

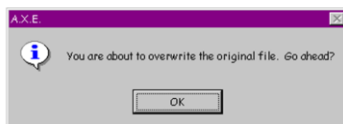
JT's note: in the interests of time this section will not be covered during the live lecture. Instead you can get the lecture content via a pre-recorded video in D2L under 'Lectures' under the appropriate week's material.

CPSC 217, Loops In Python: Part 3

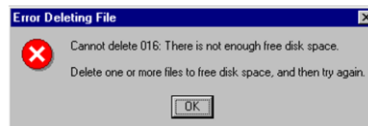
In this section you will learn some usability heuristics which can be used to design more user-friendly systems. (Coverage depends upon time constraints).

James Tam

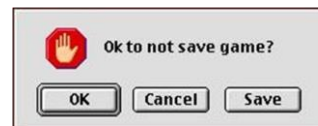
Why This Section Is Needed: Not So Friendly Examples Exist!



Do I have any choice in this? [AXE a hex editor]



Windows 95



Uhhh... I give up on this one [Mac shareware version of RISK]

James Tam

Some Heuristics (Rules Of Thumb) For Designing Software

- (The following list comes from Jakob Nielsen's 10 usability heuristics from the book "*Usability Engineering*")
 1. Minimize the user's memory load
 2. Be consistent
 3. Provide feedback
 4. Provide clearly marked exits
 5. Deal with errors in a helpful and positive manner

For more information:

- Jakob Nielsen: <https://www.nngroup.com/people/jakob-nielsen/>
- Book, "Usability Engineering (see Chapter 5)"
https://books.google.ca/books?id=DBOowF7LqIQC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

James Tam

1. Minimize The User's Memory Load

- Computers are good at 'remembering' large amounts of information.
- People are not so good remembering things.

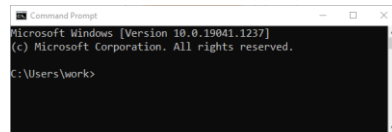
slide 4

James Tam

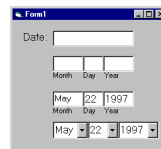
1. Minimize The User's Memory Load

- To reduce the memory load of the user:

- Poor approach: a command line interface (Windows 'cmd', MAC OS 'terminal')



- Example 1: applying the design principle with a graphical interface.

A screenshot of a graphical user interface (GUI) for entering a date. It has a title bar 'Form1'. Below the title bar, there's a 'Date:' label followed by three input fields: 'Month', 'Day', and 'Year'. The 'Month' field has a dropdown menu showing 'May'. The 'Day' field has a dropdown menu showing '22'. The 'Year' field has a dropdown menu showing '1997'.

- Describe required the input format, show examples of valid input, provide default inputs.

- Example 2: applying the design principle with a command line interface.

```
[csc loops 25 ]> python hci.py
Enter your birthday <month> <day> <year> e.g., 11 17 1977
Birthday: 
```

James Tam

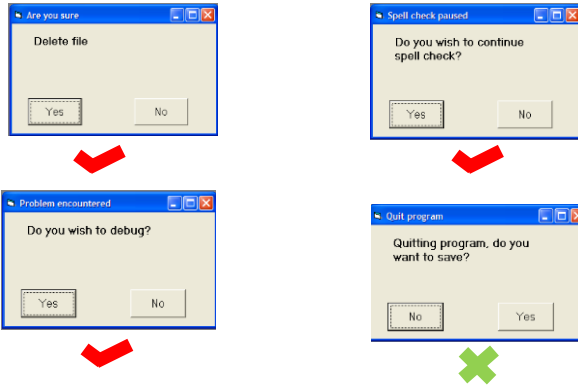
2. Be Consistent

- Consistency of effects (this command/action -> this result)
 - Same words, commands, actions will always have the same effect in equivalent situations
 - Makes the system more predictable
 - Reduces memory load
- Consistency of layout
 - Allows experienced users to predict where things should be (matches expectations).

