

CPSC 231:

Loops In Python

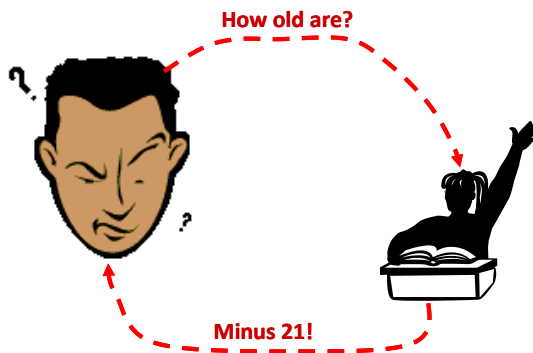
In this section of notes you will learn how to rerun parts of your program without duplicating instructions.

James Tam

Repetition: Computer View

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

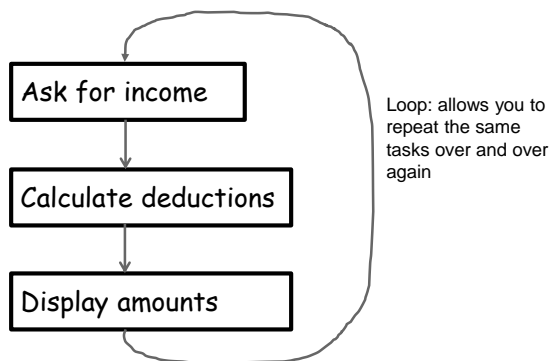
Ask for age as long as the answer is negative (outside allowable range)



James Tam

Looping/Repetition

- How to get the program or portions of the program to automatically re-run
 - Without duplicating the instructions
 - Example: you need to calculate tax for multiple people



James Tam

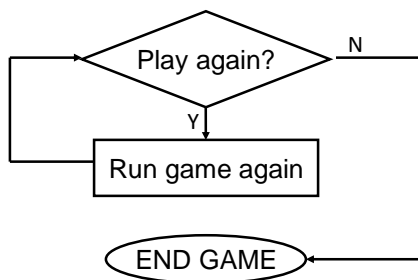
How To Determine If Loops Can Be Applied

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



Re-running the entire program

Flowchart



Pseudo code

While the player wants to play
Run the game again

James Tam

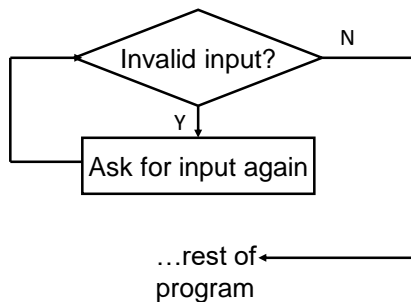
How To Determine If Loops Can Be Applied (2)

- Example 2:

```
Enter your age (must be non-negative): -1
Enter your age (must be non-negative): 37
Enter your height (must be non-negative):
```

Re-running specific parts of the program

Flowchart



Pseudo code

```
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 the control is just a variable).

- Initialize the control to the starting value
- Executing the body of the loop (the part to be repeated)
- Update the value of the control

- Somewhere: Testing the control against a stopping condition (Boolean expression)
 - Without this test the loop will never end (endless loop)

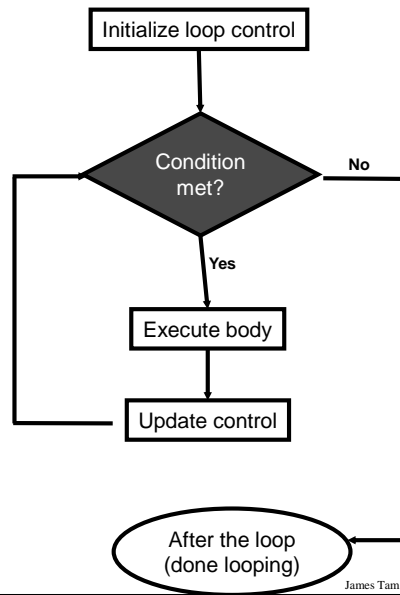
James Tam

Pre-Test Loops

Pre-test loops

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

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

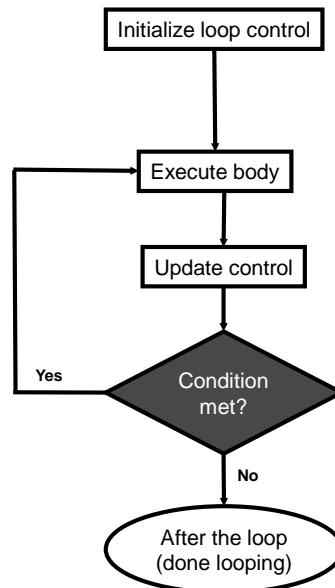


Post-Test Loops (Not Implemented In Python)

Post-test loops

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

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 repetition condition has been met
 - a. If the condition has been met then go through the loop again (go to Step 2)
 - b. If the condition hasn't been met then the loop ends.



Loops In Python

- Pre-test: for, while
- Post-test: none

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).
 - It can repeat so long as some arbitrary condition holds true.

- 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 <= 3):
    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

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)
    # i = i + 1
```



```
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
```

