

Loops In Python

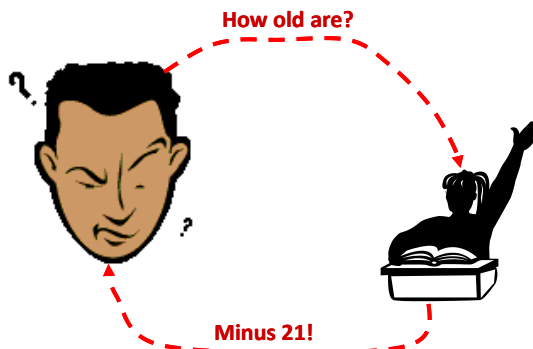
In this section of notes you will learn how to rerun parts of your program without having to duplicate the code.

James Tam

Repetition: Computer View

- Continuing a process as long as a certain condition has been met.

Ask for age as long as answer is negative



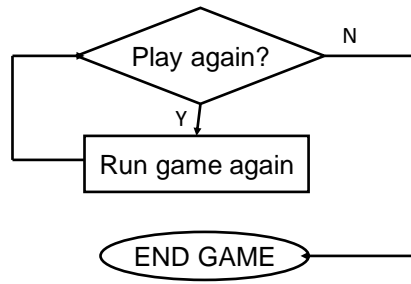
James Tam

How To Determine If Loops Can Be Applied

- Something needs to occur multiple times (generally will repeat itself as long as some condition has been met).
- Example 1:



Re-running the entire program



While the player wants to play
Run the game again

James Tam

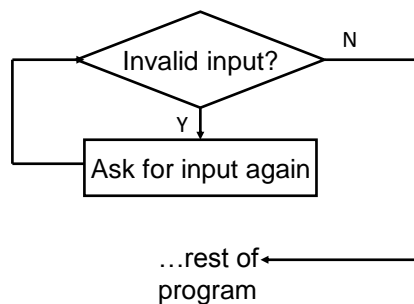
How To Determine If Loops Can Be Applied (2)

- Example 2:

```

Enter your age (must be non-negative): -1
Enter your age (must be non-negative): 27
Enter your gender (m/f): █
  
```

Re-running specific parts of the program



While input is invalid
Prompt user for input

James Tam

Basic Structure Of Loops

Whether or not a part of a program repeats is determined by a loop control (typically just a variable).

- Initialize the control to the starting value
- Testing the control against a stopping condition (Boolean expression)
- Executing the body of the loop (the part to be repeated)
- Update the value of the control

James Tam

Types Of Loops

1.Pre-test loops

- Check the stopping condition *before* executing the body of the loop.
- The loop executes *zero or more* times.

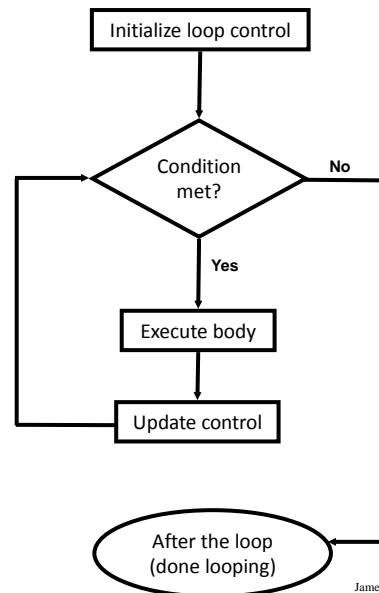
2.Post-test loops

- Checking the stopping condition *after* executing the body of the loop.
- The loop executes *one or more* times.