James Tam

2. Be Consistent

- Consistency of language and graphics
 - Same information/controls in same location on all screens / dialog boxes forms follow boiler plate.
 - Same visual appearance across the system (e.g. widgets).



Images courtesy of
James Tam

James Tam

2. Be Consistent

```
FIRST CATEGORY: ELECTRICITY
-----
You can either enter your monthly kilowatt hours or have an estimate
based on the size of the accomodation that you live in.

(e)stimate
(k)ilowatt hours used
(q)uit this question and proceed to the next question
Enter selection: q

Tons of carbon generated from powering accomodation: 0
Current tons of carbon currently generated: 0

SECOND CATEGORY: HEATING
-----
What size of place do you live:
(s)mall house or a flat
(m)edium house
(l)arge house
(q)uit this question and proceed to the next question
Enter selection: 
```

This last
option
always
allows the
user to
proceed to
the next
question.

James Tam

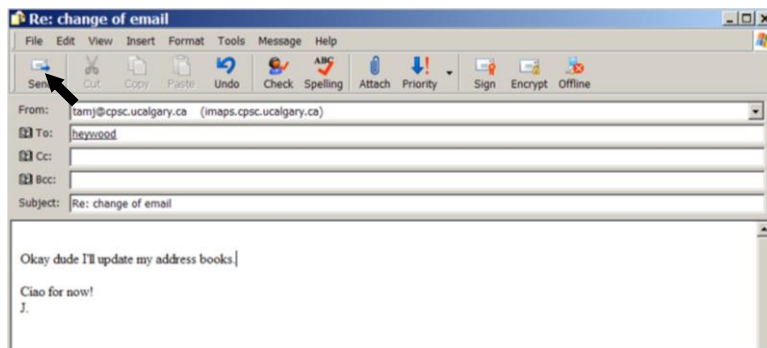
3. Provide Feedback

- Letting the user know:
 - what the program is currently doing: was the last command understood, has it finished with its current task,
 - what task is it currently working on,
 - how long will the current task take etc.

James Tam

3. Provide Feedback

- What is the program doing?

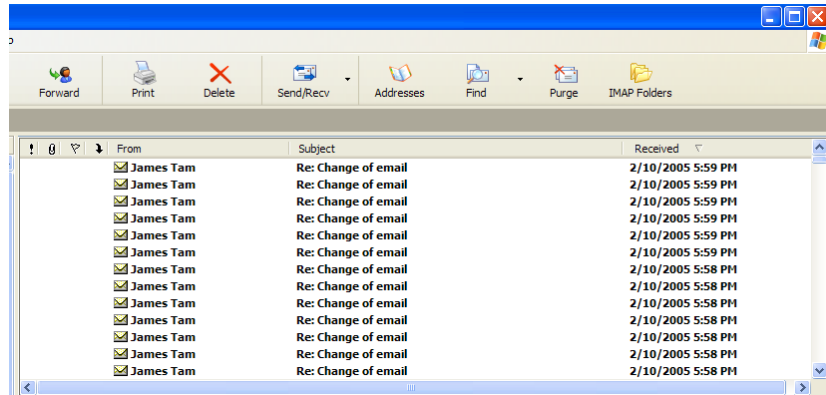


Outlook Express image courtesy of
James Tam

James Tam

3. Provide Feedback

- The rather unfortunate effect on the (poor) recipient.



Outlook Express image courtesy of
James Tam

James Tam

3. Provide Feedback

- In terms of this course, feedback is appropriate for instructions that may not successfully execute
 - what the program is doing (e.g., opening a file),
 - what errors may have occurred (e.g., could not open file),
 - and whenever possible 'why' (e.g., file "input.txt" could not be found)
- ...it's not hard to do and not only provides useful updates with the state of the program ("Is the program almost finished yet?") but also some clues as to how to avoid the error (e.g., make sure that the input file is in the specified directory).
- At this point your program should at least be able to provide some rudimentary feedback
 - E.g., if a negative value is entered for age then the program can remind the user what is a valid value (the valid value should be shown to the user as he or she enters the value):

```
age = int(input ("Enter age (0 - 114): "))
```

James Tam

4. Provide Clearly Marked Exits

- This should obviously mean that quitting the program should be easy/self-evident (although this is not always the case with all programs!).
- In a more subtle fashion it refers to providing the user the ability to reverse or take back past actions (e.g., the person was just experimenting with the program so it shouldn't be 'locked' into mode that is difficult to exit).
- Users should also be able to terminate lengthy operations as needed.