James Tam

Practice Exercise #1

- The following program that prompts for and displays the user's age.
- Modifications:
 - 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)
```

```
Age: -1
Age cannot be negative
Age: -123
Age cannot be negative
Age: 27
You are 27 years young :P
```

James Tam

The For Loop

- In Python a for-loop is used to step through a sequence e.g., count through a series of numbers or step through the lines in a file.

- Syntax:**

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

- Program name:** for1.py

```
for i in range (1, 4, 1):
    print("i=", i)
print("Done!")
```

1) Initialize control

2) Check condition

4) Update control

3) Execute body

James Tam

The For Loop

- In Python a for-loop is used to step through a sequence

- **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!")
```

```
i= 1  
i= 2  
i= 3  
Done!
```

James Tam

Counting Down With A For Loop

- **Program name:** for2.py

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

James Tam

For Loop: Stepping Through A Sequence Of Characters

- Recall: A for-loop in Python can step through any iterable sequence (number sequence, characters in a string, lines in a file).

- Example: for3.py

```
activity = input("What are you doing with dog now: ")
print("We are taking the dog for a '", end="")
```

```
We are taking the dog for a '
```

```
for ch in activity:
    print(ch + "-", end="")
print("")
```

```
b-a-t-h-
```

James Tam

Erroneous For Loops (If There Is Time)

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

- Example: for_error.py

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

```
[csc loops 18 ]> python for_error.py
Done!
```

James Tam

Loop Increments Need **Not Be Limited To One**

- **While:** while_increment5.py

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

```
i = 0
i = 5
i = 10
i = 15
i = 20
i = 25
i = 30
i = 35
i = 40
i = 45
i = 50
i = 55
i = 60
i = 65
i = 70
i = 75
i = 80
i = 85
i = 90
i = 95
i = 100
Done!
```

- **For:** for_increment5.py

```
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 e.g. sentinel: number less than zero (negative)

- **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)
```

```
Enter a positive integer (negative to end series):1
Enter a positive integer (negative to end series):2
Enter a positive integer (negative to end series):3
Enter a positive integer (negative to end series):-1
Sum total of the series: 6
```

Q: What if the user just entered a single negative number?

James Tam

Sentinel Controlled Loops (2)

- Sentinel controlled loops are frequently used in conjunction with the error checking of input.
- Example (sentinel value is one of the valid menu selections, repeat while selection is not one of these selections)

```

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' ")
    
```

```

Menu options
(a)dd a new player to the game
(r)emove a player from the game
(m)odify player
(q)uit game
Enter your selection: x
Please enter one of 'a', 'r', 'm' or 'q'
    
```

```

Menu options
(a)dd a new player to the game
(r)emove a player from the game
(m)odify player
(q)uit game
Enter your selection: A
    
```

Valid option entered, loop ends

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).
• While	• The most powerful looping construct: you can write a 'while' 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.
• For	• In Python it can be used to step through some sequence
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 with a while loop.

James Tam

Common Mistake #1

- Mixing up branches (IF and variations) vs. loops (while)
- Related (both employ a Boolean expression) but they are not identical
- Branches
 - General principle: If the Boolean evaluates to true then execute a statement or statements (**once**)
 - Example: display a popup message if the number of typographical errors exceeds a cutoff.
- Loops
 - General principle: As long as (or while) the Boolean evaluates to true then execute a statement or statements (**multiple times**)
 - Example: While there are documents in a folder that the program hasn't printed then continue to open another document and print it.

