CSCI-UA.0002-010 – Midterm Exam #1 (v1)

October 13th, 2015

Instructor: Joseph Versoza

Ask the person to your left for their first name
(leave blank if next to empty seat or wall):

Ask the person to your right for their first name
(leave blank if next to empty seat or wall):

_____

_____

Keep this test booklet closed until the class is prompted to begin the exam

- Computers, calculators, phones, textbooks or notebooks are **not allowed** during the exam
- Please turn off your phone to avoid disrupting others during the exam

1. Answer the following questions about loops? (3 points)

   a) In Python, define **count-controlled loop**. What is the construct/control structure (keyword) that represents it?

      `A loop that repeats a specific number of times... a for loop in Python`

   b) Define **condition-controlled loop**. What is the construct/control structure (keyword) that represents it?

      `A loop that repeats as long as a condition is true... a while loop in Python`

   c) Explain why you would use one kind of loop over the other?

      `Use a for loop when you know how many iterations you want; use a while loop when the number of iterations is based on a condition (number of iterations is not known beforehand)`

2. Name 2 modules. For each module, name two functions that you can call from that module. (3 points)

   Module 1: `random`   **Module 2:** `math`

   Function 1: `randint`   **Function 1:** `floor, ceil`

   Function 2: `random`   **Function 2:** `sqrt, cos, etc.`
   `           randrange, etc. (no randstr)`

3. What is the output of the following code (no output and error are possible)? Note the number of spaces if there is left or right padding. (3 points)

   a) `print(format('hello', '.2f'))`   `Error`

   b) `print(format('hello', '<8s') + format(42, '.1f'))`   `hello    42.0     3 spaces between`

   c) `print(format(0.90, '.2%'))`   `90.00%`

4. In the **truth table** below, fill out all of the possible Boolean values for **p** and **q**, as well as the result of **p or not q**. (2 points)

p	q	p or not q
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	FALSE
FALSE	FALSE	TRUE

5. Determining what the following program will print out based on the user input specified in the 1st column of the table below. **Show your work for partial credit**. (4 points)

```
n = int(input('Gimmeh a number!\n> '))
if n == 1 or n == 2:
    print(n - 1)
elif n > 0:
    prev = 0
    cur = 1
    for i in range(0, n - 2):
        cur, prev = (prev + cur), cur
    print(cur)
else:
    print('Invalid Input')
```

User Input	Resulting Output to Screen
7	8
2	1
-2	Invalid Input

6. Write the result, `True` or `False` (or error if applicable) for the following boolean expressions and statements. (2 points)

a) `'cart' > 'carp'`   **True**   b) `False and True or Not True`   **False**

c) `10 < '20'`   **Error**   d) `False or abs(-1) > 0`   **True**

7. Read the code in the first column. Answer questions about the code in the second and third columns. (6 points total)

Code	Question #1	Question #2

```result = 0```
```for num in range(15, 4, -3):```
```   if num % 2 == 0:```
```   result += 1```
```   else:```
```   result += num```
```print(result)```
	How many times will this loop run? (1 point)
	
	4
		What is the output of this program? Show your calculations/work. (2 points)
		
		26

count = 1 **while count <= 5:** ** if count != 3:** ** print(count)** ** count += 1**	What is the output of the code on the left? (1 point) **0** **1** **2** **(then loop forever)**	Change/fix the program (you can do this directly in the code in left-most column) so that the output is the same as the following (**do not use** a for loop, and do not use multiple consecutive if statements): (2 points) 1 2 4 5

8. Write an **equivalent boolean expression** that **does not contain** the logical operator, **not**, for the condition specified in the code below. Hint: DeMorgan's Laws and / or logical opposites may help here! Show work for partial credit. (1 point)

```
if not (min_til_class_starts < 60 or hours_of_sleep > 8 ):
    print(a_message_to_you)
```

