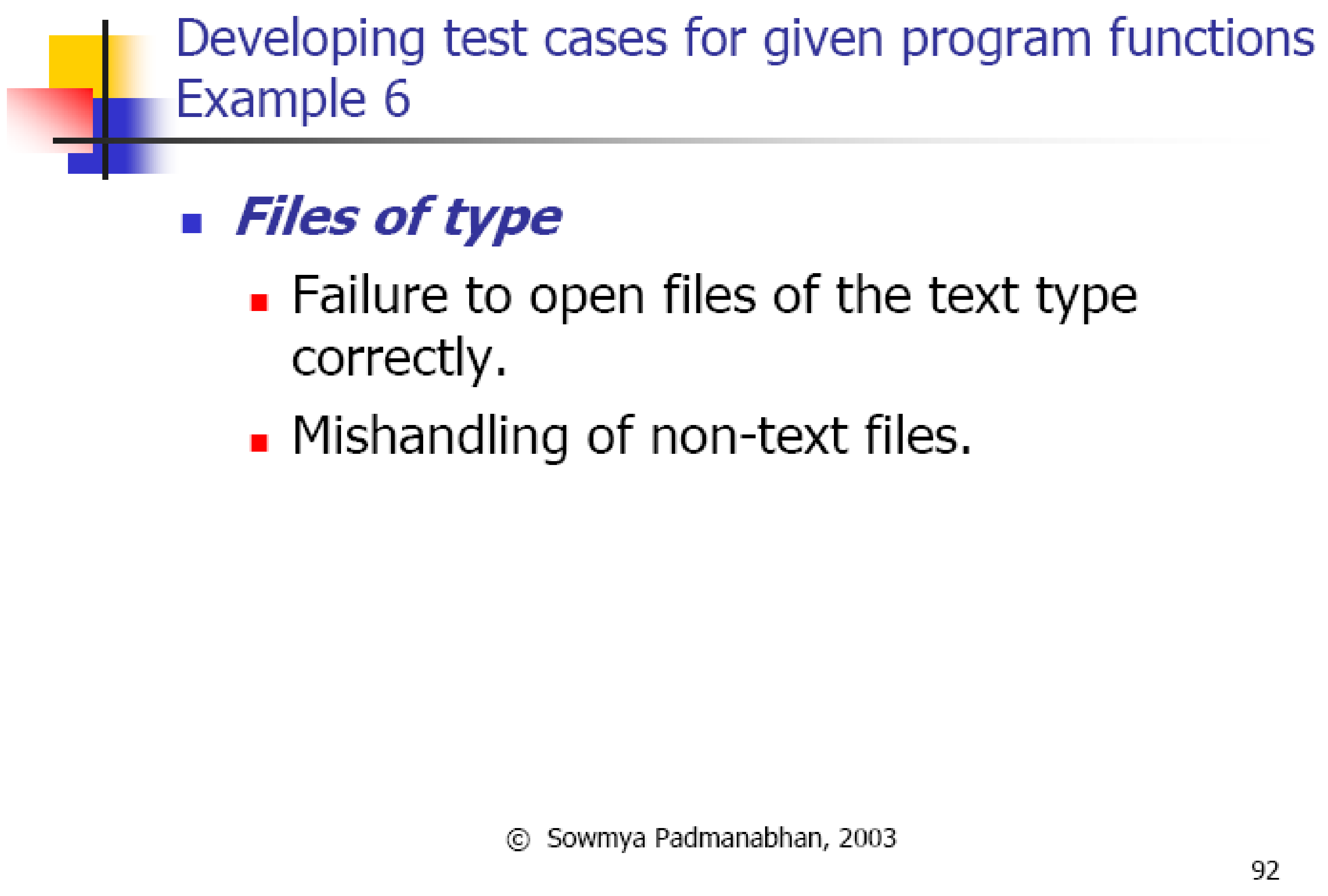
Developing test cases for given program functions Example 6

- ***Look In***
  - Failure to open files existing in drive 1 correctly
  - Failure to open files existing in drive 2 correctly
  - Failure to open files existing in drive 3 correctly
  - Mishandling when opening of files from non-existent drives or from drives that do not have any storage in them currently.



