

## 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**

## (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:

6. (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()`

7. (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)`

8. (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)`

9. (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)
```

10.(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'

11.(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 Chernow')]
- D. Chernow']]
- E. Alexander Hamilton, written by Ron Chernow
- F. [Bookshelf(1, 'John Adams, Alexander Hamilton')]

12.(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'

## (5 points) Inheritance

Consider the following class definitions of a base Person class and two child classes, Student and Teacher.

```
class Person:
    person_type = 'person' (a)

    def __init__(self, name):
        self._name = name (b)
    def get_data(self):
        raise NotImplementedError(f"You need to implement get_data().")
    def __str__(self):
        return f"{self._name} is a {self.person_type}."

class Student(Person):
    person_type = 'student'

    def __init__(self, name, takingClassCount):
        _____ (c)
        self._takingClassCount = takingClassCount
    def get_data(self):
        return (self._name , self._takingClassCount)
    def __str__(self):
        return repr(self) + f"\nCurrently enrolled in {self._takingClassCount}
classes.\n" + super().__str__()
    def __repr__(self):
        return f"Student({self._name}, {self._takingClassCount})"

class Teacher(Person):
    person_type = 'teacher'

    def __init__(self, name, teachingClassCount):
        _____ (c)
        self._teachingClassCount = teachingClassCount
    def get_data(self):
        return _____ (d)
    def __str__(self):
        return repr(self) + f"\nCurrently teaching {self._teachingClassCount}
classes.\n" + super().__str__()
    def __repr__(self):
        return f"Teacher({self._name}, {self._teachingClassCount})"
```

13.(1 pt) In terms of Object Oriented Programming what is **person\_type** on line (a)?

- A. Instance variable
- B. Class variable**
- C. Class method
- D. Dunder function

14.(1 pt) The instance variable `_name` on line **(b)** begins with an underscore. What convention does this represent?

- A. The variable is public and anyone can access and modify it.
- B. The variable is non-public and should only be used by methods of the class**
- C. The variable is private and normally only accessed by Python itself.

15.(1 pt) What line of code should appear on line **(c)** at the first of the Student and Teacher Classes' `__init__()` methods?

- A. `parent.__init__()`
- B. `self.__init__(name)`
- C. `Person.__init__(self)`
- D. `super().__init__(name)`**
- E. `super().__init__(self)`

16.(1 pt) The `get_data()` method is supposed to return a tuple containing the data about the person. What code should go in line **(d)** to do this for the Teacher class?

- A. `(data)`
- B. `_name, _teachingClassCount`
- C. `[_name, _teachingClassCount]`
- D. `(_name, _teachingClassCount)`
- E. `[self._name, self._teachingClassCount]`
- F. `(self._name, self._teachingClassCount)`**
- G. `{self._name, self._teachingClassCount}`

17.(1 pt) What would be printed if I executed the following code?

```
t = Teacher("Bob", 5)
print(t)
```

- A. `Teacher(Bob, 5).`  
Currently teaching 5 classes.
- B. `Teacher(Bob, 5)`  
Bob is a teacher.
- C. Currently teaching 5 classes.  
Bob is a teacher.
- D. Currently teaching 5 classes.
- E. `Teacher(Bob, 5)`  
Currently teaching 5 classes.  
Bob is a teacher.**

## (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): (d)
    return 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)
    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
```

18.(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])`

19.(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])`

20.(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'`

21.(1 pt) What is the type of the variable bound to the name **randNumUpTo** on line (d)?

- A. List
- B. Function
- C. String
- D. Integer
- E. Float

22.(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

23.(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`

24.(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"`