```
if min_til_class_starts >= 60 and hours_of_sleep <= 8:

    print(a_message_to_you)
```

9. Your friend is part of an avant-garde acapella group, and they've written a program to write the lyrics to their next song. The song's lyrics consists of numbers, "mmmm" and "bzzzz" (um, what? Art!). Your friend's program is supposed to:

a) **print out** numbers from **50** down to (and including) **0**, by **5**'s...
b) after each number, **add** a **random number** (1-5) of **exclamation** points
c) if the number is **greater** than or **equal** to **40**, always print out **bzzzz** (instead of the number and instead of mmmm)
d) however, for the remaining numbers, if the **number ends** in a **0**, print out **mmmm** instead of the number

Unfortunately, their program (shown below) is full of errors. It does not produce the expected output! **Circle 3 errors** (there are more than 3), **identify** if they're a syntax, runtime or logical error... and **briefly explain** why. Draw arrows or label with numbers to associate error with explanation (6 points)

Expected Output   **Broken Code (should produce output on left, but does not!)**

```
bzzzz                   for i in range(50, 0, -5): 1
bzzzz                       if i % 10 == 0: 2
bzzzz                           print("mmmm")
35!!!!!                     else if i => 40: 3, 4
mmmm                            print("bzzzz")
25!!!!!                     else:
mmmm                            num = random.randint(0, 5) 5
15!!!                           print(i + num * '!') 6
mmmm
5!
mmmm
```

Error #	Type	Explanation
1	**Logical**	**Should be range(50, -1) to include 0**
2	**Logical**	**Switch condition with i >= 40 (>= 40 takes priority for bzzzz)**
3	**Syntax**	**else if should be elif**
4	**Syntax**	**=> should be >=**
5	**Logical**	**Use randint to generate 1 to 5 (not 0 to 5)**
6	**Runtime**	**TypeError: adding non-string (i) to string**

10. Convert the following numbers . Show work for partial credit. (2 points)

a) 10000100 is **132** in decimal.   b) 15 is **00001111** in binary.

11. Circle all of the **valid variable names** (1 point): $foo   ==Foo==   ==_foo==   2_foo   ==foo2==

12. Name two data **types** in Python that **are not** *numeric*, and give a **syntactically correct** literal example of each. (2 points)

- type: **str, bool**   example: **'hello', True**

- type: **range**   example: **range(5)**

13. Write a program that computes that **asks** the user for **two numbers**. It will **compute** the **greatest common factor** that divides evenly into both numbers. (5 points)

a) Ask for the first number ('Enter num ...')
b) **No validation is necessary** – you can assume that the numbers coming in are whole numbers greater than 0
c) Print out the largest number that divides evenly into both numbers entered ('The gcf that divides both <num 1> and <num 2> is <greatest common factor>')
d) Hint: one strategy might be to try all possible numbers (what are the boundaries... does it matter which factor is smaller?) to see which ones are divisors
e) Hint: what operator would you use to determine if a number is a divisor (divides evenly)?
f) Hint: your algorithm can be entirely inefficient (that is, you can try factors that you know won't work!)
g) Example output below:

Run 1:
```
Enter num 1
> 17
Enter num 2
> 4
The gcf that divides both 17 and 4 is 1
```

Run 2:
```
Enter num 1
> 30
Enter num 2
> 36
The gcf that divides both 30 and 36 is 6
```

```
x = int(input('Enter num 1\n> '))
y = int(input('Enter num 2\n> '))
gcf = 1
for factor in range(1, x + 1):
    if x % factor == 0 and y % factor == 0:
        gcf = factor
print('The gcf that divides both', x, 'and', y, 'is', gcf)