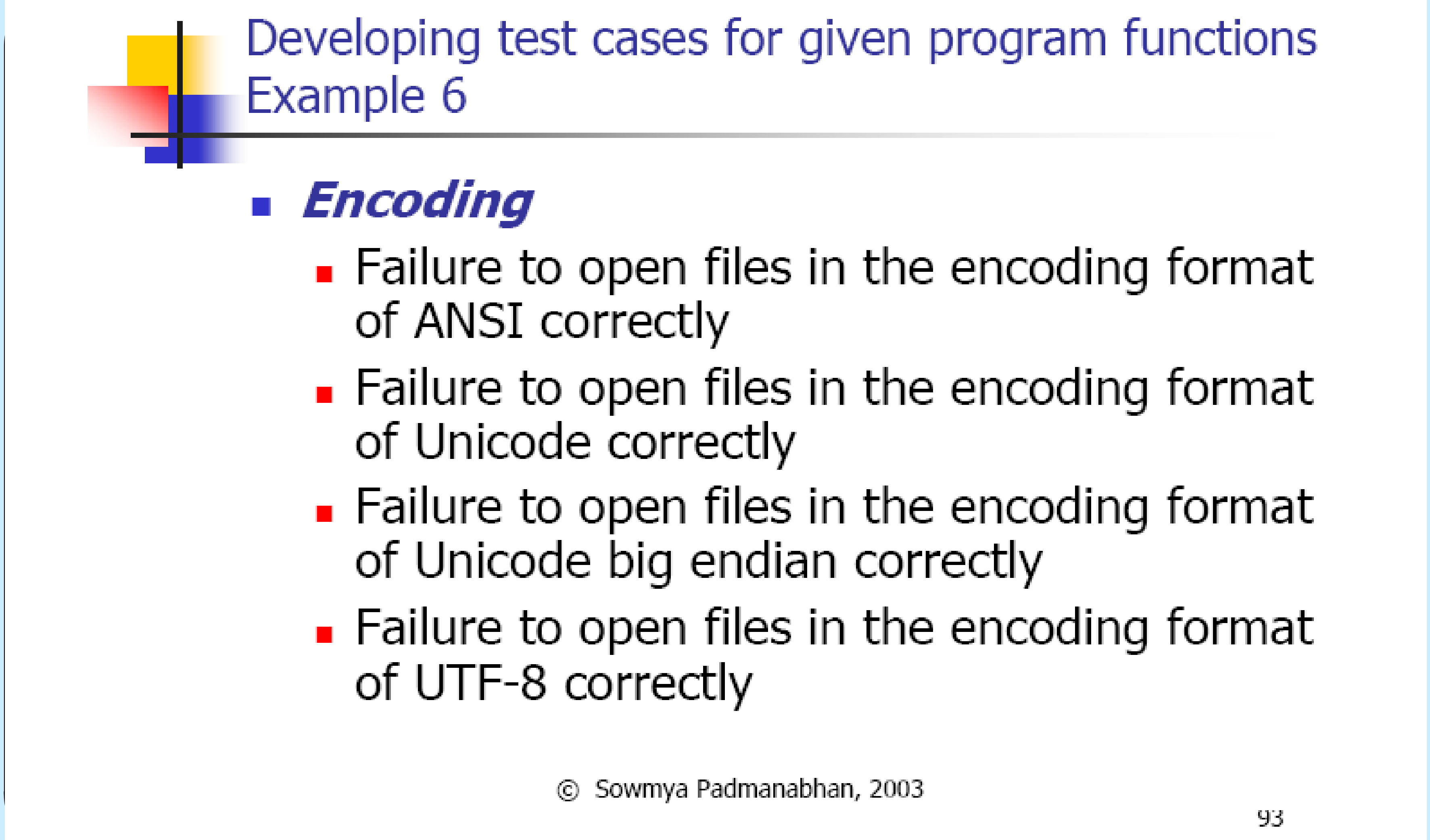
Developing test cases for given program functions Example 6

- ***File Name***
  - <u>Length:</u>
    - Failure to open files of allowable lengths correctly
    - Mishandling of file names whose lengths are outside the allowable range.
  - <u>Type of characters:</u>
    - Failure to open files correctly that contain only characters that are allowed for file names in `File name' field.
    - Mishandling of file names that contain non-allowed characters



