WWPD: Control

What would Python display?
Assume the following code has been executed.

```
def mystery(a, b, c, d):
    if a < 0:
        return None
    while b < c:
        c = c - 1
        d = d + 3
    return c > d

print(mystery(1, 2, 3, 4)) # (a)
print(mystery(1, 2, 3, -4)) # (b) & (c)
print(mystery(1, -2, -3, 4))  # (d)
print(mystery(-1, -2, 3, -1))   # (e)
```

1. (1.5 pt) What value is printed at **(a)**?
   - A. True
   - B. [✓ False]
   - C. None

2. (1.5 pt) What value is printed at **(b)**?
   - A. [✓ True]
   - B. False
   - C. None

3. (1.5 pt) What value is bound to `c` in the local frame at the *end* of the second function call to mystery at **(c)**?
   - A. 0
   - B. 1
   - C. [✓ 2]
   - D. 3
   - E. -4

4. (1.5 pt) What value is printed at **(d)**?
   - A. True
   - B. [✓ False]
   - C. None

5. (1.0 pt) What value is printed at **(e)**?
   - A. True
   - B. False
   - C. [✓ None]

Function Equivalence: Exponentiation

Definition. Two functions f and g have identical behavior if f(x) and g(x) return equal values or return functions with identical behavior, for every x that does not cause an error.

```
from operator import mul

def double(x):
    return x * 2

def triple(x):
    return x * 3

def enigma(y):
    # Hint: look at the parentheses very carefully
    return double(triple(y)) * triple(triple(y))

def multiply_by(a):
    def slow_multiplication(b):
        sum = 0
        x = 0
        while (x < a):
            sum = b + sum
            x = x + 1
        return sum
    return slow_multiplication
```

NOTE: the function `mul(x, y)` computes `x*y`.

6. (2.0 pt) The result of evaluating `` `multiply_by(2)(5)` `` has identical behavior to the result of evaluating the expression … (**check all that apply**)

   A. `enigma(6)`
   B. `enigma(2)`
   C. `enigma(5)`
   D. `double(5)`
   E. `double(2)`
   F. `triple`
   G. `triple(5)`
   H. `mul(5, 2)`
   I. `mul(2, 5)`
   J. `mul(3, 4)`
   K. `mul`

7. (2.0 pt) The result of evaluating `multiply_by(3)(5)` has identical behavior to the result of evaluating the expression … (**check all that apply**)

A. enigma(1)

B. enigma(2)

C. enigma(3)

D. engima

E. [✓ triple(5)]

F. triple(3)

G. triple

H. [✓ mul(3, 5)]

I. [✓ mul(15, 1)]

J. mul(2, 10)

K. mul

8. (2.0 pt) The result of evaluating `multiply_by(2)` has identical behavior to the result of evaluating the expression … (**check all that apply**)

A. enigma(10)

B. enigma

C. double(2)

D. double(5)

E. [✓ double]

F. mul(2, 5)

G. mul(1, 10)

H. mul

9. (2.0 pt) What is the type of the return value for the function `multiply_by`?

A. [✓ Function]

B. Integer

C. Float

D. String

E. None

10. (2.0 pt) What is/are the type of the return value(s) for the function `enigma`? (**check all that apply**)

