Midterm 1

- Midterm is Friday Oct 16, in class.
- Bring a Scantron 2000 form
- 10-12 multiple choice questions – 65 pts?
- Short (8 lines?) program – 35 pts?
- Open book, open notes.
- Please contact me beforehand if you have to miss it; only illness counts as a reason.
- Wds Oct 21 is the drop deadline.

Programming Problem

Question 16 (Programming Problem): Write your program below and use the Scantron Sheet! Write a program that calculates an object moving by a given percentage every year until less than 1% of the original mass is left. Here is an example output (note input is underlined):

Initial mass (grams): 100
Initial yearly decay rate in percent: 10
After year 1 the mass is 90.0 grams
After year 2 the mass is 81.0 grams
After year 3 the mass is 72.9 grams
After year 4 the mass is 65.9 grams
After year 5 the mass is 59.4 grams
After year 6 the mass is 53.5 grams
After year 7 the mass is 48.2 grams
Years needed to drop below 1% of initial mass: 7

Please see comments in particular, declare inputs and outputs.

Approach

- First understand what the program is supposed to do.
- Here, the program is supposed to
  1. Get user input – initial mass, rate of decay
  2. Reduce mass by (rate of decay) percent
  3. Until mass is reduced to < 1% of its initial mass
- Heart of the program will be a while loop.
- Start in the middle… on scratch paper.

While loop

- The program will require a while loop. You really have to understand how it works.
- While Boolean is True:
  - do block
- While the Boolean test is True, the block under the while statement repeats.
- Somehow the code in the block has to eventually make the Boolean False.

Step 1: Choose variables

- initialMass - The mass the user inputs at the beginning. Float.
- mass – The mass that decreases every year. Float.
- rate – The percent decay every year. Float.
- year – The number of years so far. Integer.
Step 2: Block under the while

```python
while _________:
    mass = mass - mass*rate/100.0
    year = year + 1
    print 'After year',year,
    print 'The mass is', mass
```

- What should the Boolean condition in the while statement be?

Step 3: The Boolean condition

```python
while mass > initialMass*0.01:
    mass = mass - mass*rate/100.0
    year = year + 1
    print 'After year',year,
    print 'The mass is', mass
```

- Make sure that something that is changing in the loop will end up making the condition False eventually (and that it starts off True....)

Step 4: Getting user input

```python
reply = raw_input("Enter initial mass (grams)"):"
initialMass = float(reply)
reply = raw_input("Enter yearly decay rate in percent "):"
rate = float(reply)
```

- Midterm question does not say that input has to be checked to avoid crashes; but this is required in real life and homework!

Step 5: Beginning values for other variables

```python
mass = initialMass
year = 0
while mass > initialMass/100:
    mass = mass - mass * rate / 100
    year = year+1
    print "After year",year,
    print "the mass is",mass
```

Step 6: After the loop

```python
print "Years needed to drop below 1% of initial\nmass is",year
```

- Comments! Comments! Comments!
- If you make mistakes in the code, but the comments show what you were trying to do, you might get partial credit.

Prepare for program

- If you need more work on while loops, read page 65-87. Do the examples on a computer.
- Try to re-write this program, following the steps, without looking at the answer.
- Do the program on the sample midterm. Try it on paper, then on a computer, then look at the solution.
### Data types and operator

- \(x = 10\)
- \(\text{print } x/3\)
  - Prints 3, not 3.3333 – integer division
  - \(\text{print } "Mary" + ","\)
  - Prints "Mary," – string concatenation

### Arithmetic question

**Question 2** The line...

- \(\text{print } 16\% 5==1\)
- \(\text{(A) prints out: True}\)
- \(\text{(B) prints out: False}\)
- \(\text{(C) causes an error}\)
- \(\text{(D) prints out: 3}\)
- \(\text{(E) prints out: 1}\)

### Arithmetic question

**Question 2** The line...

- \(\text{print } 16\% 5==1\)
- \(\text{(A) prints out: True}\)
- \(\text{(B) prints out: False}\)
- \(\text{(C) causes an error}\)
- \(\text{(D) prints out: 3}\)
- \(\text{(E) prints out: 1}\)

### raw_input

- Always returns a string.
- \(\text{feet} = \text{raw_input("Enter feet: ")}\)
- \(\text{inches} = \text{feet} \times 12\)
- This is going to crash!