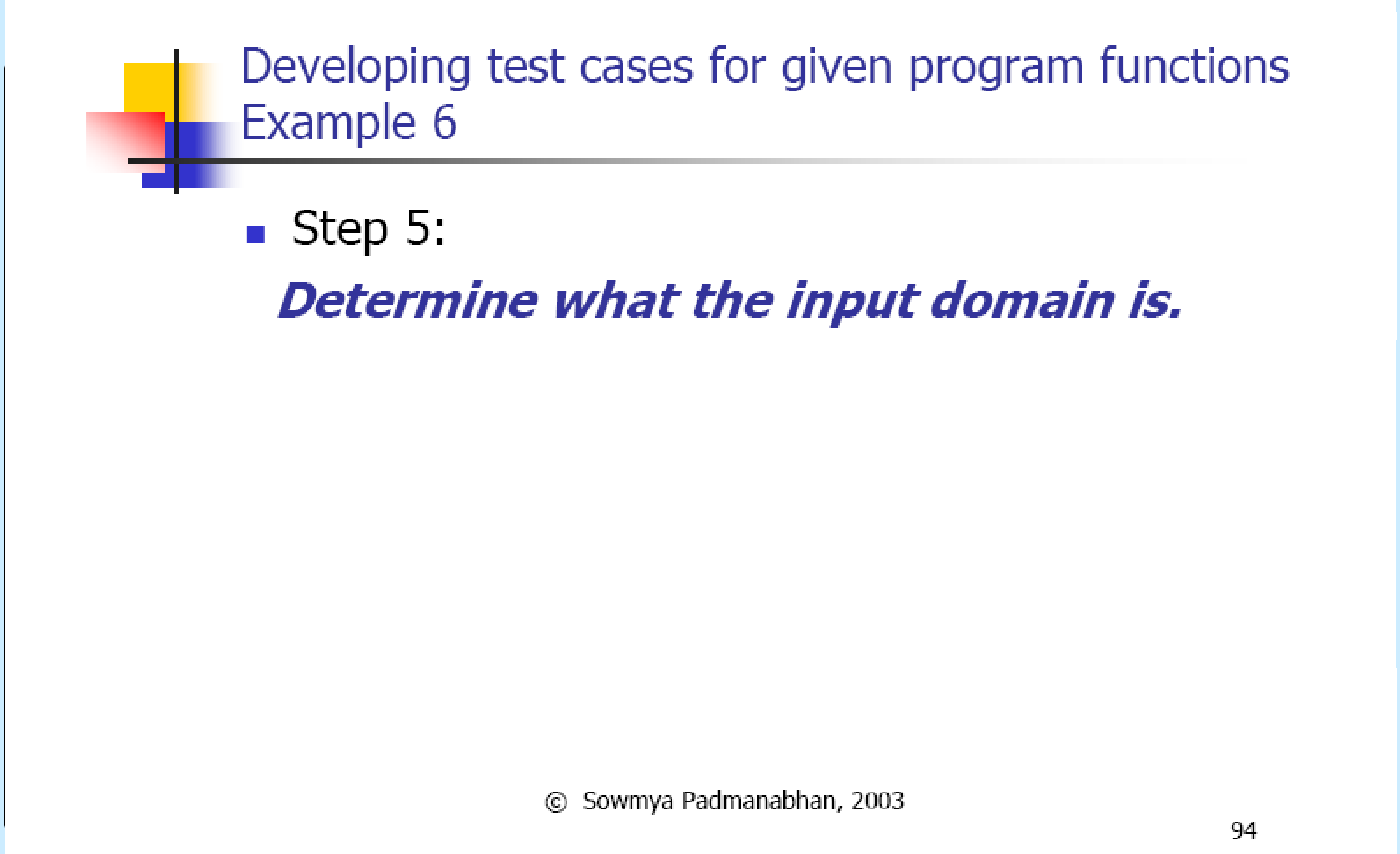
Developing test cases for given program functions Example 6

- ***Files of type***
  - Failure to open files of the text type correctly.
  - Mishandling of non-text files.



Developing test cases for given program functions Example 6

- ***Encoding***
  - Failure to open files in the encoding format of ANSI correctly
  - Failure to open files in the encoding format of Unicode correctly
  - Failure to open files in the encoding format of Unicode big endian correctly
  - Failure to open files in the encoding format of UTF-8 correctly



