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.

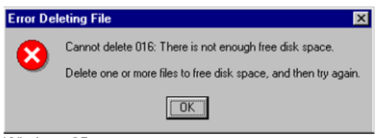
CPSC 217,
Loops In Python: Part 3

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

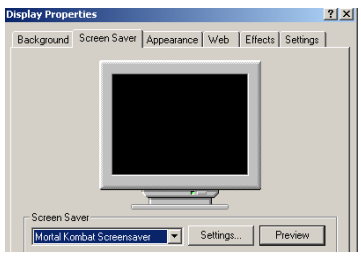
- (The following list comes from Jakob Nielsen's 10 usability heuristics from the book "*Usability Engineering*"
  1. Simple and natural dialog
  2. Minimize the user's memory load
  3. Be consistent
  4. Provide feedback
  5. Provide clearly marked exits
  6. Deal with errors in a helpful and positive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

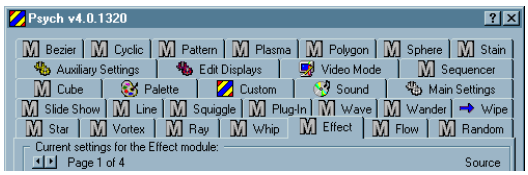
For more information:
- Jakob Nielsen: https://www.nngroup.com/people/jakob-nielsen/
- Book, "Usability Engineering (see Chapter 5)" https://books.google.ca/books?id=DBOowF7LqIQC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

James Tam

1. Simple And Natural Dialogue

- Avoid making the interaction unnecessarily complex

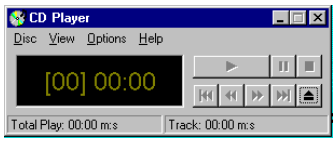




James Tam

1. Simple And Natural Dialogue

- Match the user's conceptual model
- Match the users' task in as natural a way as possible
  - Minimize mapping between interface and task semantics



James Tam

1. Simple And Natural Dialogue

- Present exactly the information the user needs when it is needed
  - Less is more
    - Less to learn, to get wrong, to distract...
    - This not only includes minimizing the information displayed but also the steps needed during an interaction
  - Remove or hide irrelevant or rarely needed information
    - Competes with important information on screen
  - Information should appear in natural order
    - Order of accessing the information matches the user's expectations
    - Related information is graphically clustered
  - Minimize or mitigate modes
  - Use windows frugally
    - Don't make navigation and window management excessively complex

James Tam

1. Simple And Natural Dialogue

- Adding a 'pause' to allow the user to read long text can be of value.

```
input("Hit enter to continue")
```

```
The Hobbit (c) Harper Collins
-----------------------------
This game is a re-creation of the J.R.R. Tolken classic novel: The Hobbit

You will take on the role of Bilbo as he sneaks into the old Dwarven kingdom
Erebor.

This game re-creates the portion of Bilbo's quest just after the secret entrance
tunnel to Erebor has been discovered and a 'volunteer' is needed to go in. The
full version of the game will allow Bilbo to sneak into the Erebor in order to
retrieve as much of the treasure as possible. The treasure is guarded by the
dragon "Smaug the Golden". Although Smaug himself doesn't directly appear in the
game the effects of his dragon fire does.

At the start of the game Bilbo starts off at the entrance to the dwarven hall.
Smaug will be asleep so Bilbo can freely move about (the game is in "mode 1").
Once Bilbo is in the main hall Smaug will awaken and begin bombarding the hall
with fireballs (the game now enters "mode 2").  Bilbo, like all Hobbits, is a
stout fellow but even he cannot withstand dragon fire for long so it will be a
tradeoff between collecting as much treasure as possible and getting out of
Erebor alive.

Hit enter to continue

Once Bilbo has reached the entrance tunnel, Smaug knows that the would be
thief has almost escaped so he will stop breathing fire into the main hall but
```