A. Function

B. [✓ Integer]

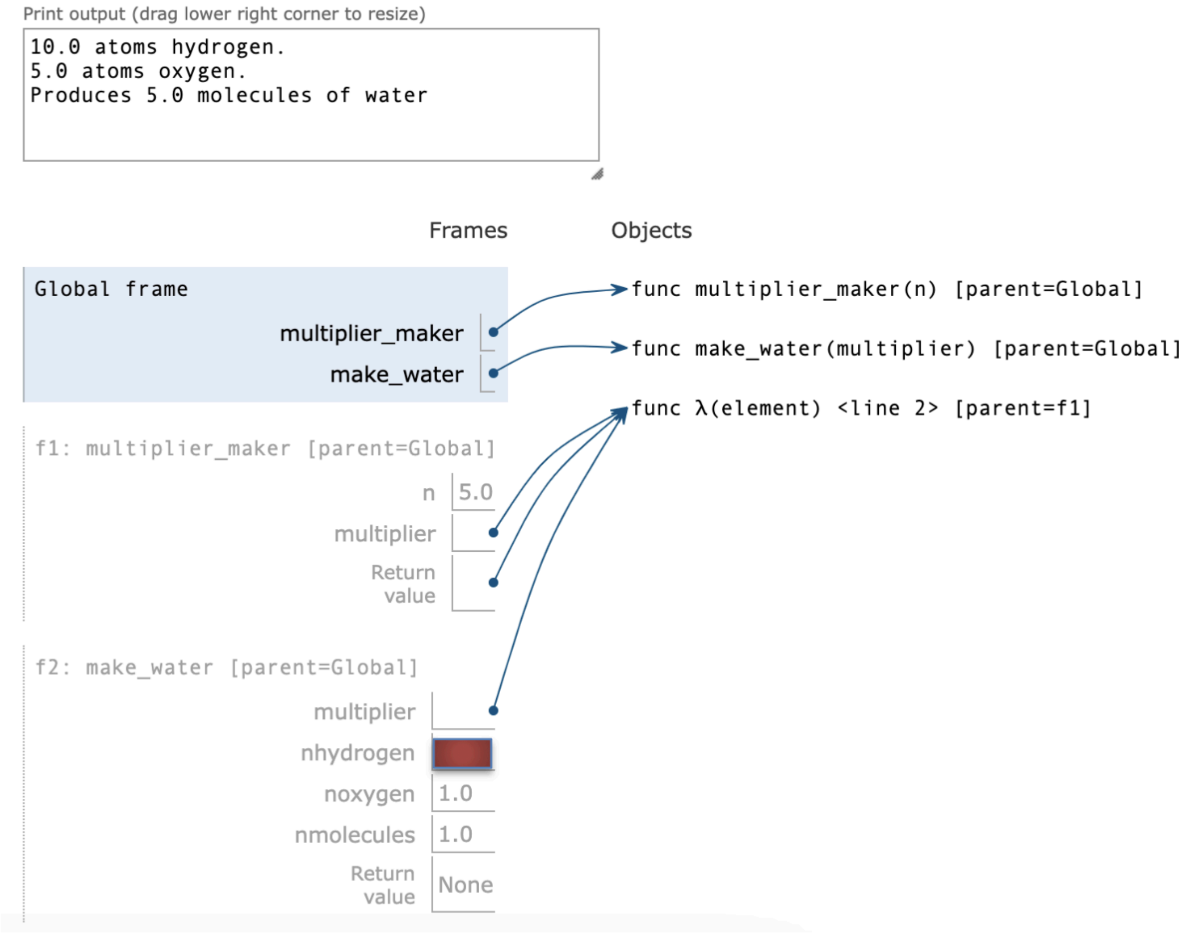
C. [✓ Float]

D. String

E. None

(8.0 points) **Chem 105**

Consider the environment diagram (and print output) below, followed by the code that generated it.



Print output (drag lower right corner to resize)

```
10.0 atoms hydrogen.
5.0 atoms oxygen.
Produces 5.0 molecules of water
```

Frames   Objects

Global frame
multiplier_maker → func multiplier_maker(n) [parent=Global]
make_water → func make_water(multiplier) [parent=Global]

func λ(element) <line 2> [parent=f1]

f1: multiplier_maker [parent=Global]
n 5.0
multiplier
Return value

f2: make_water [parent=Global]
multiplier
nhydrogen 🟥
noxygen 1.0
nmolecules 1.0
Return value None

```
def multiplier_maker(n):
      return lambda element : ________(a)* n

def make_water( multiplier ):
    nhydrogen,noxygen = 2.0, 1.0
    nmolecules = _________(d)
    print( str(multiplier(nhydrogen) ) + ' atoms hydrogen.' )
    print( str(multiplier(noxygen) ) + ' atoms oxygen.' )
    print( 'Produces ' + str(multiplier( ____(b) ))  + ' molecules of water.' )

make_water( _________(c)(5.0) )
```

11. (2.0 pt) Which one of these could fill in blank **(a)**?