James Tam

Pre-Test Loops

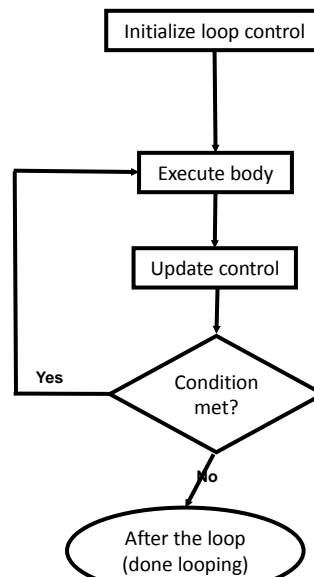
1. Initialize loop control
2. Check if the stopping condition has been met
 - a. If it's been met then the loop ends
 - b. If it hasn't been met then proceed to the next step
3. Execute the body of the loop (the part to be repeated)
4. Update the loop control
5. Go to step 2



James Tam

Post-Test Loops (Not Implemented In Python)

1. Initialize loop control (sometimes not needed because initialization occurs when the control is updated)
2. Execute the body of the loop (the part to be repeated)
3. Update the loop control
4. Check if the stopping condition has been met
 - a. If it's been met then the loop ends
 - b. If it hasn't been met then return to step 2.



James Tam

Pre-Test Loops In Python

1. While
2. For

Characteristics:

1. The stopping condition is checked *before* the body executes.
2. These types of loops execute zero or more times.

James Tam

Post-Loops In Python

- Note: this type of looping construct has not been implemented with this language.
- But many other languages do implement post test loops.

Characteristics:

- The stopping condition is checked *after* the body executes.
- These types of loops execute one or more times.

James Tam

The While Loop

- This type of loop can be used if it's *not known* in advance how many times that the loop will repeat (most powerful type of loop, any other type of loop can be simulated with a while loop).

- **Format:**

- (Simple condition)

```
while (Boolean expression):
    body
```

(Compound condition)

```
while (Boolean expression) Boolean operator (Boolean expression):
    body
```

James Tam

The While Loop (2)

- **Program name:** while1.py

```
i = 1
while (i <= 4):
    print ("i =", i)
    i = i + 1
print ("Done!")
```

1) Initialize control

2) Check condition

3) Execute body

4) Update control

James Tam

The While Loop (2)

- **Program name:** while1.py

```
i = 1
while (i <= 3):
    print ("i =", i)
    i = i + 1
print ("Done!")
```

James Tam

Tracing The While Loop

Execution	Variable
>python while1.py	i

James Tam

Countdown Loop

- **Program name:** while2.py

```
i = 3
while (i >= 1):
    print("i =", i)
    i = i - 1
print("Done!")
```

James Tam

Common Mistakes: While Loops

- Forgetting to include the basic parts of a loop.

- Updating the control

```
i = 1
while (i <= 4):
    print ("i =", i)
```

- (With some languages – not Python - forgetting to initialize the loop control is quite common).

```
int i;
while (i <= 4)
{
    printf("%d", i);
}
```

James Tam

Practice Exercise

- Write the code for the earlier example.
- Modify the following program that prompts for and displays the user's age.
- As long as the user enters a negative age the program will continue prompting for age.
- After a valid age has been entered then stop the prompts and display the age.

```
age = int(input("Age: "))
print(age)
```

James Tam

The For Loop

- Typically used when it *is known* in advance how many times that the loop will execute (counting loop).

- **Syntax:**

```
for <name of loop control> in <something that can be iterated>:
    body
```

- **Program name:** for1.py

```
total = 0
for i in range (1, 4, 1):
    total = total + i
    print ("i=", i, "\ttotal=", total)
print ("Done!")
```

1) Initialize control

2) Check condition

3) Execute body

4) Update control

James Tam

The For Loop

- Typically used when it *is known* in advance how many times that the loop will execute (counting loop).

- **Syntax:**

```
for <name of loop control> in <something that can be iterated>:
    body
```

- **Program name:** for1.py

```
i = 0
total = 0
for i in range (1, 4, 1):
    total = total + i
    print ("i=", i, "\ttotal=", total)
print ("Done!")
```

James Tam

Tracing The First For Loop Example

Execution

```
>python for1.py
```

Variables

```
i          total
```

James Tam

Counting Down With A For Loop

- **Program name:** for2.py

```
i = 0
total = 0
for i in range (3, 0, -1):
    total = total + i
    print ("i = ", i, "\t total = ", total)
print ("Done!")
```

James Tam

Tracing The Second For Loop Example

Execution

```
>python for2.py
```

Variables

```
i          total
```

James Tam

Erroneous For Loops

- The logic of the loop is such that the end condition has already been reached with the start condition.