1. Simple And Natural Dialogue

- Example dialogs that are neither simple nor natural
  - Unnecessary confirmation ("Are you sure?")

```
Enter your selection: 2
Are you sure of the direction? (y/n):
```

  - Adding unnecessary steps (hit enter to continue right after typing in a command)

```
Room actions
(o)pen the door
(l)eft entryway exit
(r)ight entryway exit
Your selection: o
Press enter to continue
The door remains stubbornly locked!
```

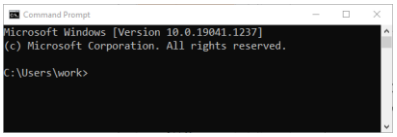
2. Minimize The User's Memory Load

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

These heuristics were covered in a video I made another semester and aside from numbering the content is the same.

2. 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.
    - 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:
```

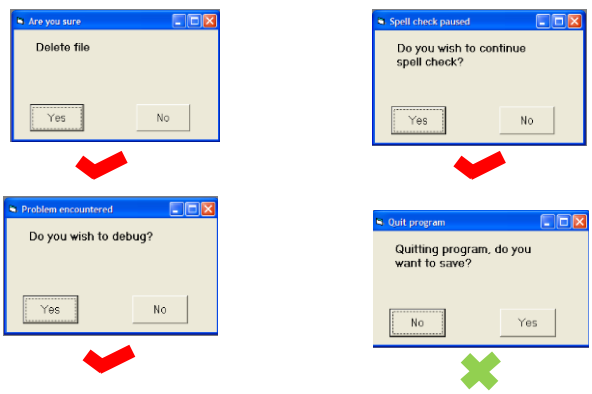
3. Be Consistent

- 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

3. Be Consistent

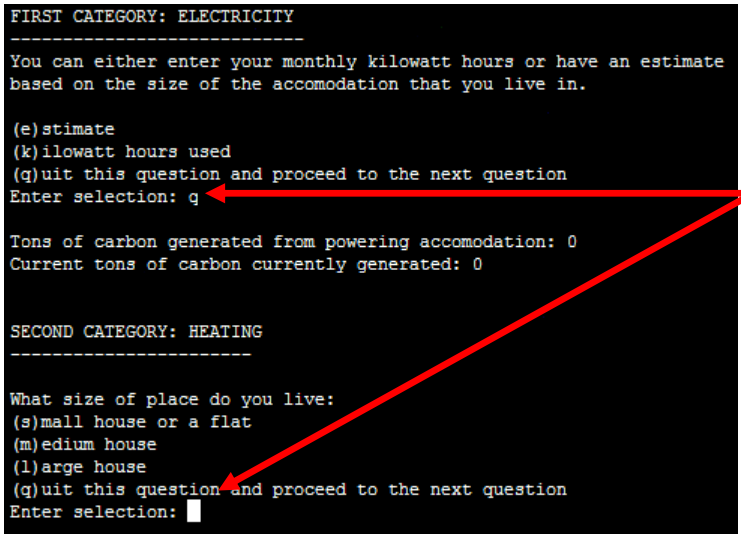
- Consistency of language and graphics