James Tam

Common Mistake #1: Example

- **Name of the file containing the program:** branchVsLoop.py
- Contrast

```
age = int(input("Age positive only: "))
if (age < 0):
    age = int(input("Age positive only: "))
print("Branch:", age)
```

Vs.

```
age = int(input("Age positive only: "))
while (age < 0):
    age = int(input("Age positive only: "))
print("Loop:", age)
```

James Tam

Nesting

- Recall: Nested branches (one inside the other)

- Nested branches:

```
If (Boolean):  
    If (Boolean):  
        ...
```

- Branches and loops (for, while) can be nested within each other

```
# Scenario 1  
loop (Boolean):  
    if (Boolean):  
        ...  
  
# Scenario 2  
if (Boolean):  
    loop (Boolean):  
        ...  
  
# Scenario 3  
loop (Boolean):  
    loop (Boolean):  
        ...
```

James Tam

Recognizing When Looping & Nesting Is Needed

- **Scenario 1:** As long some condition is met a question will be asked. As the question is asked if the answer is invalid then an error message will be displayed.

- Example: While the user entered an invalid value for age (too high or too low) then if the age is too low an error message will be displayed.

- Type of nesting: an IF-branch nested inside of a loop

```
loop (Boolean):  
    if (Boolean):  
        ...
```

James Tam

IF Nested Inside A While

- Word document containing the example:

nestingIFinsideWHILE.py

```
age = - 1
MIN_AGE = 1
MAX_AGE = 118
age = int(input("How old are you (1-118): "))
while ((age < MIN_AGE) or (age > MAX_AGE)):
    if (age < MIN_AGE):
        print("Age cannot be lower than", MIN_AGE, "years")
    age = int(input("How old are you (1-118): "))

print("Age=", age, "is age-okay")
```

James Tam

Recognizing When Looping & Nesting Is Needed

- **Scenario 2:** If a question answers true then check if a process should be repeated.

- Example: If the user specified the country of residence as Canada then repeatedly prompt for the province of residence as long as the province is not valid.

- Type of nesting: a loop nested inside of an IF-branch

```
If (Boolean):
    loop ():
        ...
```

James Tam

While Nested Inside An IF

- Word document containing the example:

nestingWHILEinsideIF.py

```
country = ""
province = ""
country = input("What is your country of citizenship: ")
if (country == "Canada"):
    province = input("What is your province of citizenship: ")
    while ((province != "AB") and (province != "BC")):
        print("Valid provinces: AB, BC")
        province = input("What is your province of citizenship: ")
    print("Country:", country, ", Province:", province)
```

James Tam

Recognizing When Looping & Nesting Is Needed

- **Scenario 3:** While one process is repeated, repeat another process.
 - More specifically: for each step in the first process repeat the second process from start to end
 - Example: While the user indicates that he/she wants to calculate another tax return prompt the user for income, while the income is invalid repeatedly prompt for income.
 - Type of nesting: a loop nested inside of an another loop
- ```
Loop():
 Loop():
 ...
```

James Tam

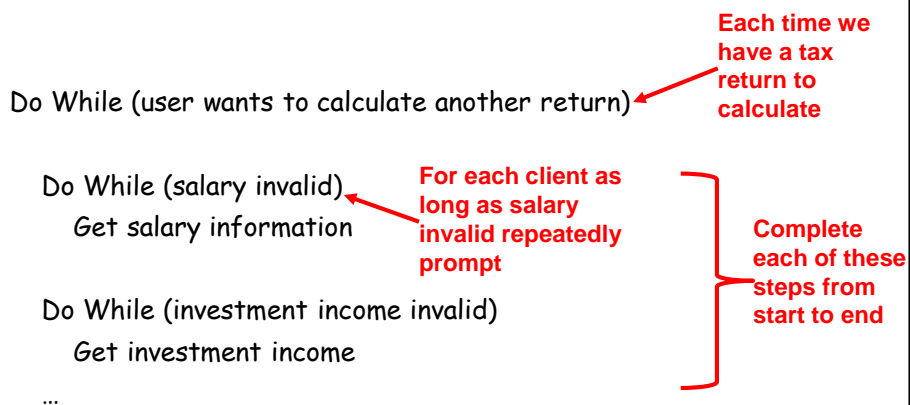


## Pseudo Code

- A high level solution or algorithm that is not specified in a programming language.
- Instead English-like statements are used.
  - “A high-level description of the actions of a program or algorithm, using a mixture of English and informal programming language syntax” – Python for Everyone (Horstmann, Necaise)
- Benefits: it allows the programmer to focus on the solution without spending a lot time worrying about details such as syntax.