James Tam

4. Provide Clearly Marked Exits

- This doesn't just mean providing an exit from the program but the ability to 'exit' (take back) the current action.
 - Universal Undo/Redo
 - e.g., <Ctrl>-<Z> and <Ctrl>-<Y>
 - Progress indicator & Interrupt
 - Length operations

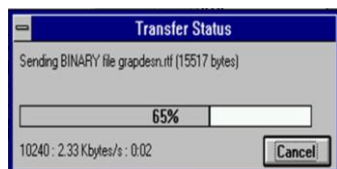
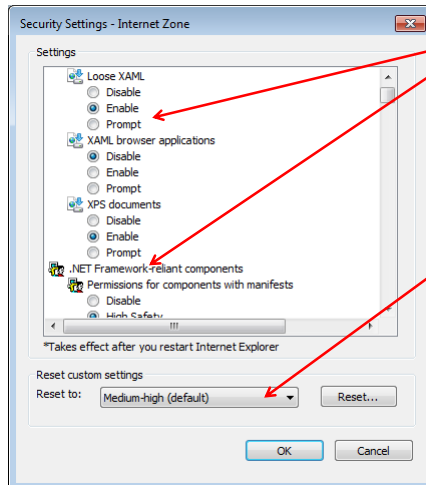


Image: From the "HCI Hall of Shame"

James Tam

4. Provide Clearly Marked Exits

- Restoring defaults
 - Getting back original settings



- What option did I change?
- What was the original setting?

- Allows for defaults to be quickly restored

Image: Internet Explorer security settings courtesy of James Tam

James Tam

4. Provide Clearly Marked Exits

```
FIRST CATEGORY: ELECTRICITY
-----
You can either enter your monthly kilowatt hours or have an estimate
based on the size of the accommodation that you live in.

(e)stimate
(k)ilowatt hours used
(q)uit this question and proceed to the next question
Enter selection: q

Tons of carbon generated from powering accommodation: 0
Current tons of carbon currently generated: 0

SECOND CATEGORY: HEATING
-----
What size of place do you live:
(s)mall house or a flat
(m)edium house
(l)arge house
(q)uit this question and proceed to the next question
Enter selection: 
```

- The user can skip or 'exit' any question.

Image: An old CPSC 231 assignment courtesy of James Tam

James Tam

5. Deal With Errors In A Helpful And Positive Manner

- (JT: with this the heuristic it states exactly what should be done).

James Tam

Rules Of Thumb For Error Messages

1. Polite and non-intimidating

- Don't make people feel stupid
- Try again, bonehead! ← **No**

2. Understandable

- Error 25 ← **Not**

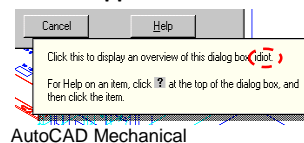
3. Specific

- Cannot open this document ← **Why?**
- Cannot open "chapter 5" because the application "Microsoft Word" is not on your system ← **Better**