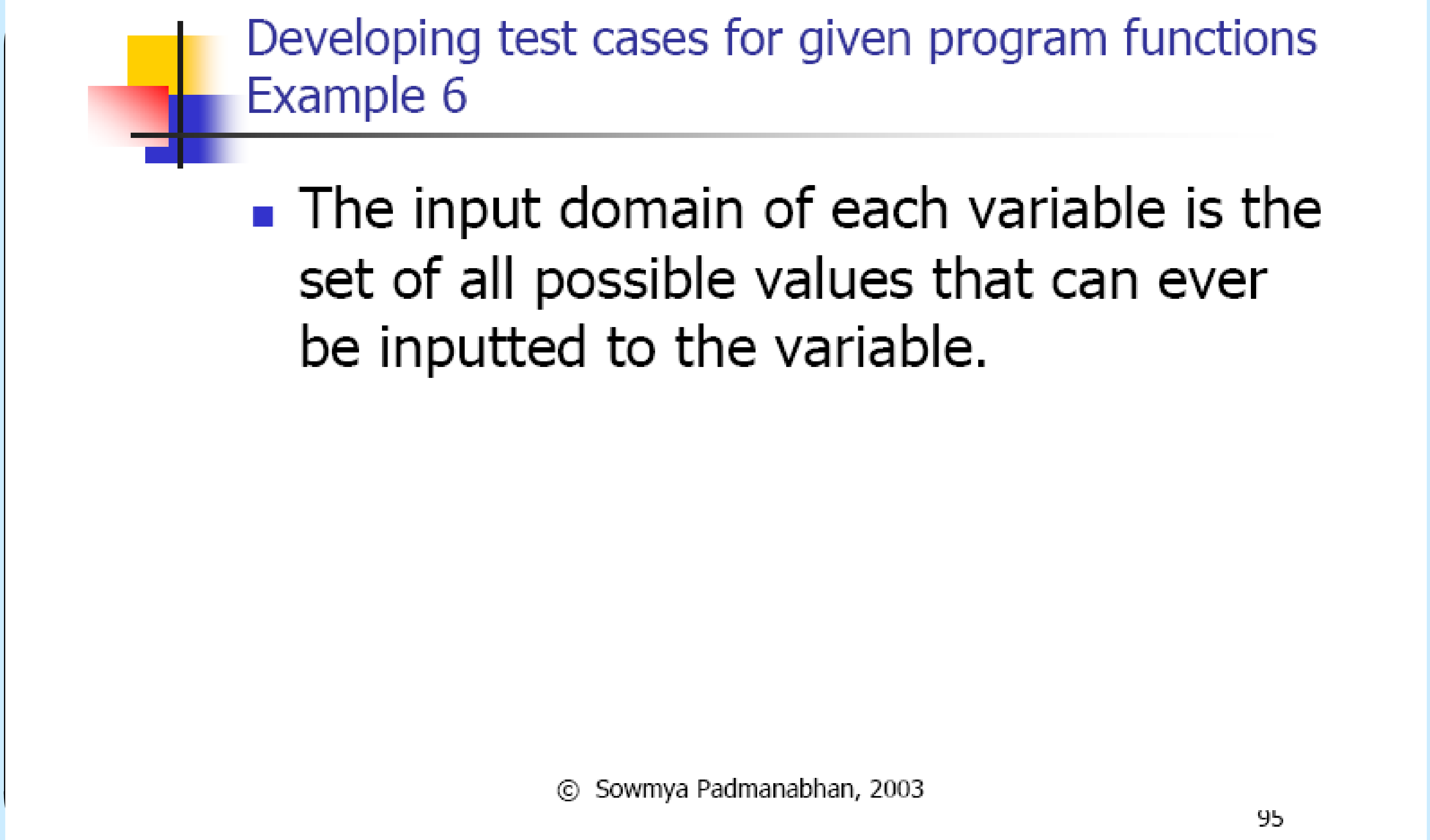
Developing test cases for given program functions
Example 6

- Step 5:

Determine what the input domain is.