James Tam

## Nested Loop: Example Process In Pseudo Code



James Tam

## While Nested Inside Another While

- Word document containing the example:  
nestingWHILEinsideWHILE.py

```
MIN_INCOME = 0
runAgain = "yes"
while (runAgain == "yes"):
 print("CALCULATING A TAX RETURN")
 income = -1
 while (income < MIN_INCOME):
 income = int(input("Income $"))
 runAgain = input("To calculate another return enter 'yes': ")
```

James Tam

## Analyzing Another Nested Loop

- 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!")

```
i = 1 j = 1
i = 1 j = 2
i = 1 j = 3
i = 2 j = 1
i = 2 j = 2
i = 2 j = 3
Done!
```

James Tam

## Practice Example #2: Nesting

1. Write a program that will count out all the numbers from one to six.
  2. For each of the numbers in this sequence the program will determine if the current count (1 – 6) is odd or even.
    - a) The program display the value of the current count as well an indication whether it is odd or even.
- Which Step (#1 or #2) should be completed first?

James Tam

## Step #1 Completed: Now What?

- For each number in the sequence determine if it is odd or even.
- This can be done with the modulo (remainder) operator: %
  - An even number modulo 2 equals zero (2, 4, 6 etc. even divide into 2 and yield a remainder or modulo of zero).
  - `if (counter % 2 == 0): # Even`
  - An odd number modulo 2 does not equal zero (1, 3, 5, etc.)
- Pseudo code visualization of the problem
  - Loop to count from 1 to 6
  - Determine if number is odd/even and display message
  - End Loop
  - Determining whether a number is odd/even is a part of counting through the sequence from 1 – 6, checking odd/even is nested within the loop

James Tam

## The Break Instruction

Q: What if the user just typed 'abc' and hit enter?

- 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): while (Condition 1):
 if (Condition 2): if (Condition 2):
 break break
```

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

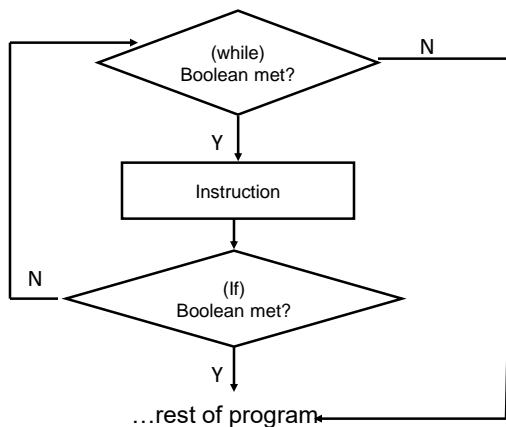
```
str1 = input("Enter a series of lower case alphabetic characters: ")
for temp in str1:
 if ((temp < 'a') or (temp > 'z')):
 break
 print(temp)
print("Done")
```

```
Enter a series of lower case alphabetic characters: abcD
a
b
c
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).



**JT: While adding a single break may not always result in 'spaghetti' it's the beginning of a bad habit that may result in difficult to trace programs**

James Tam

## An Alternate To Using A 'Break'

- Instead of an 'if' and 'break' inside the body of the loop

```
while (BE1):
 if (BE2):
 break
```

- Add the second Boolean expression as part of the loop's main Boolean expression

```
while ((BE1) and not (BE2)):
```

James Tam

## Another Alternative To Using A 'Break'

- If the multiple Boolean expressions become too complex consider using a 'flag'

```
flag = True
while (flag == True):
 if (BE1):
 flag = False
 if (BE2):
 flag = False
 # Otherwise the flag remains set to true
 # BE = A Boolean expression
```

- Both of these approaches still provide the advantage of a single exit point from the loop.

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
```

```
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
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
print("Done!")
```

```
i = 14477
i = 14478
i = 14479
i = 14480
i = 14481
i = 14482
i = 14483
```

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

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

Program name: testing.py