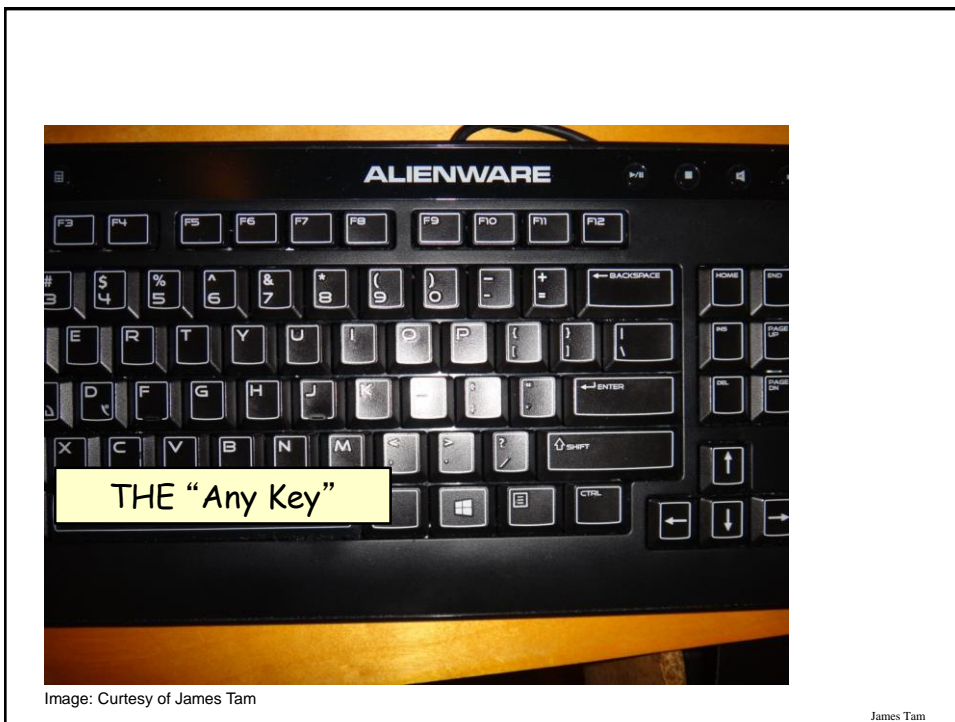
4. Helpful

- Cannot open "chapter 5" because the application "Microsoft Word" is not on your system. Open it with "WordPad" instead?

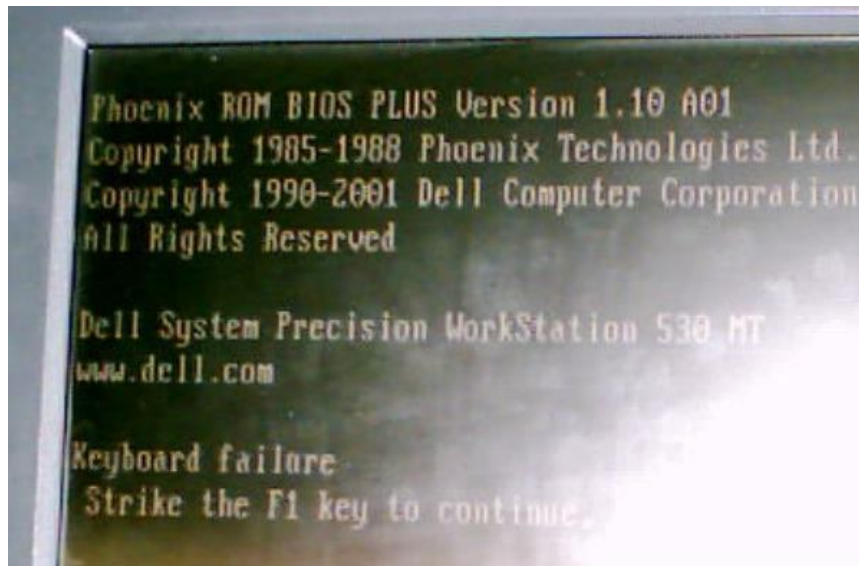
So obvious it could never happen?



James Tam



I'd Rather Deal With The 'Any' Key



Picture courtesy of James Tam: An error message from a Dell desktop computer.

James Tam

After This Section You Should Now Know

- Rules of thumb for designing more user-friendly technology.
 1. Minimize the user's memory load
 2. Be consistent
 3. Provide feedback
 4. Provide clearly marked exits
 5. Deal with errors in a helpful and positive manner

James Tam

Copyright Notification

- “Unless otherwise indicated, all images in this presentation are used with permission from Microsoft.”