Codeitz Assessment/ Post-Test

Benjamin Xie
bxie@uw.edu

Supplementary Material

Xie, Benjamin, Greg L. Nelson, Harshitha Akkaraju, William Kwok, and Amy J. Ko. “The Effect of Informing Agency in Self-Directed Online Learning Environments.” In *Proceedings of the Seventh (2020) ACM Conference on Learning @ Scale*. L@S 2020. ACM, 2020.

Grading guidelines

- I ignore minor typo mistakes that don't impair understanding of what learner was trying to do (e.g. upper vs lowercase, missing punctuation)
  - => printing literal to an exactness is not important knowledge
- Not too strict on quotes around strings or not

Write down all values printed as output after this code runs.

```
x = 2
y = 5
z = 3

if (y % x == 1):
    print("a")
    x = x * x;
elif (y % x == 2):
    print("b")
    z = z * z
else:
    print("c")
    y = y * y
if (y / x == 1):
    print("g")
    x = x + 3
else:
    y = y * 2
    print("h")

print(x)
print(y)
print(z)
```

Answer

```
a
h
4
10
3
```

Scoring: 4

- 1 pt for first 2 lines (-0.5 for each incorrect, additional, or missing line)
- 3 pts for last 3 lines
  - -1 pt if var name included w/ correct number (e.g. `x = 2`)
  - -1 total if lines begin with "x=", "y=", "z="
  - -0.5 for each additional line
- -1 total if no new lines
- -0 if quotes around strings
  - -1 total if quotes around numbers

Justification

This exercise assesses knowledge of variable updates, conditional

What is printed as a result of this code segment?

```
name = "james"
time = "night"

print("hi")
if(time != "day" and name == "Alice"):
    print("hi alice")
elif(time != "day"):
    print("hello")
    print("name")
else:
    print("good day to you")
print("done")
```

Answer

```
hi
hello
name
done
```

Scoring: 2.5

- 0.5 pt for each line except 3rd (1.5 total)
- 1 if "name" line missing, 0.5 if "james"
- -0.25 lines 1 and 2 merged ("good night name"), -0.5 if 1 and 2 merged with var name ("good night james")
- -1 (total) if lines have additional info on them (e.g. `output = "hello"`)

Notes

- Expect 3rd line to be common error (variable vs literal).
- No points off if new lines missing?
- Never actually asks to print var value
-

3. For the next three questions (3A-3C), consider the following code. The code below assumes that the variables `a`, `b`, and `c` all store numbers (integers or floats).

```
x = -1
y = -1

if(a < b and a < c):
    print(1)
    x = a
elif(b < c):
    print(2)
    x = b
else:
    print(3)
    x = c

if(a > b and a > c):
    print(4)
    y = a
elif(b > c):
    print(5)
    y = b
else:
    print(6)
    y = c

val = y - x

if(val > 0):
    print("THE VALUE:")
    print(val)
```

Given the variable values `a = 1.1, b = 5, c = 2`, determine the output of the code and write the output below:

```
1
5
THE VALUE:
3.9
```

Scoring: 3

- -0.5 pt each for first 3 lines (-0.25 for each additional line, e.g. var update)
- 1.5 pts for last line correct (-0.5 if "THE VALUE:" and "3.9" on same line, so 1.5 total for "THE VALUE: 3.9")
- -0.5 for each line > 4

In the box below, summarize in plain English what the code does.

```
Finds difference between max and min of 3 numbers.
```

Scoring: 3

- 2 pt for mentioning max/min (1 pt for max, 1 pt for min)
- 1 pt for mentioning finding *difference* between max and min
- -50% if describing code line by line (e.g. mentioning every variable specifically)

If 'OUTPUT:' was <u>not</u> printed, what is the relationship between the variables `a`, `b`, and `c`?

```
Output not printed when a, b, c all <= 0 or all equal to each other.
```

Scoring: 3

- 1.5 pts for saying output not printed when a, b, c all <= 0 (-0.25 if they say < 0)
- 1.5 pts for saying output not printed when a, b, and c all same (a==b, b==c)
- 1 pt if only mention when conditional false

The code below assumes that the variables `a`, `b`, and `c` all store integers.

```
x = a%2==0
y = b%2==0
z = c%2==0

u = 0

if(x):
  u = u + 1
if(y):
  u = u + 1
if(z):
  u = u + 1

print(u)
```

