Outline

1. Modular Design
   - multiple choice quiz
   - bottom-up design
2. The Software Cycle
   - quality of product and process
   - waterfall and spiral model

MCS 260 Lecture 24
Introduction to Computer Science
Jan Verschelde, 10 July 2023

modular design
software cycle & quality

Multiple Choice Quiz

problem specification

To test our knowledge of computer science concepts, we would like an automatic multiple choice quiz.

Our friend the computer will

1. show a question with multiple possible answers
2. wait for us to make a choice
3. immediately evaluate our choice
4. ask us if we want to continue
5. at the end print how well we did

Wanted: a small but extendable program.

a screen shot: a first prototype

The main program quiz.py, executed at the prompt `$`:

```
$ python quiz.py
The octal system has base ...
1. 16
2. 8
3. 10
4. 2
type a number between 1 and 4 : 2
this is the correct answer
continue ? (y/n) n
#correct : 1 #wrong : 0 -> 100.00%
```

Tools for Use

what does Python provide?

Python provides

1. a command line or Python shell environment
2. data structures: lists, tuples, and dictionaries
3. if-elif-else, loops, and functions
4. random number generators in the module `random`
5. our modules will reflect our bottom-up design

What is there at the bottom of our program?

modular design
software cycle & quality

1. Modular Design
   - multiple choice quiz
   - bottom-up design
2. The Software Cycle
   - quality of product and process
   - waterfall and spiral model

The Bottom of our Program

A multiple choice quiz consists of

1. questions that have short answers;
2. answers, correct ones and variations.

Every question and answer will be a string.

Dictionaries seem very appropriate:

1. the questions are numbered, key = number, the values are strings
2. answers for the corresponding question have the same key as the question
3. answers are dictionaries of dictionaries
4. convention: correct answer comes first

The Other Modules

We separate the functionalities:

1. generating questions and evaluating answers;
2. showing questions and asking answers.

Two modules of functions: questions and dialogue.

Both modules import the Q&A dictionary
and export their functions to the main program.

The formulation of questions and answers is centralized
in the Q&A dictionary.

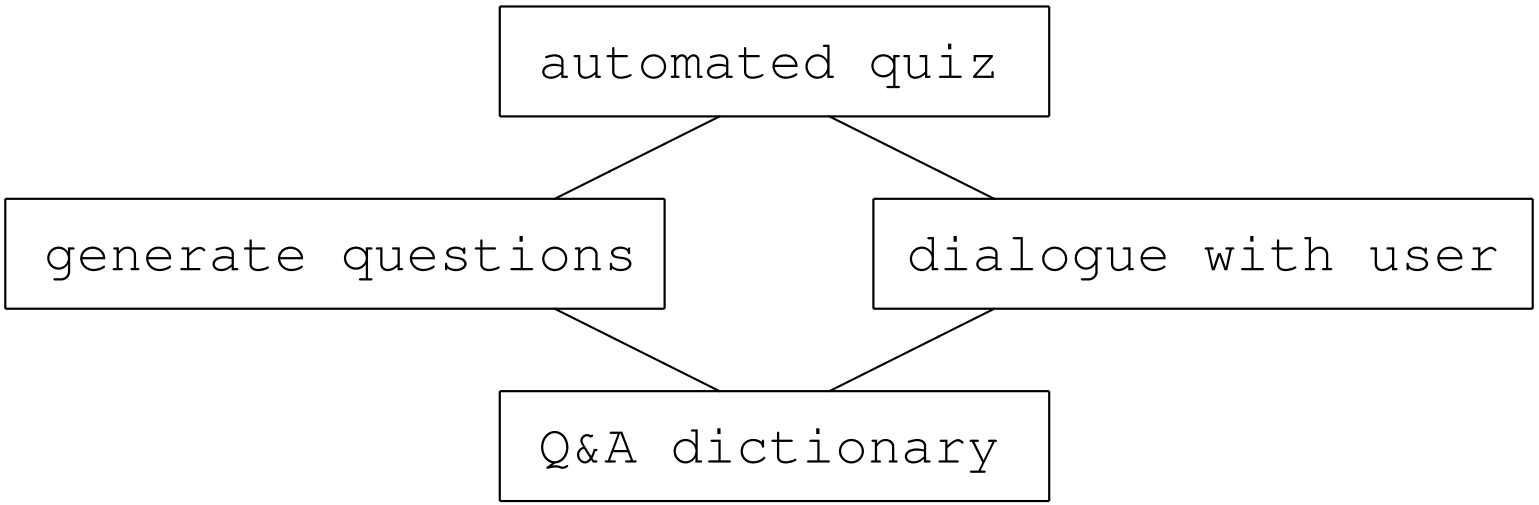
A multiple choice question then consists of