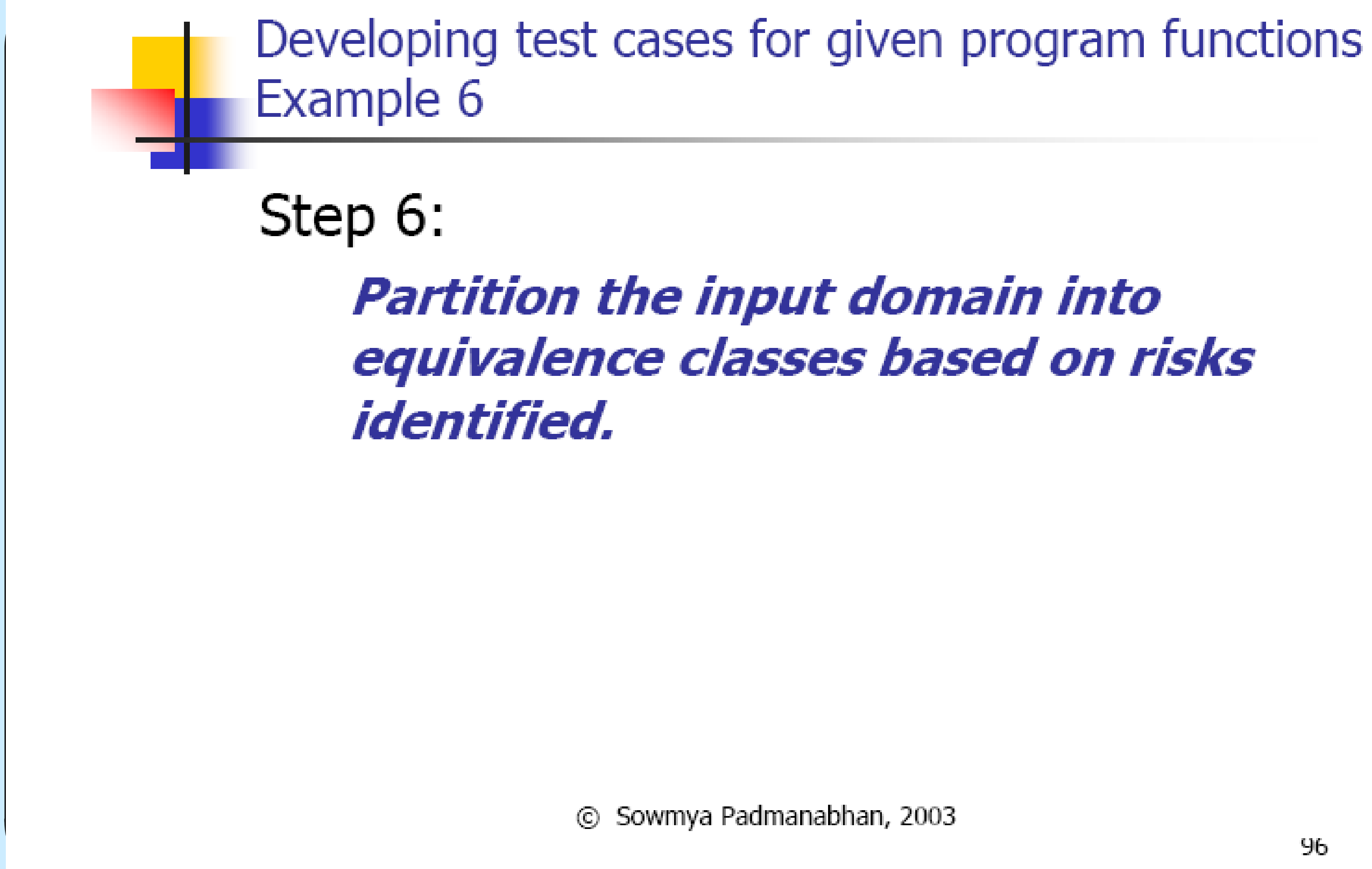
© Sowmya Padmanabhan, 2003

94



Developing test cases for given program functions Example 6

- The input domain of each variable is the set of all possible values that can ever be inputted to the variable.



Developing test cases for given program functions Example 6

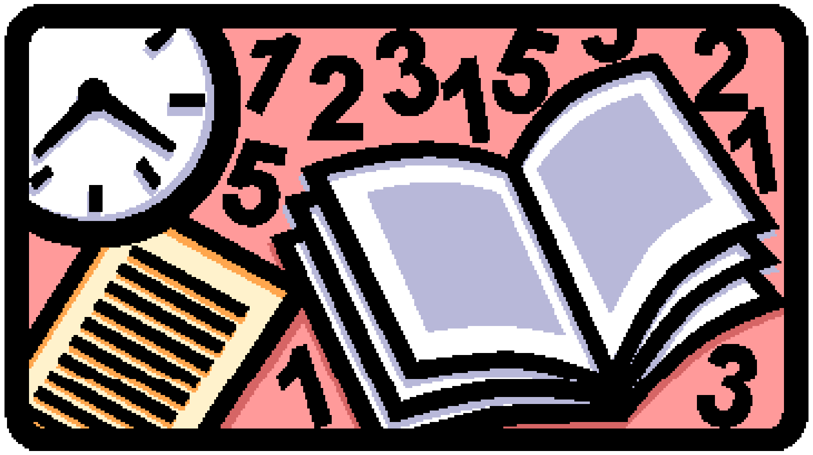
Step 6:

Partition the input domain into equivalence classes based on risks identified.

© Sowmya Padmanabhan, 2003

96

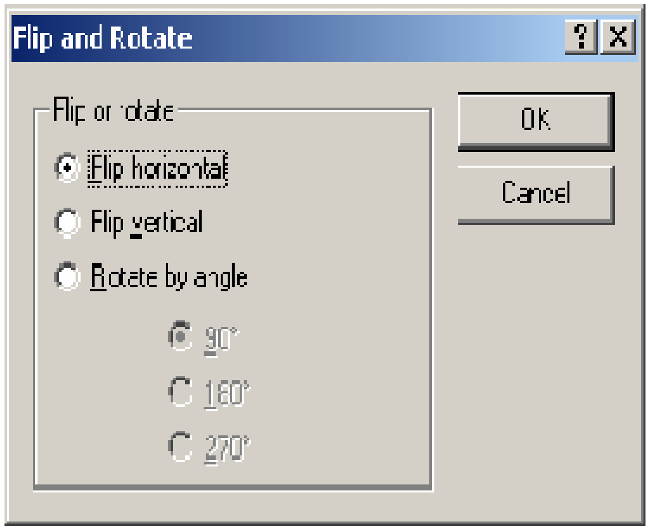
It's exercises time!