Given the variable values `a = -2, b = 3, c = 4`, determine the output of the code and write the output in the box below:

```
2
```

Scoring: 2

- 2 pts for right answer (-1 pt if write "u" or "u =")
- -0.5 for each additional line (0 pts total if last line does not have "2" in it)

In the box below, summarize in plain English what the code does.

```
Prints the number of even numbers stored in variables
```

Scoring: 3

- 2 points if mentions "even" (or "divisible by 2")
  - -1 if say "number of variables where %2 == 0"
  - -1.5 if just mentioning conditional ("if statement is true")
- 1 pt for mentioning "counting" or "how many"
  - 0.5 pt for mentioning updating `u`
- Only 1 if only mentions conditionals or boolean ("how many statements are true")
- -50% if describing code line by line (e.g. mentioning every variable specifically)

What would variables `a`, `b`, and `c` have to be for `0` to be printed?

```
a,b, and c would have to all be odd numbers.
```

Scoring: 2

- -0.5 if don't mention even/odd or divisible by 2 and instead say "when a,b, and c %2 does not equal 0"
- -1-1.5 if says only subset of variables must be odd
- -1 if additional constraint added (e.g. a, b, and c must be absolute value, positive)
- -1 if think it must be even number or when a, b, and c %2 !=0
- -1.5 if only mentions when `u` equals 0
- -1.5 if only mentions if statements (e.g. "when all if statements invalid")
- -1.5 if give specific valid example (e.g. a = 1, b = 3, c = 5)

Two friends regularly play chess against each other and they want to keep track of who was the last person to the win and how many previous games in a row they won. To do so, they ask you write some code to help them.

Predefined Variables

Four variables have already been defined:

- The variable `leader` has the name of the person who won the previous game(s).
- The variable `follower` contains the name of the person who lost the previous game.
- The variable `current_streak` contains the number of consecutive games that have been won by `leader`.
- The variable `winner` contains the name of the person who just won a game.

Code Instructions

They ask you to write code to do the following:

1. If `winner` is equal to `follower`, then there is a new champion.
    a. Swap the names stored in `leader` and `follower` to reflect this change.
    b. Reset `current_streak` to 0.
    c. Print "`new leader`"
2. If `winner` is equal to `leader`, then the person who won the previous game has won another one
    a. Update `current_streak` by adding `1` to the previous value.
    b. Print "`same leader`"
3. If `winner` is not equal to `follower` or `leader`, then there is an unknown player.
    a. Print "`unknown player`"

Example Execution

Here are a few examples of what how the code would execute:

- If the variable `winner` was set to "`Luca`" and the variable `follower` was also set to "`Luca`", the values stored in `leader` and `follower` would swap, `current_streak` would be set to `0`, and "`new leader`" would be printed.
- If the variable `winner` was set to "`Abby`", the variable `leader` was also set to "`Abby`", and the variable `current_streak` were set to `4`, then `current_streak` would be updated to `5` and "`same leader`" would be printed.
- If the variable `winner` was set to "`Kim`", the variable `leader` was set to "`Juan`", and the variable `follower` were set to `"Olaf"`, then "`unknown player`" would be printed.

Solution:

```
if winner==follower:
  follower = leader
  leader = winner
  current_streak = 0
  print("new leader")

elif winner == leader:
  current_streak = current_streak + 1
  print("same leader")

else:
  print("unknown player")
```

Scoring: 4

- 2.5 for if condition (-2 if swap wrong)
    - -0.5 if updates winner w/ correct swap
    - -1 if swap uses temp var but still wrong
    - -1.5 if swap w/o 3rd var
- 1 for elif
    - -0.5 if current_streak not updated correctly
- 0.5 for else
    - -0.25 if condition added to it (ok if elif with condition; not grading condition for logical correctness)

- -0 if uses 3 if statements (technically incorrect code, but instructions were unclear)
- -0.25-0.5 for minor syntax errors (logic is correct, code may need small adjustment to run correctly).
    - E.g. single = for equality check (-0.5 if done everywhere, -0.25 if only done once)
- -0.5-1 for major syntax errors: unclear what code is trying to do, major refactor for code to work
    - E.g. multiple wrong variable names used, quotes around var names (-1 if all vars)

Notes

- "Winner" and "leader" being different vars is confusing
- Logic error where if use multiple if statements (if 1st condition true, 2nd condition also true b/c of var update). That's ok b/c question miswritten.