A. n
B. element
C. multiplier
D. make_water
E. 3.0
F. 5.0

12. (2.0 pt) Which one of these could fill in blank **(b)** to get the environment diagram shown?

A. nmolecules
B. nhydrogen
C. noxygen
D. lambda x : x
E. 10.0
F. 1.0

13. (2.0 pt) Which one of these could fill in blank **(c)**?

A. element
B. make_water
C. nmolecules
D. lambda x : multiplier_maker(x)
E. multiplier
F. lambda element : element * n
G. multiplier_maker

14. (2.0 pt) Which one of these could fill in blank **(d)**?

A. 3.0
B. 6.0
C. 9.0
D. 1.0
E. 9.0, 3.0
F. 5.0
G. multiplier

(8 points) **Classes/Objects - Fill-in-the-blank and WWPD**

Consider the following class definitions:

```
class Bookshelf:
    def __init__(self, capacity, books=[]):
        self.capacity = capacity
        self.books = []
        for book in books:
            self.addBook(book)

    def addBook(self, book):
        if len(self.books) == self.capacity:
            print(f'Bookshelf is full. Could not add \'{book.title}\'.')
            return
        if ____(a)____: # verify that ‘book’ is the right type
            self.books.append(book)

    def __add__(self, other):
        if isinstance(other, Bookshelf):
            return [self, other]
        elif isinstance(other, Book):
            shelf2 = Bookshelf(self.capacity, list(self.books))
            shelf2.addBook(other)
            return shelf2

    def __str__(self): # this gets called by print() and str()
        book_string = ', '.join([str(a) for a in self.books])
        space = self.capacity - len(self.books)
        return f'Books: {book_string}; This shelf can fit {space} more books'

    def __repr__(self): # this gets called by repr() or when the object is
displayed within an iterable/collection
        book_string = ', '.join([repr(a) for a in self.books])
        return f'Bookshelf({self.capacity},[{book_string}])'

class Book:
    def ________(b)________:
        self.title, self.author = title, author

    def ________(c)________:
        return f'Book(\'{self.title}\',\'{self.author}\')'

    def ________(d)________:
        return self.title + ', written by ' + self.author
```

Indicate what should appear in blanks **(a)** - **(d)** above:

15.(1 pt) Which of the following should appear in blank **(a)**

A. `is Book('Frankenstein','Mary Shelley')`
B. `== Book('Frankenstein','Mary Shelley')`
C. `isinstance(book, Bookshelf)`
D. [✓ `isinstance(book, Book)`]
E. `== new Book()`

16.(2 pts) Which of the following should appear in blank **(b)**

A. [✓ `__init__(self, title, author)`]
B. `__add__(self, other)`
C. `__repr__(self)`
D. `__act__(self)`
E. `__str__(self)`

17.(1 pt) Which of the following should appear in blank **(c)**

A. `__init__(self, author, title)`
B. `__add__(self, other)`
C. [✓ `__repr__(self)`]
D. `__act__(self)`
E. `__str__(self)`

18.(1 pt) Which of the following should appear in blank **(d)**

A. `__init__(self, author, title)`
B. `__add__(self, other)`
C. `__repr__(self)`
D. `__act__(self)`
E. [✓ `__str__(self)`]

Given the code below, what would Python display for each of the following?

```
fiction_shelf = Bookshelf(10)
nonfiction_shelf = Bookshelf(1)
frankenstein = Book('Frankenstein','Mary Shelley')
coraline = Book('Coraline','Neil Gaiman')
print(frankenstein) (e)
adams = Book('John Adams','David McCullough')
hamilton = Book('Alexander Hamilton','Ron Chernow')
nonfiction_shelf.addBook(adams)
nonfiction_shelf += hamilton (f)
fiction_shelf.addBook(frankenstein)
fiction_shelf += coraline
str(fiction_shelf) (g)
```

19.(1 pt) Which of the following would be displayed by executing **(e)**

A. `Coraline`
B. `Frankenstein`
C. `Book('Frankenstein','Mary Shelley')`
D. `'Frankenstein'`
E. `'Frankenstein, written by Mary Shelley'`