- **Example:**

```
for i in range (5, 0, 1):  
    total = total + i  
    print ("i = ", i, "\t total = ", total)  
print ("Done!")
```

James Tam

Loop Increments Need Not Be Limited To One

- **While**

```
i = 0  
while (i <= 100):  
    print ("i =", i)  
    i = i + 5  
print ("Done!")
```

- **For**

```
for i in range (0, 105, 5):  
    print ("i =", i)  
print ("Done!")
```

James Tam

Sentinel Controlled Loops

- The stopping condition for the loop occurs when the ‘sentinel’ value is reached.
- **Program name:** sum.py

```
total = 0
temp = 0
while (temp >= 0):
    temp = input ("Enter a non-negative integer (negative to end series):")
    temp = int(temp)
    if (temp >= 0):
        total = total + temp

print ("Sum total of the series:", total)
```

James Tam

Sentinel Controlled Loops (2)

- Sentinel controlled loops are frequently used in conjunction with the error checking of input.
- Example:

```
selection = " "
while selection not in ("a", "A", "r", "R", "m", "M", "q", "Q"):
    print ("Menu options")
    print ("(a)dd a new player to the game")
    print ("(r)emove a player from the game")
    print ("(m)odify player")
    print ("(q)uit game")
    selection = input ("Enter your selection: ")
if selection not in ("a", "A", "r", "R", "m", "M", "q", "Q"):
    print ("Please enter one of 'a', 'r', 'm' or 'q' ")
```

James Tam

Recap: What Looping Constructs Are Available In Python/When To Use Them

Construct	When To Use
Pre-test loops	You want the stopping condition to be checked before the loop body is executed (typically used when you want a loop to execute zero or more times).
<ul style="list-style-type: none"> • While 	<ul style="list-style-type: none"> • The most powerful looping construct: you can write a 'while-do' loop to mimic the behavior of any other type of loop. In general it should be used when you want a pre-test loop which can be used for most any arbitrary stopping condition e.g., execute the loop as long as the user doesn't enter a negative number.
<ul style="list-style-type: none"> • For 	<ul style="list-style-type: none"> • A 'counting loop': You want a simple loop to repeat a certain number of times.
Post-test: None in Python	You want to execute the body of the loop before checking the stopping condition (typically used to ensure that the body of the loop will execute at least once). The logic can be simulated in Python however.

James Tam

The Break Instruction

- It is used to terminate the repetition of a loop which is separate from the main Boolean expression (it's another separate Boolean expression).
- **General structure:**

```

for (Condition 1):
    if (Condition 2):
        break
while (Condition 1):
    if (Condition 2):
        break

```

- Specific example (mostly for illustration purposes at this point):
break.py

```

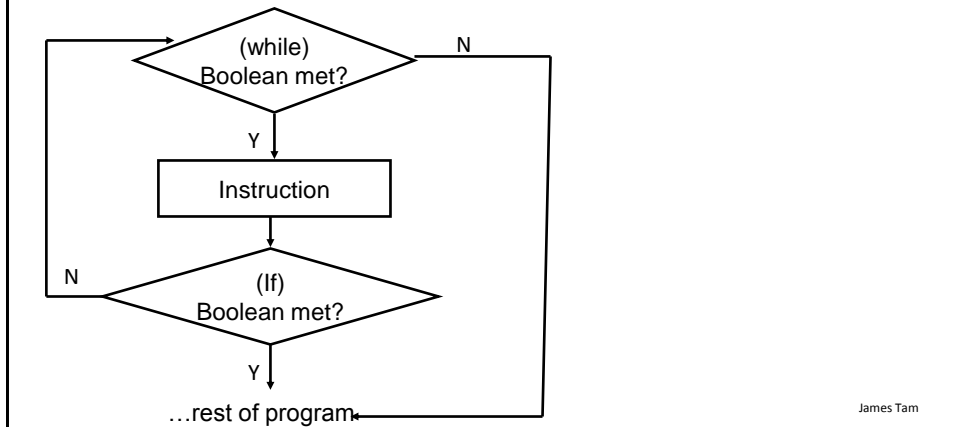
str = input("Enter only lower case alphabetic characters: ")
for temp in str:
    if (temp < 'a') or (temp > 'z'):
        break;
    print(temp)
print("Done")