```
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)
```

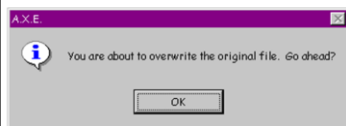
James Tam

## Extra Practice #3

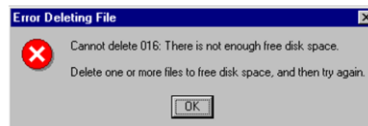
- Write a loop that will continue repeating if the user enters a value that is negative.
- Write a program that will prompt the user for number and an exponent. Using a loop the program will calculate the value of the number raised to the exponent.
  - To keep it simple you can limit the program to non-negative exponents.

James Tam

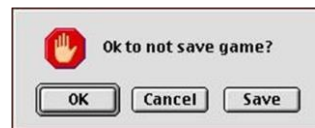
## Not So Friendly Examples (If There Is Time)



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 (Of Thumb) For Designing Software (If There Is Time)**

- (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 (If There Is Time)**

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

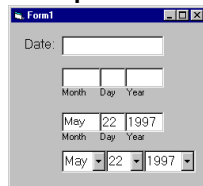
slide 52

James Tam

## 1. Minimize The User's Memory Load (If There Is Time)

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

### Example 1:



The screenshot shows a window titled 'Form1' with a 'Date:' label. Below it are three input fields for 'Month', 'Day', and 'Year'. The 'Month' field contains 'May', 'Day' contains '22', and 'Year' contains '1997'. Below these is a single dropdown menu that also displays 'May 22 1997'.

### Example 2:

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

James Tam

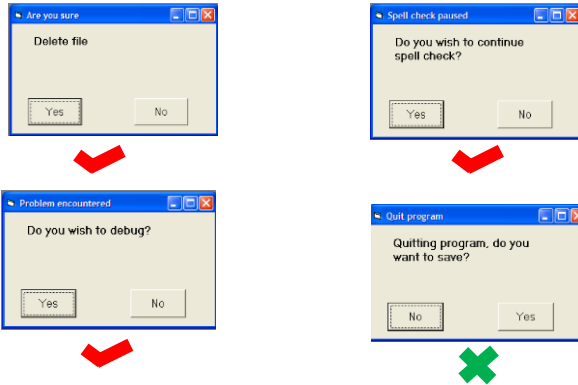
## 2. Be Consistent (If There Is Time)

- 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 (If There Is Time)

- 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 courtesouey of James Tam

James Tam

## 2. Be Consistent (If There Is Time)

```
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 allows the user to proceed to the next question.

James Tam

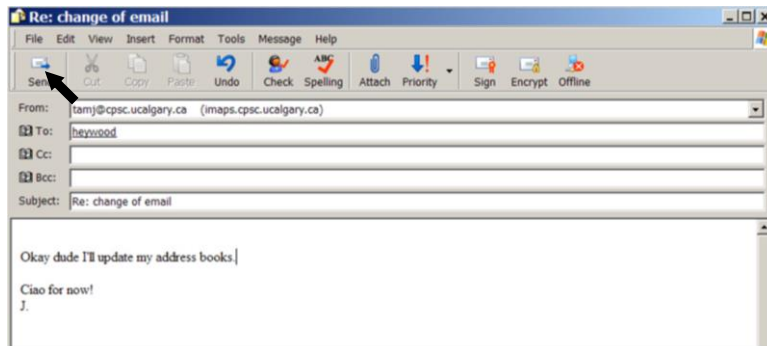
### 3. Provide Feedback (If There Is Time)

- 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 (If There Is Time)

- What is the program doing?

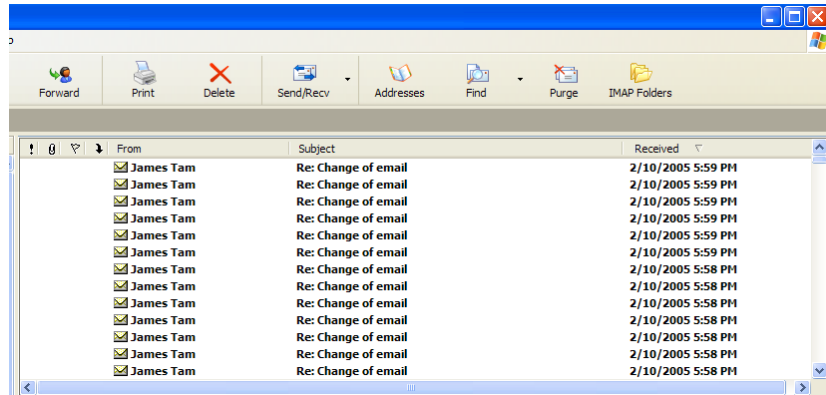


Outlook Express image courteously  
of James Tam

James Tam

### 3. Provide Feedback (If There Is Time)

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



Outlook Express image courteously  
of James Tam

James Tam

### 3. Provide Feedback (If There Is Time)

- 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 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 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 (If There Is Time)

- 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 (If There Is Time)