Developing test cases for given program functions

- Example 7:

For the 'Flip and Rotate' function of the Paint program, identify its variables, the dimensions, data type along each dimension. Develop a series of test cases by performing equivalence class analysis and boundary-value analysis on each of the variables. A screenshot of the function is provided on the right hand side.


Flip and Rotate

Flip or rotate
- [✓ Flip horizontal]
- Flip vertical
- Rotate by angle
  - [✓ 90°]
  - 180°
  - 270°

Results: Formative assessment

- Students performed well on tasks and quizzes during the instructional sequence.
  - Some students required more coaching than others.
- Course evaluations (each day) were very favorable to the instructor and the materials

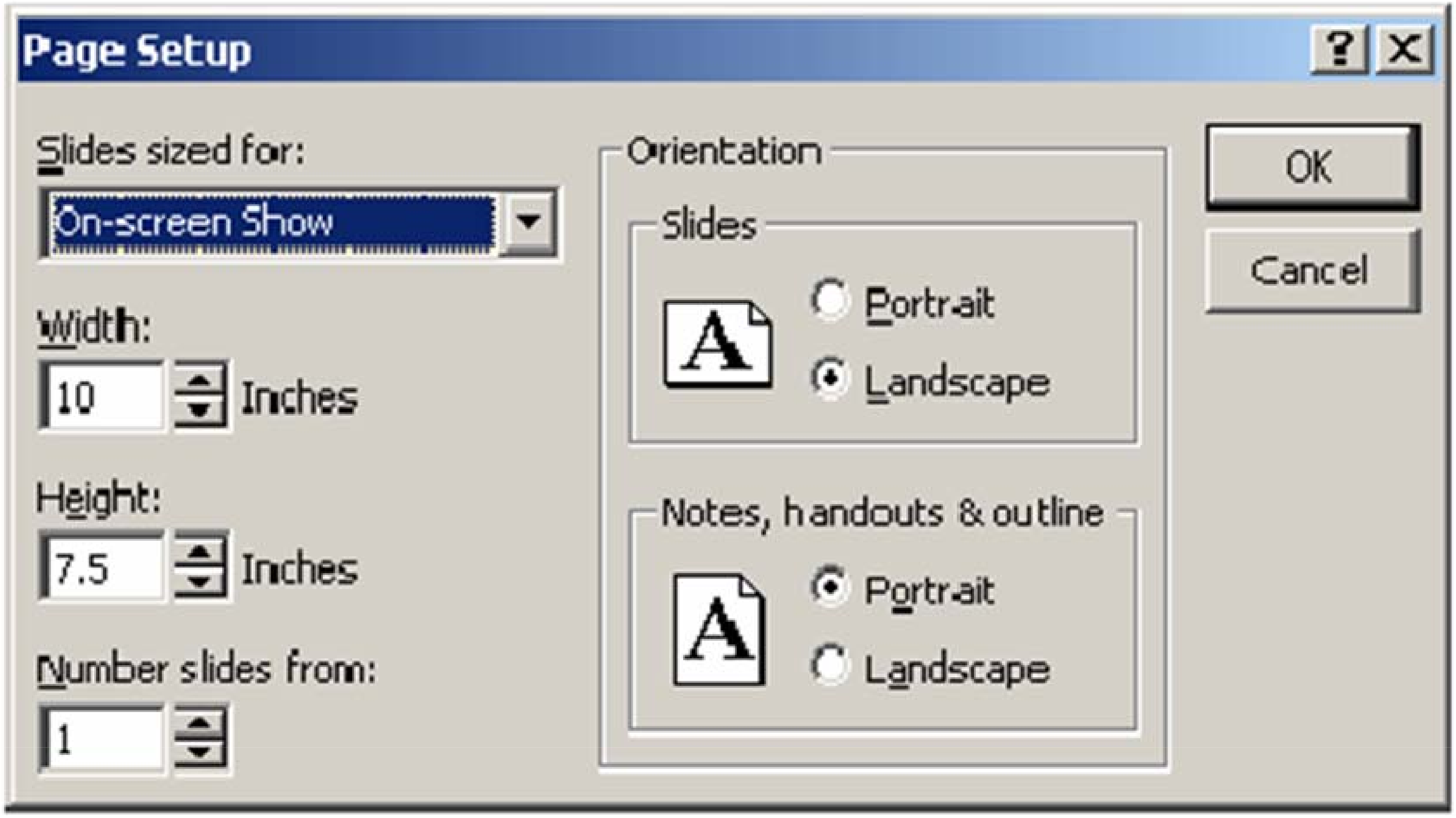
Results: Final exam (Pre-test / Post-test)

All learners improved from pre-test to post-test

- Pre-test mean grade 34.6%
- Post-test mean grade 91.4%

Final exam: Performance test

For the *Page Setup* function of Microsoft PowerPoint application, whose screenshot is provided below, develop a series of tests for this function by performing equivalence class and boundary value analysis on it.