```

James Tam

The Break Should Be Rarely Used

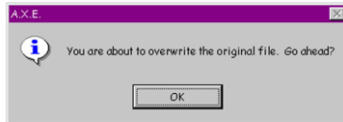
- Adding an extra exit point in a loop (aside from the Boolean expression in the while loop) may make it harder to trace execution (leads to 'spaghetti' programming).



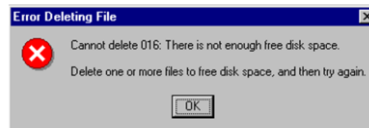
User-Friendly Software

- In today's world it's not just sufficient to create software that has implemented a set of operations.
- If the person using the system cannot understand it or has troubles using common functions then the software or technology is useless.
- Reference course: If you're interested in more information:
 - <http://pages.cpsc.ucalgary.ca/~tamj/2008/481W/index.html>

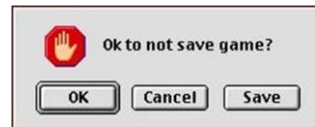
Not So Friendly Examples



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 Rules 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

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.

James Tam

1. Minimize The User's Memory Load

- To reduce the memory load of the user:
 - Describe required the input format, show examples of valid input, provide default inputs
- Examples:

Example 1:

Example 2:

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

James Tam

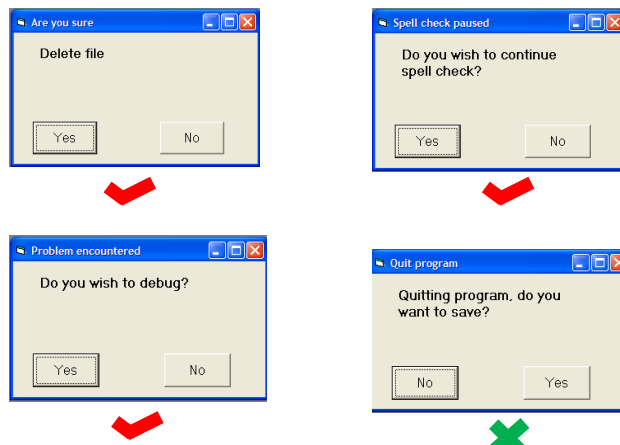
2. Be Consistent

- Consistency of effects
 - 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).



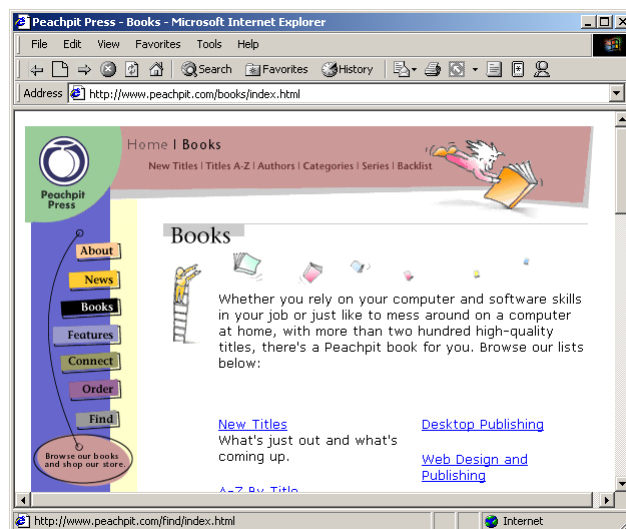
James Tam

2. Be Consistent



James Tam

2. Be Consistent



James Tam

2. Be Consistent

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

```
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: 
```

