

Quiz 1 Solutions

Name: _____

Lab room: _____

Directions

Congrats on completing Week 1!

- **This quiz contains 14 questions and will last 1 hour.** There is also one optional bonus problem at the end you can try if you finish all the other questions early.
- No computers. Trace through the code by hand.
- The questions are not in order of difficulty. Use your time wisely. If you are having too much trouble on a question, skip it and return to it later. **Avoid getting stuck.**
- If a loop never stops, write “Infinite”.
- For questions with *circular bubbles*, you should select *exactly one* choice.
 - ☐ You must choose either this option
 - ☐ Or this one, but not both!
- For questions with *square checkboxes*, you may select *multiple* choices.
 - ☐ You could select this choice.
 - ☐ You could select this one too!

1. [warm-up] Write the value of each expression.

- a) `10 / 2` `5.0`
- b) `17 // 5` `3`
- c) `17 % 5` `2`
- d) `"2" == 2` `False`
- e) `(3 > 1) and (2 > 5)` `False`

Write the type of each expression.

- f) `type([1, 2, 3])` `list`
- g) `type(4 + 0.5)` `float`

2. [find the bugs] Each program below may have no errors or some errors. Circle **every** mistake, and write down the total number of errors you found.

- a)
- ```
Check if someone is an adult
age = 16
if age = 18:
 print(Adult)
print("Done")
```

Number of errors you found in the program above: \_\_\_\_\_

b)

```
Print even numbers from a list
nums = [1, 2, 3]
for n in nums:
 if n % 2 == 0:
 print(n)
```

Number of errors you found in the code above: \_\_\_\_\_

c)

```
Print every name using indices
names = ["Jelani", "Timnit", "Rediet"]
for i in range(names):
 print(names[i])
```

Number of errors you found in the code above: \_\_\_\_\_

3. [strings] Consider the following string.

```
city = "Gondar"
```

Optional: You may fill out the indexes below to help you answer the questions below, but it is **not necessary**.

| character | G | o | n | d | a | r |
|-----------|---|---|---|---|---|---|
| index     |   |   |   |   |   |   |

Write out the values for each expression below relating to the string variable `city = "Gondar"`. You may write your answers with or without quotations.

a)

```
city[1:4]
```

```
"ond"
```

b)

```
city[-1]
```

```
"r"
```

c)

```
len(city)
```

```
6
```

d)

```
city[3:]
```

```
"dar"
```

e)

```
city[:3]
```

```
"Gon"
```

4. **[lists]** Write the output of the following code.

```
prices = [30, 12, 45, 12]
prices.append(20)
prices[3] = 10
print(prices)
```

```
[30, 12, 45, 10, 20]
```

5. **[conditionals]** What does the following code print?

a)

```
score = 72
if score >= 90:
 print("Excellent")
elif score >= 70:
 print("Good")
elif score >= 50:
 print("Pass")
else:
 print("Try again")
```

```
Good
```

b)

```
num_seats = 12

if num_seats % 3 == 0:
 print("Rows of 3")

if num_seats % 4 == 0:
 print("Rows of 4")
```

```
Rows of 3
Rows of 4
```

c)

```
x = 12
if x > 10:
 print("A")
if x < 5:
 print("B")
else:
 print("C")
```

```
A
C
```

d)

```

x = 20
if x >= 18:
 print("A")
 if x <= 65:
 print("B")
else:
 print("C")

```

A  
B

6. [trace] Trace the code below line by line. Then write the final printed output. The first pass through the loop is already done for you as an example.

```

1 total = 0
2 count = 1
3 while count <= 4:
4 total = total + count
5 count = count * 2
6 print(total)
7 print(count)

```

*Optional:* You may use this table to help you trace the code, but it is **not necessary**.

| Loop pass         | Is count <= 4? | total after line 5 | count after line 5 |
|-------------------|----------------|--------------------|--------------------|
| (before the loop) | —              | 0                  | 1                  |
| 1st               | True           | 1                  | 2                  |
| 2nd               |                |                    |                    |
| 3rd               |                |                    |                    |
| 4th               |                |                    |                    |
| 5th               |                |                    |                    |
| 6th               |                |                    |                    |

What is the final printed output?