Slides sized for: On-screen Show

Width: 10 Inches

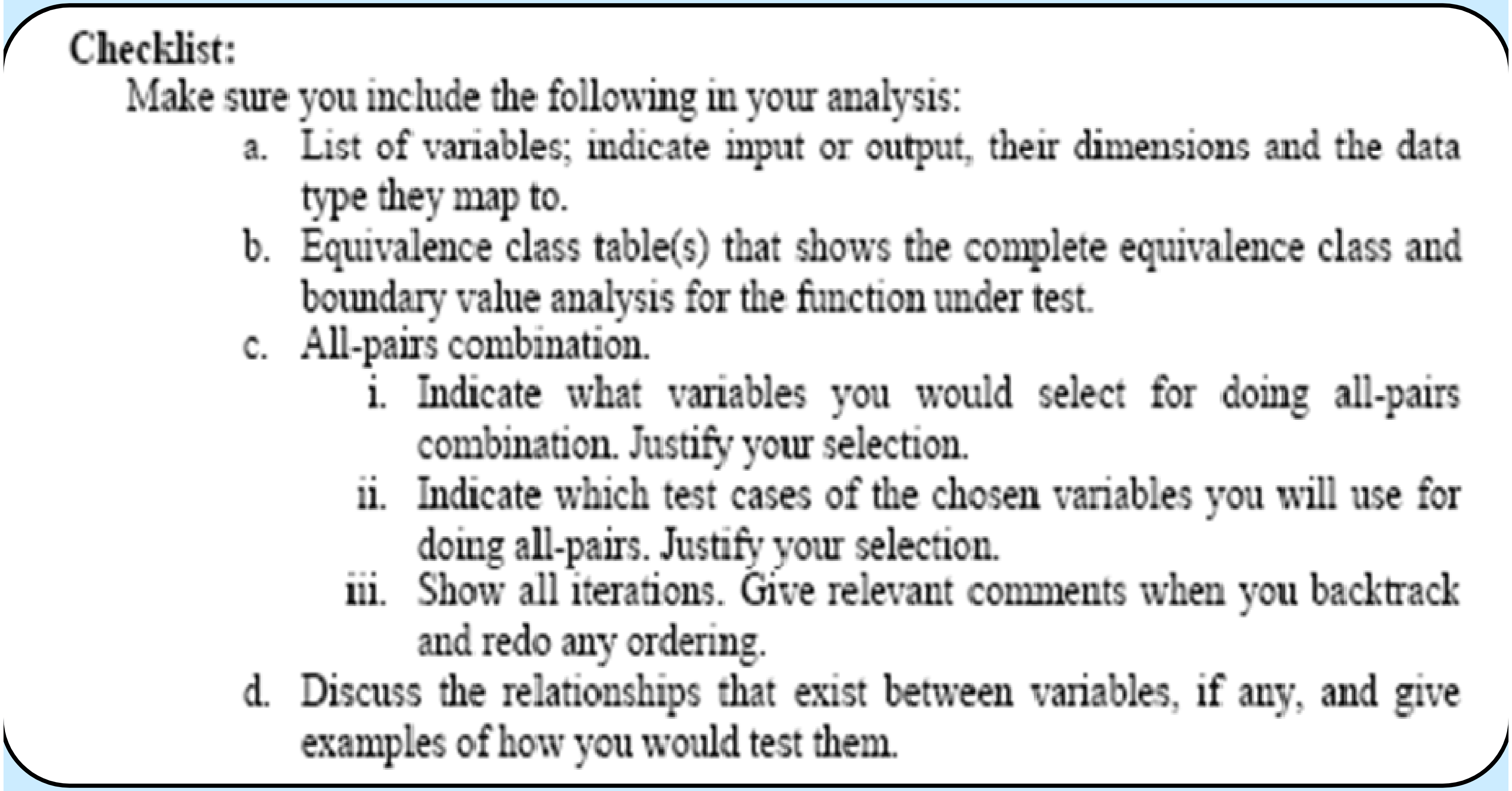
Height: 7.5 Inches

Number slides from: 1

Slides: ( ) Portrait [✓ Landscape]

