These are the multiple choice questions for Question 5. You must use the bubble sheet at the end of the midterm to answer these questions.

1. Suppose we begin with the statement `correct = False`. What is the result of evaluating the expression `correct and 1 / 0 == 1`

   (a) True (b) False (c) Error

2. Suppose we begin with the statement `name = 'Anya Tafliovich'`. What is the result of evaluating the expression `name.lower().find('a').find('a')`

   (a) 0 (b) 3 (c) 6 (d) Error

3. Suppose we begin with the statement `name = 'Kaveh Mahdaviani'`. What is the result of evaluating the expression `name.count('a', 1, -3)`

   (a) 1 (b) 2 (c) 3 (d) Error

4. Which of the following boolean expressions evaluate(s) to `True`?

   (a) `-4 == -4 // 1` (b) `2.0 == 2.0 // 1`
   (c) `1.2 == 1.2 // 1` (d) both (a) and (b)

5. Suppose we begin with the statement `large = True`. What is the result of evaluating the expression `42 + '2' == 44 or large`

   (a) True (b) False (c) Error

6. Suppose we begin with the statement `phrase = 'You ARe AlmOsT TheRE'`. What is the result of evaluating the expression `phrase.swapcase() + ', ' + phrase[8:phrase.find(' The')]`.

   (a) 'yOU arE aLMoSt tHEre, aLMoSt tHEre' (b) 'yOU arE aLMoSt tHEre, AlmOsT TheRE'
   (c) 'yOU arE aLMoSt tHEre, aLMoSt' (d) 'yOU arE aLMoSt tHEre, AlmOsT'

7. A not-so-good Python programmer has written the following function:

```
def sign_function(num: int) -> str:
    if num > 0:
        return 'Positive'
    elif num < 0:
        return 'Negative'
```

   What is the result of evaluating the expression `str(sign_function(0)) in 'Positive'`

   (a) True (b) False (c) Error

8. Consider another piece of code from that same not-so-good Python programmer:

```
def my_function(num: int) -> str:
    if num % 2 == 0:
        result = 'Even'
    else:
        result = 'Odd'

my_function(0)
```

Suppose we run this code. Then what is the result of evaluating the expression `type(result)`

(a) None (b) <class 'str'> (c) <class 'NoneType'> (d) Error

9. Consider the following code:

```
x = 1
y = 2
count = 0
if x > 0:
    count = count + 1
elif x + y < 4:
    count = count + 2
if y - x >= 1:
    count = count + 3
else:
    count = count * 2
```

After running this code, what is the result of evaluating the variable `count`

(a) 1 (b) 2 (c) 4 (d) 6

10. Suppose we begin with the statement
`last_word = 'If you have time, double check your answers before you leave! :)'`.
What is the result of evaluating the expression `last_word[-3:][::-1][:-1]`:

(a) ':)' (b) '):' (c) ':(' (d) '(:'

Short Python help descriptions:

```
Operator precedence (from highest to lowest):

  ()              Parentheses
  **              Exponent
  +x, -x          Unary plus, Unary minus
  *, /, //, %     Multiplication, Division, Floor division, Modulus
  +, -            Addition, Subtraction
  ==, !=, >, >=, <, <=, in, not in    Comparisons, Membership operators
  not             Logical NOT
  and             Logical AND
  or              Logical OR

Built-in functions:

  int(x: object) -> int
    Convert x to an integer, if possible. A floating point argument will be
    truncated towards zero.
  len(x: object) -> int
    Return the length of list, tuple, or string x.
  min(iterable: object) -> object
  min(a, b, c, ...) -> object
      With a single iterable argument, return its smallest item.
      With two or more arguments, return the smallest argument.
  print(values: object) -> None
    Prints the values.
  str(x: object) -> str
    Return an object converted to its string representation, if possible.
  type(x: object) -> the object's type
    Return the type of the object x.

str:

  x in s --> bool
    Produce True if and only if string x is in string s.
  str(x: object) -> str
    Convert an object into its string representation, if possible.
  str.count(sub: str[, start: int[, end: int]]) -> int
    Return the number of non-overlapping occurrences of substring sub in
    the string slice with endpoints start and end.  Optional arguments
    start and end are interpreted as in slice notation.
  str.find(sub: str[, i: int]) -> int
    Return the lowest index in the string (starting at index i, if i is
    given) where the string sub is found or -1 if sub does not occur in the string.
  str.index(sub: str) -> int
    Like find but raises an exception if sub does not occur in the string.
  str.isalnum() -> bool
```

```
  Return True if and only if all characters in the string are alphanumeric
  and there is at least one character in the string.
str.isalpha() -> bool
  Return True if and only if all characters in the string are alphabetic
  and there is at least one character in the string.
str.isdigit() -> bool
  Return True if and only if all characters in the string are digits
  and there is at least one character in the string.
str.islower() -> bool
  Return True if and only if all cased characters in the string are lowercase
  and there is at least one cased character in the string.
str.isupper() -> bool
  Return True if and only if all cased characters in the string are uppercase
  and there is at least one cased character in the string.
str.lower() -> str
  Return a copy of the string converted to lowercase.
str.lstrip([chars: str]) -> str
  Return a copy of the string with leading whitespace removed.
  If chars is given and not None, remove characters in chars instead.
str.replace(old: str, new: str) -> str
  Return a copy of the string  with all occurrences of the string old replaced
  with the string new.
str.rstrip([chars: str]) -> str
  Return a copy of the string with trailing whitespace removed.
  If chars is given and not None, remove characters in chars instead.
str.strip([chars: str]) -> str
  Return a copy of the string with leading and trailing whitespace removed.
  If chars is given and not None, remove characters in chars instead.
str.swapcase() -> str
  Return a copy of the string with uppercase characters converted to lowercase
  and vice versa.
str.upper() -> str
  Return a copy of the string converted to uppercase.
```