20.(1 pt) Which of the following would be displayed by executing **(f)**

A. `Nothing`
B. `Bookshelf is full. Could not add 'Alexander Hamilton'.`
C. `[Book('John Adams','David McCullough'),Book('Alexander Hamilton','Ron`
D. `Chernow')]`
E. `Alexander Hamilton, written by Ron Chernow`
F. `[Bookshelf(1,'John Adams, Alexander Hamilton’)]`

21.(1 pt) Which of the following would be displayed by executing **(g)**

A. `'This shelf can fit 0 more books; Books: John Adams, written by David McCullough'`
B. `'Books: John Adams, written by David McCullough; This shelf can fit 0 more books'`
C. `'Coraline, written by Neil Gaiman; This shelf can fit 8 more books, Books: Frankenstein, written by Mary Shelley'`
D. `'Books: Frankenstein, written by Mary Shelley, Coraline, written by Neil Gaiman; This shelf can fit 8 more books'`
E. `'Books: Frankenstein, written by Mary Shelley, Coraline, written by Neil Gaiman; This shelf can fit 8 more books | Books: John Adams, written by David McCullough; This shelf can fit 0 more books'`

(7 points) **File I/O, Random Numbers, & Lists**

Consider the following program which is invoked by passing in three command-line arguments: 1) an input filename, 2) an output filename, and 3) and integer for the number of output sets to produce.

rand_num_game.py:

```
import sys
from random import randint

def randNumUpTo(n):
   return lambda : randint(1,n)

if __name__ == '__main__':
   iFile = open(sys.argv[1])
   oFile = ________________ (a)
   threshold = _________________ (b)

   lines = iFile.readlines()
   names = [player.strip() for player in lines] (c)

   oneToHundred = randNumUpTo(100) (d)
   for i in range(threshold): (e)
       oFile.write("Round " + str(i + 1) + "\n" )
       for name in names:
           multiplier = randNumUpTo(5)()
           randScore = oneToHundred() * multiplier (f)
           oFile.write(f"{name}: {randScore}\n") (g)
       oFile.write("\n")

   iFile.close()
   oFile.close()
```

Assume the program is invoked with the following command:

```
python rand_num_game.py players.txt scores.txt 4
```

And ***players.txt*** contains the following lines:

```
Dylan
Bob
Jim
Quentin
Ralph
```

22. (1 pt) Which of the following would be the correct syntax to open the output file (scores.txt) for writing at line **(a)**?

A. open(argv[2])
B. open(argv[2],’w’)
C. [✓ open(sys.argv[2],’w’)]
D. open(sys.argv[2])

23. (1 pt) Which of the following would be the correct syntax to convert the last command-line argument to an integer on line **(b)**?

A. argv[3]
B. sys.argv[3]
C. int(argv[3])
D. [✓ int(sys.argv[3])]
E. float(argv[3])
F. float(sys.argv[3])

24. (1 pt) What is the content of the list generated by the list comprehension on line **(c)**?

A. [Dylan\n, Bob\n, Jim\n, Quentin\n, Ralph\n]
B. [✓ ['Dylan', 'Bob', 'Jim', 'Quentin', 'Ralph']]
C. ['Dylan\n', 'Bob\n', 'Jim\n', 'Quentin\n', 'Ralph\n']
D. 'Dylan', 'Bob', 'Jim', 'Quentin', 'Ralph'

25. (1 pt) What is the type of the object bound to the name **oneToHundred** on line **(d)**?

A. List
B. [✓ Function]
C. String
D. Integer
E. Float

26. (1 pt) What is the range of values that ***i*** can have on line **(e)**?

A. 1, 2
B. 1, 2, 3
C. 0, 1, 2
D. [✓ 0, 1, 2, 3]
E. 0, 1, 2, 3, 4

27. (1 pt) What are the minimum and maximum values ***randScore*** can have on line **(f)**?

A. min = 0, max = 100
B. min = 1, max = 250
C. [✓ min = 1, max = 500]
D. min = 1, max = 100
E. min = 3, max = 500

28. (1 pt) If ***randScore*** is bound to the number 43, what will be the string written the **fifth** time line **(g)** is executed?

A. “Ralph: 43”
B. “Bob:43”
C. “Jim: 34\n”
D. [✓ “Ralph: 43\n”]
E. “Quentin:43\n”