Notes, handouts & outline: [✓ Portrait] ( ) Landscape

Final exam: Performance test



Checklist:

Make sure you include the following in your analysis:

a. List of variables; indicate input or output, their dimensions and the data type they map to.

b. Equivalence class table(s) that shows the complete equivalence class and boundary value analysis for the function under test.

c. All-pairs combination.

   i. Indicate what variables you would select for doing all-pairs combination. Justify your selection.

   ii. Indicate which test cases of the chosen variables you will use for doing all-pairs. Justify your selection.

   iii. Show all iterations. Give relevant comments when you backtrack and redo any ordering.

d. Discuss the relationships that exist between variables, if any, and give examples of how you would test them.

Evaluation of performance test

3 graders (Bach, Kaner, McGee)

- each with over 15 years in the field and supervisory experience
- “Compare the results from this Performance Test to the results that you would expect from a tester who claimed to have a year’s experience and who claimed to be good at domain testing.”

Performance exam results

All learners:

- approached the task in the same order,
- identified the same variables,
- analyzed them in essentially the same way,
- presented the results in extremely similar tables
- missed the same bugs.

- Result of:
  - guiding their work from the detailed procedures / examples
  - not from collaboration (cheating) during the exam

Performance exam example (see paper for more…)

This version of PowerPoint gave an error message for large pages:

"The current page size exceeds the printable area of the paper in the printer. Click Fix to automatically fit the page to the paper. Click Cancel to return to the Page Setup dialog box. Click OK to continue with the current page size."

- Here is new set of equivalence classes—page sizes that will fit the printer versus sizes that will not.
- No learner based a test on this, even though every one should have seen this message while checking page size limits.

(In post-experiment interviews months later, some students remembered seeing this but could not explain why they didn't follow it up. It just didn't occur to them to follow this up.)

Performance exam results

In the tested version of PowerPoint, the slide was rescaled.

- Resize slide from 8.5” wide / 11” tall to 56” wide /1” tall, text on that slide becomes short and wide.
- Some of the distortions look very bad.
- Could trigger intermittent errors (including a crash) by repeatedly resizing pages that had text and graphics.

No learner checked what the slides actually looked like after dimensions had changed. All tests stopped at the dialog.

The checklists repeatedly mentioned considering output variables (such as the actual displayed resized slide), but never drilled students on this.

No learner generalized in this way from lecture to application.

What next?

We reviewed the course notes and found various potential tweaks—ways to

- generalize the instruction
- introduce a wider variety of risks
- practice students in checking the impact of a change to a variable

These would have produced better performance on this particular performance exam, but what about the next variation of performance exam?

We didn't see how these would lead to a better general transfer

What next?

Over the next semesters, Kaner met some of the students who had taken the course. (They participated anonymously but often voluntarily self-identified.)

- Their approach was stereotyped and rigid
- They generally looked for procedures for other techniques
- They seemed disadvantaged compared to other students, when it came to learning new techniques.

Performance exam results

Conclusion:

- The students learned the procedures well and followed them closely
- The students did not transfer what they learned from the procedures and heard in the lectures:
  - to consider possibilities / risks beyond those covered in class, or
  - to a slightly more complex example.
- The students did not learn domain testing to the level we would have expected from modestly-experienced practitioners.

In retrospect...

To many educational theorists, this is hardly news:

- Lecture is good for transmitting basic information but not for fostering skilled application, evaluation or extension
- Procedural practice is good for getting across basic procedures, but as with mathematics instruction, stellar performance on examinations doesn't mean that students can transfer the knowledge beyond the classroom.

My response

http://www.testingeducation.org/BBST

Rather than tweaking the procedural approach, Kaner decided to try a new direction.

- Lectures on video
- Class-time on meaningful activities
- This is work in progress
- Some results are in Becky Fiedler's and my latest NSF proposal, http://www.kaner.com/pdfs/CirculatingCCLI2007.pdf

Implications?

- Back to1st year Java:
  - – Emphasis on larger, meaningful assignments motivates some students, but dismays many others. Serious dropout rate.
  - – Emphasis on narrowly-focused examples (many smaller exercises) seems easier, seems to foster faster connection with basics and more self-confidence, but maybe weak transfer.

Implications?

- Perhaps it is possible to
  - structure the course completely around constructivist experiences / complex enough tasks that they have to figure things out
  - without a high waste (dropout + failure) rate
- But I don't know how yet.
- What I'm now seeking is
  - a balanced approach (procedural foundation plus many challenging applications)
  - that fits within a semester, without creating an impossible workload.
- Thoughts?