James Tam

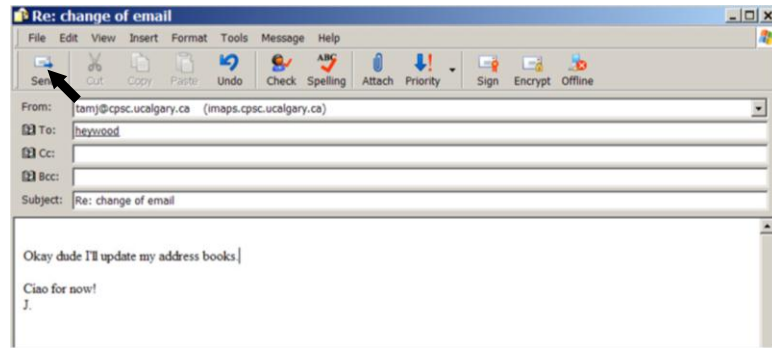
3. Provide Feedback

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

James Tam

3. Provide Feedback

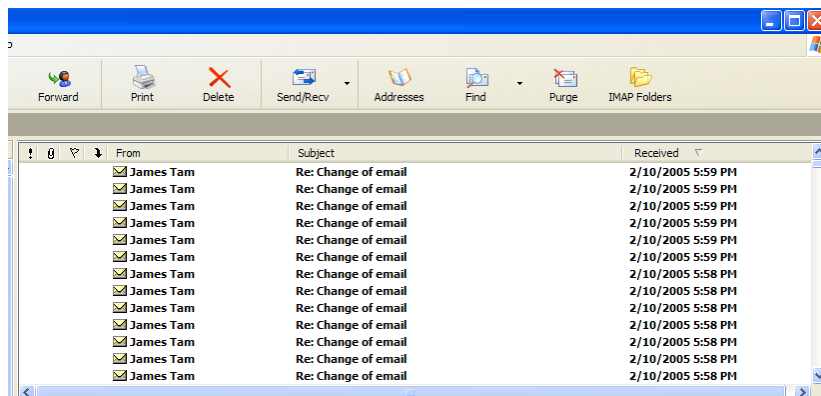
- What is the program doing?



James Tam

3. Provide Feedback

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



James Tam

3. Provide Feedback

- In terms of this course, letting the user know:
 - what the program is doing (e.g., opening a file),
 - what errors may have occurred (e.g., could not open file),
 - and why (e.g., file “input.txt” could not be found)

...is 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 likely be shown to the user as he or she enters the value):
 - `age = int(input("Enter age (0 – 114): "))`

James Tam

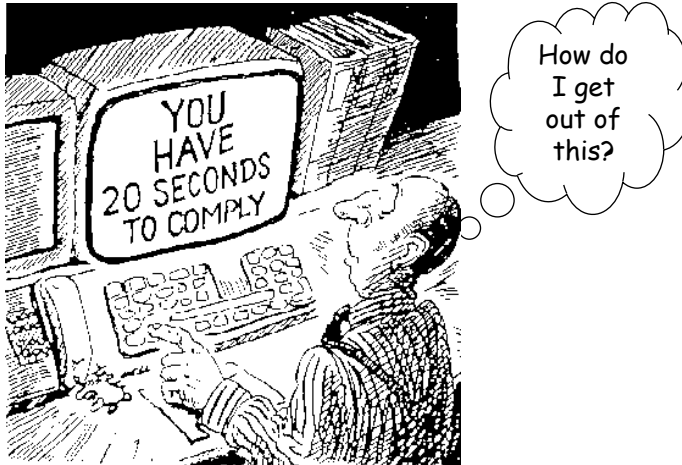
4. Provide Clearly Marked Exits

- This should obviously mean that quitting the program should be 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

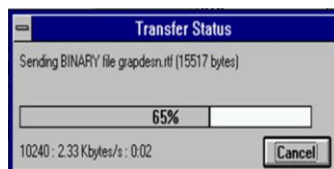
- User's should never feel 'trapped' by a program.