  - Same information/controls in same location on all screens / dialog boxes forms follow boiler plate.
  - Same visual appearance across the system (e.g. widgets).



Images courtesy of James Tam

James Tam

3. Be Consistent



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

James Tam

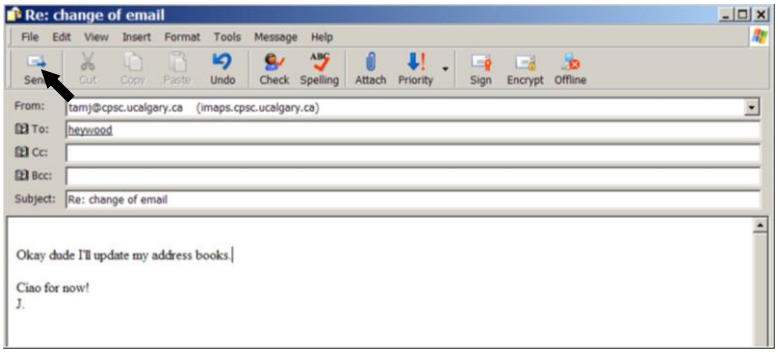
4. Provide Feedback

- 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

4. Provide Feedback

- What is the program doing?

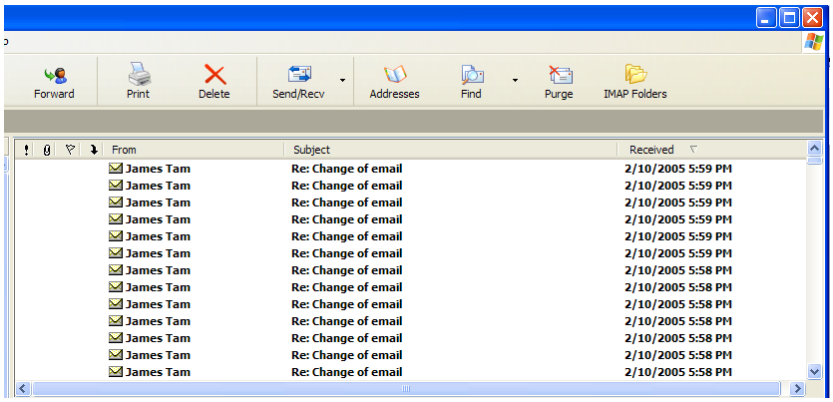


Outlook Express image courtesy of James Tam

James Tam

4. Provide Feedback

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



Outlook Express image courtesy of James Tam

James Tam

4. 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

5. 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

5. 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

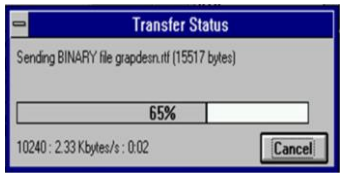
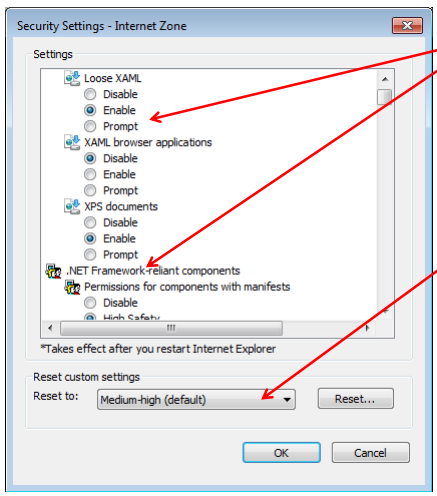


Image: From the "HCI Hall of Shame"

James Tam