### raw_input()

- Programs often end with raw_input() or raw_input("Press enter to exit.")
- This does NOT exit the program. It makes the program wait for the user to type something.
- Why?

### Variables and assignment

- \(x = 0\)
- Variable on left.
- Expression (something which is or computes a data value) on the right.
- \(x = x+1\)
- Uses old value of x on the right to compute a new data value, which is stored in variable x on left.
= or == question
The lines:
\[ x = 0 \]
\[ x == 1 \]
- a) Will result in \( x \) containing the value 0.
- b) Will result in \( x \) containing the value 1.
- c) Will result in \( x \) containing the value False.
- d) Will cause an error.

= or == question
The lines:
\[ x = 0 \]
\[ x == 1 \]
- a) Will result in \( x \) containing the value 0.
- b) Will result in \( x \) containing the value 1.
- c) Will result in \( x \) containing the value False.
- d) Will cause an error.

Boolean expressions
- Either have value True or False (capitalized!)
  \( x == 0 \)
- Use it in an if or a while statement.
  \[
  \text{while } x == 0: \\
  x = x + \text{random.randint}(2)
  \]

Complicated Booleans
\[
\text{if not ((reply == 'r') or (reply == 'p') or (reply == 's')):} \\
\text{True when reply is NOT 'r', 'p', or 's'}
\]
\[
\text{if (user == 'r') and (user == 'p'):} \\
\text{Always False, so the block under if never done.}
\]
\[
\text{if (reply != 'Y') and (reply != 'N'):} \\
\text{True when reply is not 'R' or 'D'}
\]

Example if-elif-else question
The following lines:
\[
\text{import random} \\
\text{points = 0} \\
\text{x = random.randint(3)} \\
\text{if x == 0:} \\
\text{print 'rock'} \\
\text{elif x <= 1:} \\
\text{print 'paper'} \\
\text{else:} \\
\text{print 'scissors'}
\]
- a) Will print one of 'rock','paper','scissors'
- b) Might print both 'rock' and 'paper'
- c) Will never print 'rock'
- d) Might not print anything.

Example if-elif-else question
The following lines:
\[
\text{import random} \\
\text{points = 0} \\
\text{x = random.randint(3)} \\
\text{if x == 0:} \\
\text{print 'rock'} \\
\text{elif x <= 1:} \\
\text{print 'paper'} \\
\text{else:} \\
\text{print 'scissors'}
\]
- a) Will print one of 'rock','paper','scissors'
- b) Might print both 'rock' and 'paper'
- c) Will never print 'rock'
- d) Might not print anything.
random.randint function

- It's in the book and 10/12 lecture. Test it out in IDLE.
  
  random.randint(10)

- Produces a random number between 0 and 9.

Prepare for if-elif-else question

- Review lecture notes from 10/12 and rps.py program.
- Read pages 61–64 in book and try out all examples in IDLE.
- Programming exercise:
  
  - Write a program that gets a random number between 1 and 4, prints it out, and then correctly identifies it with the sentence "It is four", "It is three", "It is two", or "It is one", using if-elif-elif-else.

Common errors

- Have to think like a computer.
- Does not understand what you mean, just does exactly what you write.
- Need to be able to recognize programs that look meaningful, but are incorrect.

Question 7: What does this program print?

```python
# Program: Compare M and N
M = 1.0
N = M + 10
if N > M:
    print 'M is greater'
if M > N:
    print 'M is greater'
# The END ...

(A) M is greater
(B) N is greater
(C) both (A) and (B)
(D) nothing because M == N
(E) an error message
```

Common errors

- balance = 1000
- interest = 13.0/100
- balance + interest

What is contained in the variable balance?
Common errors

Common errors

balance = 1000
interest = 13.0/100
balance + interest
What is contained in the variable balance?

- balance is still 1000, since the value computed in the third line is not stored in the variable.
- The program does not crash, but it is probably not correct either.

How many times?

How many times?

How many times will this program print?
n = 16
while n > 1:
    n = n/2
    print n

State variable

State variable

gotAnswer = False
while not gotAnswer:
    answer = raw_input("Enter Y or N: ")
    if answer == "Y" or answer == "N":
        gotAnswer = True

- After running this, gotAnswer contains True and answer contains either "Y" or "N".

How many times?

How many times?

How many lines will this program print?
n = 16
while n > 1:
    n = n/2
    print n

- Pretend to be the program, and write out what the program will print, on scratch paper.