- 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

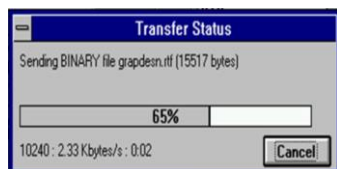
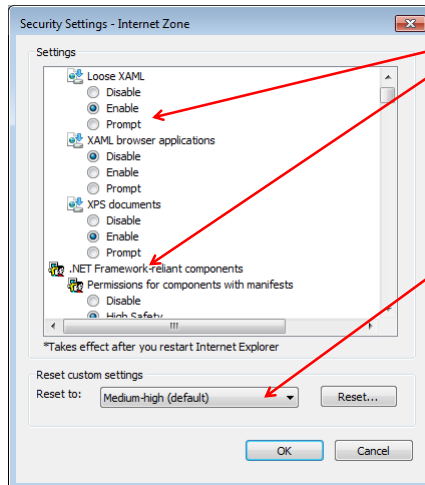


Image: From the "HCI Hall of Shame"

James Tam

## 4. Provide Clearly Marked Exits (If There Is Time)

- 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 (If There Is Time)

```
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: █
```

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 (If There Is Time)

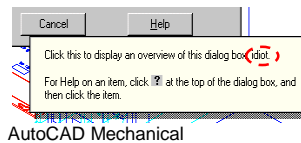
- (JT: with this the heuristic it states exactly what should be done).

James Tam

## Rules Of Thumb For Error Messages (If There Is Time)

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?

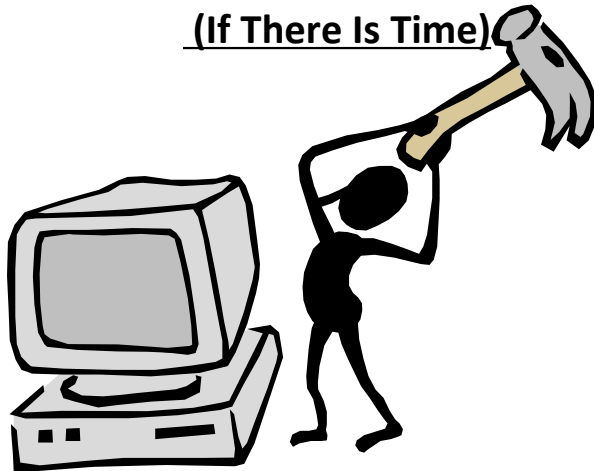


Even better: A potentially helpful suggestion

James Tam



(If There Is Time)



"HIT ANY KEY TO CONTINUE"

James Tam

(If There Is Time)

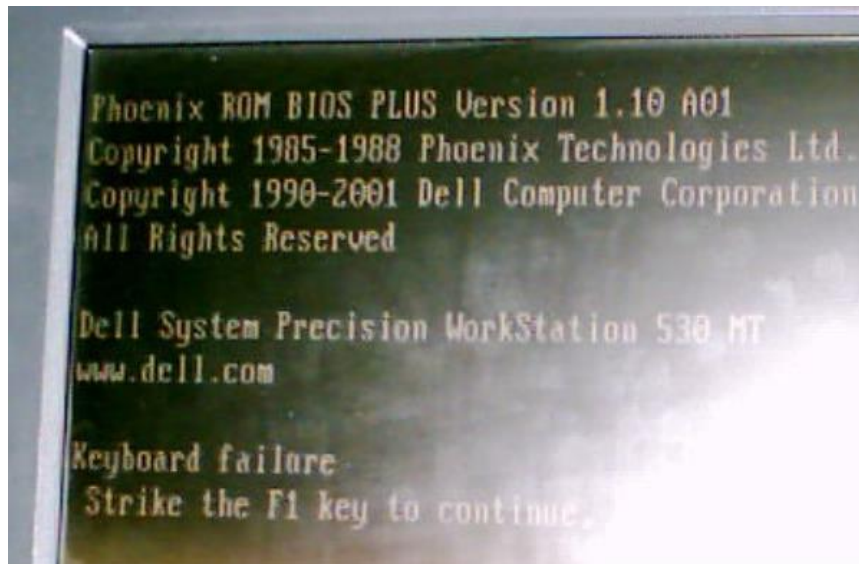


THE "Any Key"

Image: Courtesy of James Tam

James Tam

## I'd Rather Deal With The 'Any' Key (If There Is Time)



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

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
- What are nested loops and how do you trace their execution
- How to test loops
- Some rules of thumb for interaction design (if there is time)
  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.”