James Tam

4. Provide Clearly Marked Exits

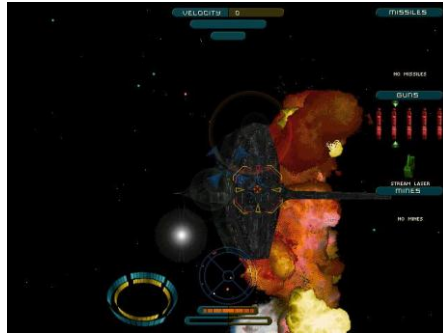
- This doesn't just mean providing an exit from the program but the ability to 'exit' (take back) the current action.
 - Universal Undo/Redo
 - e.g., <Ctrl>-<Z> and <Ctrl>-<Y>
 - Progress indicator & Interrupt
 - Length operations



James Tam

4. Provide Clearly Marked Exits

- Restoring defaults
 - Getting back original settings



Wing Commander: Privateer 2 © Origin-EA

James Tam

4. Provide Clearly Marked Exits

The user can skip any question

```

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:
  
```

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

5. Deal With Errors In A Helpful And Positive Manner



Error handling
in "The good
'ole days"

What is "error 15762"?

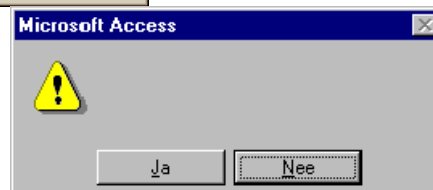
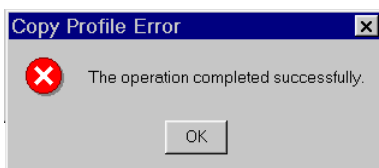
James Tam