7  
8

**7. [for vs while]** Answer the questions below.

- a) The for loop below prints the squares 1, 4, 9, 16, 25. Fill in the three blanks so the while loop prints exactly the same thing.

```
Version A --- for loop

for i in range(1, 6):
 print(i * i)
```

```
Version B --- while loop

i = 1
while i < 6:
 print(i * i)
 i = i + 1
```

- b) What happens when you run the following program? Select exactly one.

```
num = 1
while num < 10:
 print(num)
```

- ☐ It prints 1 once, then stops.
- ☒ It prints 1 forever, an infinite loop.
- ☐ It prints nothing.
- ☐ Python gives an error before the loop starts.

**8. [print vs return]** Answer the following questions.

a) What is the printed output?

```
def double_print(n):
 print(n * 2)

def double_return(n):
 return n * 2

a = double_print(5)
b = double_return(5)

print(a)
print(b)
```

```
10
None
10
```

b) A shop wants to write a function to apply a 15% tax, but it's incorrect:

```
1 def add_tax(price):
2 print(price * 1.15)
3 total = add_tax(100)
4 print(total)
```

We need to change line 2 in `add_tax` so that `total` correctly becomes 115.0. Write the fixed line.

```
1 def add_tax(price):
2 *@\solfill{return price * 1.15}@*
3 total = add_tax(100)
4 print(total)
```

9. [scope] What does the following code print?

```
x = 10

def do_something():
 x = 20
 print(x)

do_something()
print(x)
```

```
20
10
```

10. [break and continue] What does the following code print?

```
i = 0
while i < 11:
 i = i + 1
 if i % 2 == 0:
 continue
 print(i)
 if i == 5:
 break
```

```
1
3
5
```

11. [list aliasing] Answer the questions below.

a) What does the following code print?

```
a = [1, 2, 3]
b = a
b[0] = 9
print(a)
print(b)
```

```
[9, 2, 3]
[9, 2, 3]
```



b) What does the following code print?

```
a = [1, 2, 3]
b = a[1:]
b[0] = 9
print(a)
print(b)
```

```
[1, 2, 3]
[9, 3]
```

**12. [style / indentation]** The code below is correct, but it has lost all indentation. Rewrite the code block below with the proper indentation. Make sure your indentation can be understood visually.

```
ORIGINAL (UNINDENTED) CODE
Return how often target appears in items.
Examples:
count_target([1, 2, 3], 3) --> 1
count_target(["a", "b", "a"], "a") --> 2
count_target(["a", "b", "a"], "c") --> 0

def count_target(items, target):
count = 0
for item in items:
if item == target:
count += 1
return count
```

Re-write the code above with correct indentation below. You must use clear and correct indentation so it's easy to read.

```
def count_target(items, target):
 count = 0
 for item in items:
 if item == target:
 count += 1
 return count
```

**13. [write a function]** Write a function `absolute(n)` that takes a number and returns its absolute value (the number itself if it is positive or zero, and its opposite if it is negative).

Examples:

- `absolute(5) → 5`
- `absolute(-3) → 3`
- `absolute(0) → 0`

Your function must **return** the number (not print it). Do not use the built-in `abs` function.

```
def absolute(n):
 if n < 0:
 return -n
 return n
```

**14. [write a function]** Write a function `sum_list(nums)` that takes a list of numbers and returns the sum of all of them.

Examples:

- `sum_list([1, 2, 3, 4]) → 10`
- `sum_list([10, -2]) → 8`
- `sum_list([]) → 0`

Your function must **return** the number (not print it). Do **not** use the built-in `sum` function.

```
def sum_list(nums):
 total = 0
 for num in nums:
 total = total + num
 return total
```

**15. [BONUS — nested loops]** *Optional bonus problem:* only attempt this if you finish the rest of the quiz and have spare time.

A *table* of numbers can be stored as a list of lists. Each inner list is one row. For example:

```
table = [
 [3, 7, 2],
 [9, 1, 4],
 [5, 8, 6]
]
```

a) What does the following code print?

```
table = [
 [1, 4],
 [7, 2]
]
total = 0
for row in table:
 for num in row:
 total = total + num
print(total)
```

14

b) Write a function `table_max(table)` that uses **nested loops** (loops inside loops) to return the largest number in a table (a list of lists of numbers). Do **not** use the built-in `max` function.

Examples:

- `table_max([[3, 7, 2], [9, 1, 4], [5, 8, 6]]) → 9`

- `table_max([[2, 2], [2, 2]]) → 2`

Your function must **return** the number (not print it).

```
def table_max(table):
 biggest = table[0][0]
 for row in table:
 for num in row:
 if num > biggest:
 biggest = num
 return biggest
```