## As you arrive:

1. Start up your computer and plug it in
2. ***Log into Angel*** and go to CSSE 120
3. Do the ***Attendance Widget*** – the PIN is on the board
4. Go to the course ***Schedule Page***
5. Open the ***Slides*** for today if you wish
6. Check out today's project: `11-WhileLoops`

*Plus in-class time working on these concepts AND practicing previous concepts, continued as homework.*

## Exam 1 preview

- Date and time of exam
- Exam location
- Format of exam (paper part + programming part)
- How to prepare for the exam

## Indefinite Loops

`while` statements

`break` statements

# *Checkout today's project:* `11-WhileLoops`

***Troubles getting today's project? If so:*** →

***Are you in the Pydev perspective? If not:***

- **`Window ~ Open Perspective ~ Other`** then **`Pydev`**

***Messed up views? If so:***

- **`Window ~ Reset Perspective`**

***No SVN repositories view (tab)? If it is not there:***

- **`Window ~ Show View ~ Other`** then **`SVN ~ SVN Repositories`**

***In your SVN repositories view (tab), expand your repository (****the top-level item) if not already expanded.*

- If no repository, perhaps you are in the wrong Workspace. Get help as needed.

***Right-click on today's project,*** *then select* ***`Checkout`****. Press* ***`OK`*** *as needed.*

The project shows up in the **`Pydev Package Explorer`** to the right. Expand and browse the modules under **`src`** as desired.

# Outline of Today's Session

- Exam 1
  - What to bring
  - How to prepare
- Questions?
- 5 Big Ideas for Exam 1
- Definite Loops (review)
- Indefinite Loops
  - Indefinite versus definite loops
  - `while` statement
  - `break` statement

*Checkout today's project:*
**`11-WhileLoops`**

***Practice, practice, practice!***

- For Exam 1

***Not*** on Exam 1

# Exam 1 – What to bring

- **Exam is Tuesday *evening***
- **No regular class Tuesday *afternoon***

- **When? Where? See schedule page**
- **Format:** *Paper-and-Pencil* and *On-the-Computer.* 50 points each.
- **What to bring:**
  - For the *Paper-and-Pencil* part:
    - Your own textbook (***Zelle***)
    - Your ***own cheat sheet*** – One 8.5 by 11 page (both sides),with whatever you want on it. Prepare this carefully!
  - For the *On-the-Computer* part :
    - ***Any printed or handwritten material you choose*** (notes, books, printouts, …)
    - Your ***computer***, with ***power adapter*** and ***network cable***
      - Computer: You may access anything on it, for this part.
      - ***Network: You may access ONLY your own SVN repository and any material directly reachable from the CSSE 120 Angel and web sites for this term***

***How to prepare?***

- See next slide

***Note!***

# Exam 1 – How to prepare

- **Exam is Tuesday *evening***
- **No regular class Tuesday *afternoon***

- ***Topics And Sample Problems*** document
  - Linked from Schedule Page for today
  - Very long, because it is very thorough. Use it like this:
    - Skim pages 2-3 (for Paper-and-Pencil part) and 4-5 (for On-the-Computer part). For each, circle the items that you are unsure about.
    - Talk to someone – your instructor, Review Session people, classmates, trusted friends – about your circled items. Make notes as needed.
    - As time permits, choose some Practice Problems to try – ones that you *don't* know how to do (skip the ones that you *do* know how to do) and that you think might help your learning and your score on the test. Strive for big ideas first.
    - As time permits, either solidify your understanding of the big-ticket ideas, or go back to pages 6-7 for lesser items on the On-the-Computer part and repeat step 3 on those. Try more Practice Problems, either big-ticket ideas of the lesser ideas, depending on where you are in your mastery of this course.
- Review session: Monday 8 p.m. to 10 p.m., CSSE lab (Moench F-217)
- Homework problems: Recall Amnesty through Wednesday!