5. 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 curtesy of James Tam

James Tam

5. Provide Clearly Marked Exits

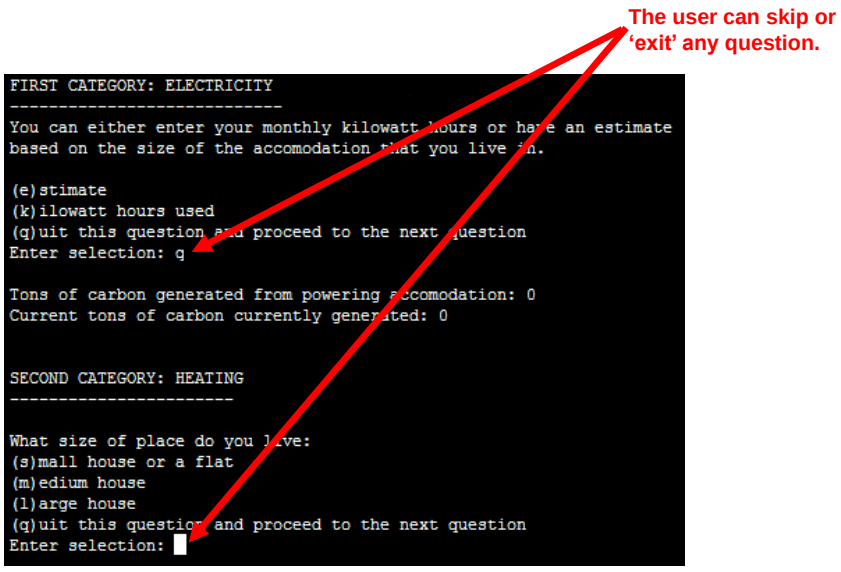


Image: An old CPSC 231 assignment curtesy of James Tam

James Tam

6. Deal With Errors In A Helpful And Positive Manner

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

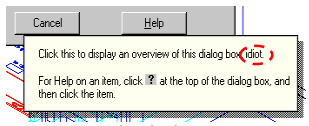
James Tam

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?

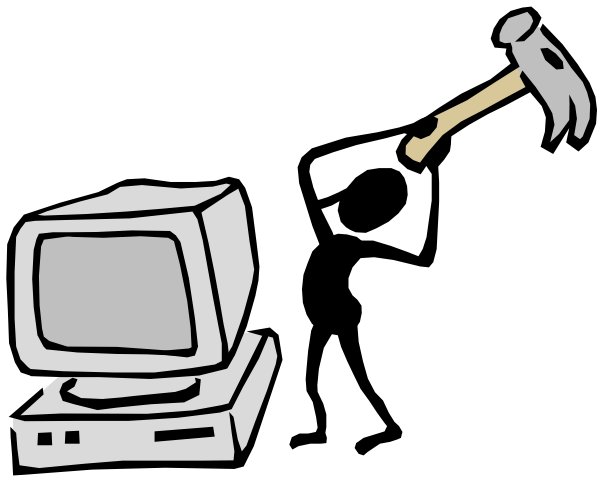
Even better: A potentially helpful suggestion

So obvious it could never happen?



AutoCAD Mechanical

James Tam



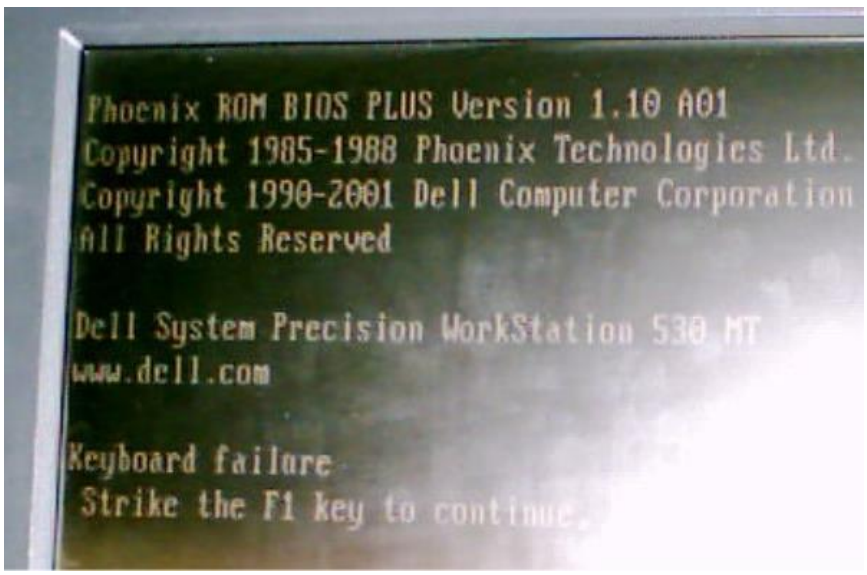
"HIT ANY KEY TO CONTINUE"

James Tam

Image: Curtesy of James Tam

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. (New for subsequent semesters): simple and natural dialog
  2. Minimize the user's memory load
  3. Be consistent
  4. Provide feedback
  5. Provide clearly marked exits
  6. 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."

James Tam