Focus of question around swap being used correctly as well as conditionals used effectively

~~You and a few friends go out to eat at a restaurant and decide to split the bill and pay in bitcoin cryptocurrency. The meal costs each of you a very small fraction of a single bitcoin. You want to write code 1) determine how much each person owes and 2) ensure that you all have paid the bill off.~~

~~First, you want to know how much each of you owe. The total cost of the meal is stored in the variable `cost`. The total number of people eating is stored as a number in the variable `num_people`.~~

~~**Given the variables `cost` and `num_people`, write code that divides `cost` by `num_people` and stores the result in a new variable `cost_per_person`. Then print the output of the variable `cost_per_person`.**~~

~~Solution~~

~~cost_per_person = cost / num_people~~
~~print(cost_per_person)~~

~~Scoring: 1.5~~

- ~~-1 for 1st line of code incorrect~~
- ~~-0.5 if 2nd line of code incorrect~~

Say you and 2 friends (a total of 3 people) split a bill. The amounts each of you paid are decimal numbers stored in the variables `amt1`, `amt2`, and `amt3`. You want to determine if you paid within `0.000001` (`1e-6`) bitcoin of the bill. The cost of the meal is stored in the decimal variable `cost`. You are worried that you may have underpaid or overpaid.

Write code that determines if you and your friends properly paid for the bill.

- If in total you all paid at least `0.000001` less than the cost, your code should print "underpaid" and then the amount that you underpaid on the next line.
- If in total you all paid within `0.000001` of the cost, your code should print "paid in full".
- If in total you all paid at least 0.000001 more than the cost, your code should print "overpaid" and then the amount you all overpaid on the following line.

In example, say
`amt1 = 0.001111`, and
`amt2 = 0.002222`, and
`amt3 = 0.000033`, and
`cost = 0.003368.`

The output of the code would be:

```
underpaid
0.000002
```

Solution

```
paid = amt1 + amt2 + amt3
thres = 0.0001

diff = paid - cost

if diff < 0 and abs(diff) > thres:
    print("underpaid")
    print(abs(diff))
elif abs(diff) < thres:
    print("paid in full")
else:
    print("overpaid")
    print(abs(diff))
```

Scoring: 6

- 1 point for total paid
- 1 point for difference between cost and sum of amounts paid
- 1.5 point for 3 conditionals
    - 0.5 for having 3 conditions
    - 1 pt for having conditional statements relating to float equality
- 2 point for float equality check in conditionals w/ threshold, abs value.
    - 1 pt for correct math operation
    - 0.5 pt for threshold value
    - 0.5 for abs value function (or equivalent behavior)
    - -1 for each incorrect w/ major error (logic, major syntax).
    - -0.5 for each w/ minor syntax error
- 0.5 point for correct print statements (-0.25 if values not printed)

Write code that determines if the variable `inp`, a 4 digit integer value (between `1000`-`9999`), is a valid passcode. `inp` is a valid passcode if the sum of the first 3 digits modulus `7` is equal to the last digit. If `inp` is valid, the code should print '`valid'`. If the string is not valid, it should print '`NOT valid'`.

So if `inp` were set to `5312`, it would be a valid passcode and your code would print `valid` because the first 3 digits (`5`, `3`, and `1`) sum to `9` and `9` modulus `7` equals the last digit (`2`). `1234` would ***not*** be a valid passcode and your code would print `NOT valid`. Write your solution in the box below.

Assume a variable `inp` has already been declared and stores a 4 digit integer value (between `1000`-`9999`).

Solution

```
digit_4 = inp % 10
inp = inp // 10
digit_3 = inp % 10
inp = inp // 10
digit_2 = inp % 10
inp = inp // 10
digit_1 = inp %10

sum_3 = digit_1 + digit_2 + digit_3

if(sum_3 % 7 == digit_4):
    print("valid")
else:
    print("NOT valid")
```

Scoring: 7

- 3 pts for digit processing
  - -0.5 to -1.5 if inp not truncated properly
  - -0.5 to -1.5 if digits not saved properly
- 1 pt for using // instead of /
  - Ok if use `int()` (even though we didn't teach that…)
- 1 point for summing digits properly
- 1.5 point for conditional % 7 (-0.5 if 2 ifs used)
  - -1 if no %7
  - -0.5 if not comparing to 4th digit
- 0.5 point for writing correct print statements