Q3-4

# Exam 1 – Big Idea #1: The *input-compute-output pattern*

- The input-compute-output pattern

```
def chaos():
    '''
        Computes and prints a chaotic sequence of numbers,
        as a function of a number input from the user.
    '''
    print("This function illustrates a chaotic function.")
    x = float(input("Enter a number between 0 and 1: "))

    for k in range(10):
        x = 3.9 * x * (1 - x)
        print(x)
    print("Chaotic number after 10 iterations is", x)
```

# Exam 1 – Big Idea #2:
# Functions: *Defining* & *Calling*

```
def factorial(n):
    '''
        Returns n!. That is, returns n * (n-1) * (n-2) * ... * 1.
        Returns 0 if n < 1.  Assumes n is an integer.
    '''
    product = 1
    for k in range(1, n + 1):
        product = product * k

    return product

def main():
    ''' Prints a table of factorial values. '''
    for k in range(21):
        kFactorial = factorial(k)
        print("{}! is {}".format(k, kFactorial))
```

# Exam 1 – Big Idea #3a: *Definite loops, the Accumulation Loop pattern*

```
def countedLoop(iterationsToRun):
    '''
        Returns a partial sum of:
          cosine(0) + cosine(1) + cosine(2) + ...

        Stops after summing the given number of terms
    '''
    sum = 0
    for k in range(iterationsToRun):
        sum = sum + math.cos(k)

    return sum
```

# Exam 1 – Big Idea #3b: *Definite loops, looping through a sequence TWO WAYS*

```
def loopThroughSequenceDirectly(sequence):
    ''' Prints the items in the sequence, each on own line. '''
    for item in sequence:
        print(item)

def loopThroughSequenceUsingIndices(sequence):
    ''' Prints the items in the sequence, each on own line. '''
    for k in range(len(sequence)):
        print(sequence[k])
```

# Exam 1 – Big Idea #4: *Objects*

- Objects
  - *Constructing*
  - Applying *methods*
  - Referencing *instance variables*
- How to determine what methods apply
- Difference between an object and the class that the object is an instance of

```
def moveCircle(circle, window):
    circle.setFill('PaleVioletRed3')
    circle.draw(window)

    for k in range(100):
        sleep(.05)
        circle.move(2, -3)

def main():
    window = GraphWin("Moving circle",
                      500, 500)
    center = Point(50, 450)
    radius = 40
    circleToMove = Circle(center, radius)
    moveCircle(circleToMove, window)

    window.getMouse()
    window.close()
```

# Exam 1 – Big Idea #5: ***Debugging***

- ***Syntax errors: Read the error message.***
  - Often, the error is on the line just *BEFORE* the indicated line.
- When the program does not run right:
  - Look for ***RED*** in the Console window. Decipher its very important message.
  - The ***BLUE*** link in the Console window takes you to the offending line.
  - Use the Debugger to track down hard-to-debug errors
    - Use breakpoints/stepping to locate the place where things are going wrong
    - Use the Variables View (window) to "wake you up" as to what is going wrong
- ***You cannot debug the program if you don't know what the program is supposed to do!***
  - Use a concrete example to understand WHAT the program is to do.
  - Use that same example to figure out by hand HOW the program should do it.
  - Draw upon patterns/ideas that you have seen, e.g. the Accumulator Loop pattern and looping through a sequence.

# Definite Loops (review)

- ***Definite*** loop:
  - Knows *before the loop starts to execute* the number of iterations of the loop body
  - Usually implemented by using a `for` statement
- Syntax: `for` *loop-variable* `in` *sequence*`:`
  *body*
- Examples: **1-DefiniteLoops.py** in today's project
  - ***Counted*** loop: A loop where the sequence is a range expression
  - ***Loop through a sequence***:
    - ***Directly***
    - ***Using indices*** generated by a ***range*** statement
  - ***Loop through a file***

# Is This Loop a Definite Loop?

```
# Open the file for reading
inputFile = open(inputFileName, 'r')

# Process each line of file.
# Here, that means to construct and draw the
# images specified on the lines of the file.
for line in inputFile:
    image = Image(imageCenter,
                  line.rstrip())
    image.draw(win)
    time.sleep(delay)
    image.undraw()

# When the user presses the mouse,
# she is done.
# Close the window and the file
win.getMouse()
win.close()
inputFile.close()
```

***This is NOT a definite loop***, assuming that reading the file is implemented as one would expect:

Opening the file sets a file pointer to the beginning of the file. Each iteration of the loop advances the file pointer to the next line of the file. The loop ends when the end of the file is reached – the number of iterations is not known (by the program) when the loop begins.