1. the number of the question;
2. a permutation of the answers.

In Python we use a tuple, e.g.: (3,[2,1,3,4]).

The Modular Design

At the bottom: the Q&A dictionary.
Two libraries of functions: questions and dialogue.



Information Hiding:

the main program does not import the Q&A dictionary.

The Software Cycle

We are experiencing the software crisis:
$\rightarrow$ *the industrial sector can not keep up with the ever growing demands of the market.*

Symptoms:

1. products are delivered late
2. the cost exceed budgets
3. too many bugs appear

The relative cost of hardware versus software

- was 80% versus 20% in the 1960s
- became 20% versus 80% in the 1980s
- keeps going in the same direction.

Reason: hardware production is technology intensive, whereas software *creation* is human intensive.

modular design
software cycle & quality

1. Modular Design
   - multiple choice quiz
   - bottom-up design
2. The Software Cycle
   - quality of product and process
   - waterfall and spiral model

Quality of Software Design

industrial quality

Industrial quality refers to products

1. that must meet a target value on average,
2. and within a tolerable standard deviation.

example: a search engine should give relevant answers within an average of 10 seconds, and standard deviation of 2 seconds.

We distinguish between quality of product and process:

1. a high quality product meets user's expectations,
2. also the quality of the production process matters!

External qualities can be observed directly by the user.
example: easy to use

Internal qualities can be measured only by the producer.
example: design documentation

Software Quality Features

what is good software?

A short list of important criteria:

correctness must satisfy requirements
How reliable is the software?

efficiency economical use of resources such as hardware and time
Do we understand how the complexity of the problem and the software relate?

modifiability to different environments and requirements
How hard is it to correct errors or to add new features?

reusability of the same software in different applications
Mathematical libraries are a typical example of reusable software.

modular design
software cycle & quality

1. Modular Design
   - multiple choice quiz
   - bottom-up design
2. The Software Cycle
   - quality of product and process
   - waterfall and spiral model

The Waterfall Model

seven phases in sequence

The software life cycle model describes the software production in seven phases:

1. feasibility study: cost-benefit analysis
2. system requirements specification: what must it do?
3. system architecture design: modules and relations
4. implementation and verification of modules
5. system integration and verification
6. installation: deliver software to the user
7. maintenance: fixing bugs and adding new features

As the phases occur in a strict sequence, just as water never flows upwards, this model is like a waterfall.

This deductive or top-down description of the life cycle has its merits, especially with feedback.

The Spiral Model

Four sectors in the software development:

1. requirements analysis and specification
2. system architecture design
3. implementation
4. verification

Incremental development of software:

1. think of a product with limited scope
2. create the design of the first product
3. implement a prototype
4. test and deliver the prototype

Based on feedback, enlarge the specifications, revise the designs, build new prototypes, test and deliver again.
As the cycles over the four sectors get larger,
the complexity of the software grows.

Exercises

1. Enlarge the Q&A answer dictionary with a least six more questions about relevant CS concepts.
2. Modify the program so the user has the option to see the correct answer when wrong.
3. Design your own trivia quiz.
4. Consider a quiz about the first 13 states of the union. The quiz links names of states with their captitals and year when they joined the union.
5. Develop a modular implementation of your own trivia quiz or of the quiz in the previous exercise.