# or count down and break
# optionally use smaller number to set limits on range
```

14. Write a tiny betting game (wait a second, is this even legal!?). The player will **choose** either **(L)ower** or **(H)igher**, and place a bet. The computer will **generate** a **random number** between **1 and 7** inclusive. If the resulting **number matches** the player's **choice relative** to **4** (that is, lower or higher than 4), you keep your bet, and you **win the same amount you bet**. However, if it's the **opposite**, you **lose** the **amount** you **bet**. Finally, if it's a **tie**, you **don't lose anything**. Do this until you have no longer have any money to bet (you must have at least 1 dollar to bet)... or until you've doubled your money (you start out with $100). (9 points)

a) Start the player with **$100**
b) Ask the player to choose L or H for lower or higher: `'(L)ower or (H)igher than 4?'`
c) If the player doesn't type in either L or H, **default to L**
d) Ask the player how much they'd like to bet
e) If they don't enter a positive number, **default to 1**
f) Generate a random number between 1 and 7 (inclusive)
g) Print out `'you won'` , and add the player's bet to their total (including their original bet) if they guess correctly
h) If the opposite occurs, print out `'you lost!'`, and subtract the player's bet from their money
i) If it's a tie, no money is added or deducted from the player
j) Once a winner is determined, print out the current total: `'Total: <total>'`
k) **All dollar amounts should have two decimal places and a dollar sign**: `$2.00`
l) Repeat again, starting with step b ... until the player's money is less than 1 or greater than or equal to $200
m) Print out `'Game over.'`, and the player's total once the game is finished (negative amounts are ok)
n) Example output below:

```
(L)ower or (H)igher than 4?           <continued from left collumn>
> L                                    (L)ower or (H)igher than 4?
How much are you betting?              > L
> 50                                   How much are you betting?
1                                      > 105
you won!                               2
Total  $150.00                         you won!
(L)ower or (H)igher than 4?            Total  $202.00
> L                                    Game over. You have  $202.00
How much are you betting?
> 53
7
you lost!
Total  $97.00
```

```python
import random
money = 100
while money > 0 and money < 200:
    choice = input('(L)ower or (H)igher than 4?\n> ')
    bet = int(input('How much are you betting?\n> '))
    if choice != 'L' and choice != 'H':
        choice = 'L'
    roll = random.randint(1, 7)
    print(roll)
    if roll == 4:
        print('no one won!')
    elif choice == 'L' and roll < 4 or choice == 'H' and roll > 4:
        print('you won!')
        money += bet
    elif choice == 'L' and roll > 4 or choice == 'H' and roll < 4:
        print('you lost!')
        money -= bet
    print('Total ', '$' + format(money, '.2f'))
print('Game over. You have ', '$' + format(money, '.2f'))
```

15. You're a mad scientist, and one of your hobbies is creating mutants. To assist in your creation of magnificent mutant minions, you write a program to generate DNA sequences. DNA is made up of triplets (codons) of nucleotides: guanine, adenine, cytosine and thymine (G, A, C and T). Each triplet, or codon, is made up of 3 nucleotides (nucleic acids). Generate a random sequence of G, A, C, and T based on a number of codons specified by the user. (6 points)

Continually ask the user for the number of codons if they do not specify a number greater than 0:

```
How many nucleotide triplets (codons)?
> -1
Please enter a number greater than 0...
How many nucleotide triplets (codons)?
> 7
Your DNA is ready! ATCAATGGAATCGTGATATAC
```

```
0.5: import random
0.5: ask for input, convert to int
1:   continually ask for valid input
1:   sting accumulator
0.5: generate random number
2:   appropriate if/elif, along with addition
0.5: print
```

```python
import random
triplets = int(input('How many nucleotide triplets (codons)?\n> '))
while triplets < 0:
    triplets = int(input('Please enter a positive number\n> '))

dna = ''
for i in range(triplets * 3):
    gact = random.randint(1, 4)
    if gact == 1:
        dna += 'G'
    elif gact == 2:
        dna += 'A'
    elif gact == 3:
        dna += 'C'
    elif gact == 4:
        dna += 'T'
print('Your DNA is ready! ' + dna)
```