# Indefinite Loops

- Number of iterations is not known when loop starts
- Is typically a conditional loop
  - Keeps iterating as long as a certain condition remains true
  - Conditions are Boolean expressions
- Typically implemented using **while** statement
- Syntax:

```
while <condition> :
        <body>
```

Examples: **2-IndefiniteLoops.py** in today's project

Q5-6

# Tips to Debug Effectively

- Reproduce the error
- Simplify the error. Use divide-and-conquer to locate it.
- Know what your program should do
- Look at the details:
  - Read the Red in the console window!
  - Follow the Blue link in the console window!
  - Use the Debugger to track down exactly where things are going wrong
- Understand each bug before you fix it
- Practice debugging!

Use the scientific method:

- hypothesize
- experiment
- fix bug
- repeat experiment

Q7-8

# While Loop

- A *pre-test loop*
  - Condition is tested at the top of the loop
- Example use of ***while*** loops

  Nadia deposits $100 in a savings account each month. Each month the account earns 0.25% interest on the previous balance. How many months will it take her to accumulate $10,000?
- Open **3-moneyDeposit.py** in today's project

  Find and fix its bugs.

*Use the Debugger to find out why the loop does not appear to stop as expected.*

Q9

# Infinite loops on purpose

- With for loops, we could make the program run for a really long time, but not forever.
- Create a very simple while loop that runs forever.
- One answer:

```
while True:
    pass
```

# Break statement

- Useful if you want to break out of a loop in the *middle* of the loop, like this pattern:

```
while True:
    # Do some processing
    if processingSaysToStop:
        break
    # Do some more processing
```

# Break statement – Useful if you want to break out of a loop in the *middle* of the loop

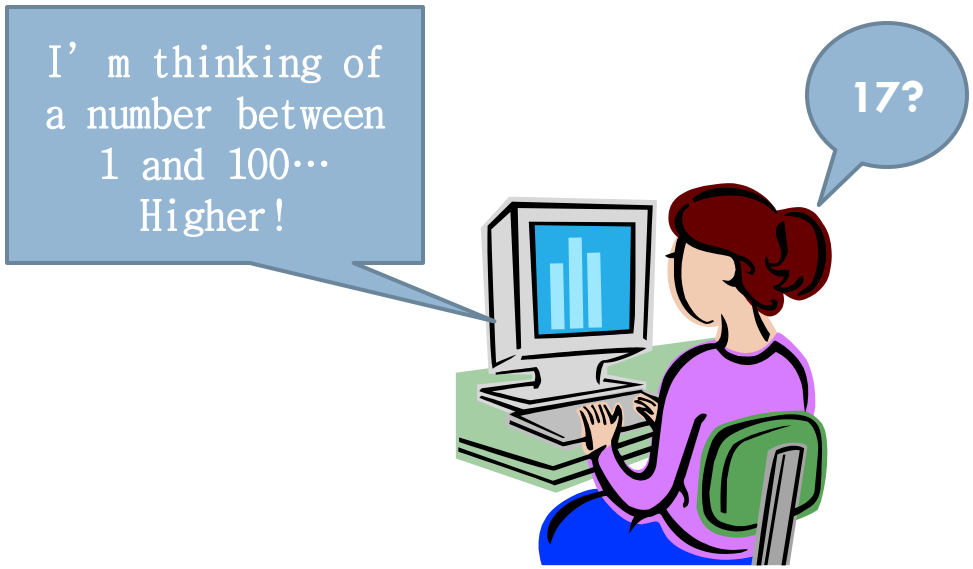
```
def breakOutOfMiddleOfLoop():
    ''' Demonstrates a reasonable use of a BREAK statement '''
    while True:
        number = int(input("Enter a number bigger than 10: "))
        if number > 10:
            break   # User entered valid input, great!

        print("You idiot!  Your number was",
              number,
              "which is NOT bigger than 10.")
        print("Try again!")

    print()
    print("OK, now that I have your number that is")
    print("bigger than 10, let's boogie!")
    print("The base 10 log of your number is",
          math.log10(number))
```

# Exercise: While Loops

- Open 5-guessMyNumber.py in today's project.
- Follow the instructions there and demo your program to your instructor or an assistant when you finish.
- Commit your work




- When you are done, please start HW11.

Q10, *turn in quiz*