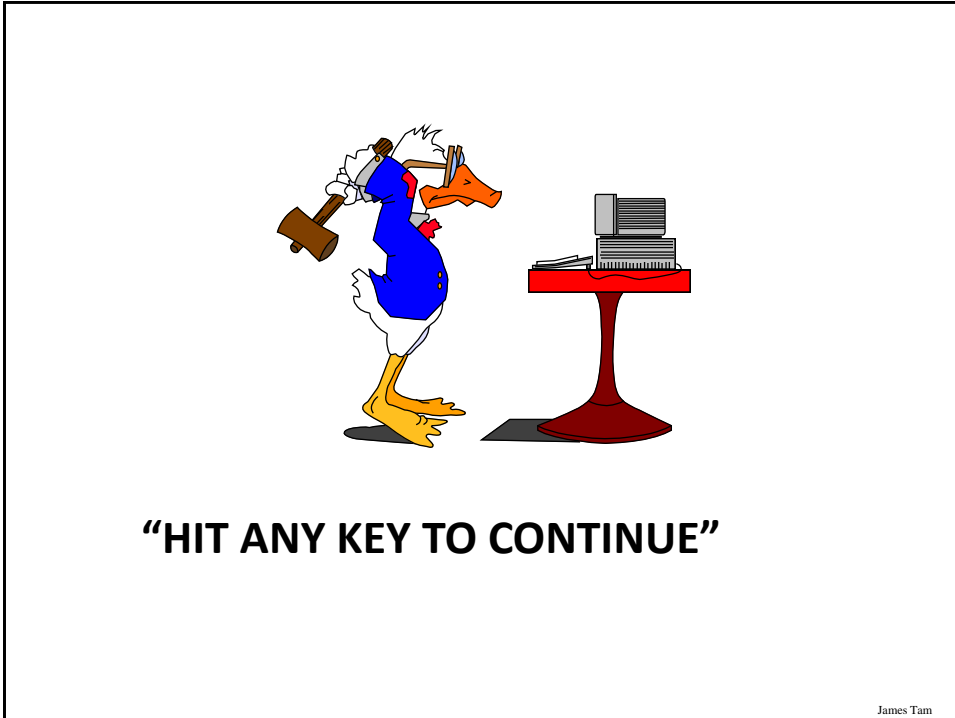
Rules Of Thumb For Error Messages

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

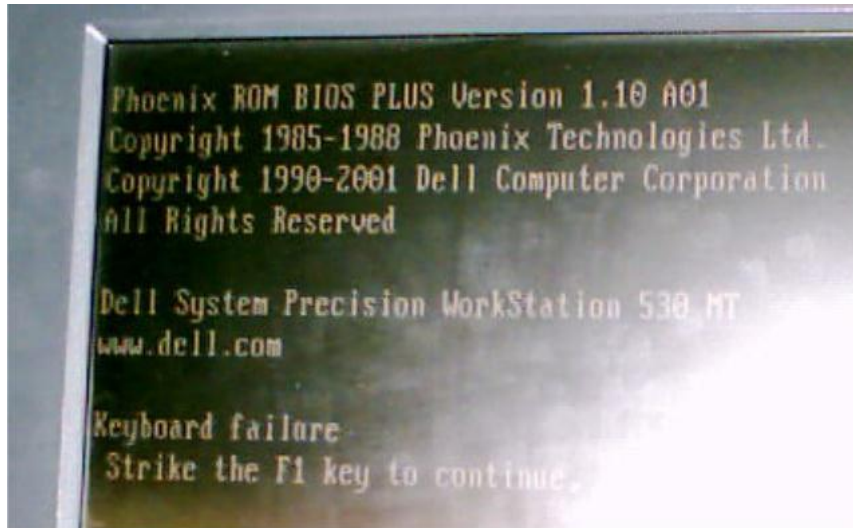
James Tam

Examples Of Bad Error Messages





I Think I'd Rather Deal With The Any Key!!!



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

James Tam

Solving A Problem Using Loops

- **Problem:** Write a program that will execute a game:
 - The program will randomly generate a number between one and ten.
 - Players will be prompted to enter their guess.
 - The program will continue the game until players indicate that they no longer want to continue.
- **Program name:** guessingGame.py

James Tam

Guessing Game

```

guess = 0
answer = 0
choice = "Y"
while choice not in ("q", "Q"):
    answer = random.randrange (10) + 1
    guess = int(input ("Enter your guess: "))
    if (guess == answer):
        print ("You guessed correctly!")
    else:
        print ("You guessed incorrectly")
    print ("Number was", answer, ", your guess was", guess)
    print ("Play again? Enter 'q' to quit, anything else to play again")
    choice = input("Choice: ")
    print ()
print ("Exiting game")

```

James Tam

Infinite Loops

- Infinite loops never end (the stopping condition is never met).
- They can be caused by logical errors:
 - The loop control is never updated (Example 1 – below).
 - The updating of the loop control never brings it closer to the stopping condition (Example 2 – next slide).
- **Example 1:** infinite1.py


```

i = 1
while (i <=10):
    print ("i = ", i)
    i = i + 1

```

To stop a program with an infinite loop in Unix simultaneously press the <ctrl> and the <c> keys

James Tam

Infinite Loops (2)

- **Example 2:** infinite2.py

```
i = 10
while (i > 0):
    print ("i = ", i)
    i = i + 1
```

To stop a program with an infinite loop in Unix simultaneously press the <ctrl> and the <c> keys

James Tam

Nested Loops

- One loop executes inside of another loop(s).
- Example structure:

```
Outer loop (runs n times)
  Inner loop (runs m times)
    Body of inner loop (runs n x m times)
```

- **Program name:** nested.py

```
i = 1
while (i <= 2):
    j = 1
    while (j <= 3):
        print("i = ", i, " j = ", j)
        j = j + 1
    i = i + 1
print("Done!")
```

James Tam

Testing Loops

- Make sure that the loop executes the proper number of times.
- Test conditions:
 - 1) Loop does not run
 - 2) Loop runs exactly once
 - 3) Loop runs exactly 'n' times

James Tam

Testing Loops: An Example

```
sum = 0
i = 1
last = 0

last = int(input("Enter the last number in the sequence to sum : "))
while (i <= last):
    sum = sum + i
    print("i = ", i)
    i = i + 1

print("sum =", sum)
```

James Tam

Extra Practice

- Write a loop that will continue repeating if the user enters a value that is negative.
- Write a program that will prompt the user for number and an exponent. Using a loop the program will calculate the value of the number raised to the exponent.

James Tam

After This Section You Should Now Know

- When and why are loops used in computer programs
- What is the difference between pre-test loops and post-test loops
- How to trace the execution of pre-test loops
- How to properly write the code for a loop in a program
- Some rules of thumb for interaction design
 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
- What are nested loops and how do you trace their execution
- How to test loops

James Tam