NOTE

- For all questions in this document, especially the ones that require the output to be specified for a given code segment, you should focus on understanding the semantics and logic aspects of the given code segment.
- All questions must be completed without using an IDE or a Python interpreter.

Section A Expression Evaluation, Function Calls, Arguments

Q1 Consider the following Python program:

```
a = len('four')
b = input('c')
d = 'e'
print('f')
```

Write down all the variables in the program.

```
a
b
d
```

Q2 Determine how many arguments this function call has:

```
print('1'+'2'+'1,2', '1,2', '1', '+', '2')
```

5

Q3 Consider the following code segment to answer the questions given below:

```
input("Welcome")
cat = 'Fluffy'
print("My cat's name is "+cat)
years = 4
print(cat,'is',years,'years old')
print(type(len('hello')))
```

How many **literal strings** are there in the given code segment?.6
How many **literal integers** are there in the given code segment? 1
How many **variables** are there in the given code segment? 2
Write the names of all **built-in functions that have** been called in the given code segment?print,input,len,type
How many **arguments** are there in the given code segment? 9

Q4. Consider the following Python program:

```
x = len('four')
y = len(input('enter your age: '))
```

Identify all arguments of some function calls in this program.

```
'four'
'enter your age: '
input('enter your age: ')
```

Q5. Determine what parts of the following statement are arguments:

```
answer = input('>, <')
'>, <'
```

Q6. When each of the following expressions is evaluated, either there is an error, or the result object will have a type. Select the correct option for each expression:

`+'3'`	`error`
`2+-3`	`int`
`'2'+'3'`	`str`
`+3`	`int`
`'3'+2`	`error`
`'2'+'cat'`	`str`

Q7. When each of the following expressions is evaluated, there will either be an error, or no error. Select the correct option for each expression:

`27-'1'`	`error`
`'2'*'4'`	`error`
`'2'+'4'`	`no error`
`-2+-2`	`no error`

Q8. Which of the following Python programs will produce the output line

>>>67
Select one:

`input('>>>'+'6'+'7')`
`print('>>>')+print('67')`
`print('>>>')` `print('67')`

`x=print('>>>')`
`y=print('67')`
`print(x,y)`

`print('>>>', '+6', '+7')`

Section B Selection Statements

Q1. Evaluate each of the following code segments and write the output. If there is no output write 'No output'.Assume the identifiers not_raining and sunny are both bound to True.

Code Segment	Output

`if not_raining:`
`  if sunny:`
`  print('Play outside')`
	`Play outside`

`if sunny:`
`  if not not_raining:`
`  print('Rainbow')`
`else:`
`  print('Blue Sky')`
	`No output`

`if not sunny or not_raining:`
`  print('Cloudy')`
`else:`
`  print('Watch TV')`
`print('Windy')`
	`Cloudy`
	`Windy`

Q2. When the following Python program is evaluated, it may assign different values to the variables a, b, and c, depending on the values of x and y.

```
# A pair of assignment statements goes here
a = '0'
b = '0'
c = '0'
if  x + y < 10 :
  x = x * 2
  if x + y >= 10:
     a = 'A'
  else:
     b = 'B'
     y = y + 2
if x + y > 10:
   c = 'C'
```

Select all of the pairs of assignment statements from below, which result in the following statements being true after the above program is evaluated:

a == '0' and b == 'B' and c == 'C'

x = 10 y = 1
x = 2 y = 5
x = 2 y = 3
x = 4 y = 2
x = 8 y = 1
x = 8 y = 2

Q3 Evaluate the following code segment and write the output,

Code	**Output**

x = 3
y = 7
if x + y >= 13:
    x = 7
    y = x + 1
    print(x+y)
elif y < 10:
    x = x*2
    print(x+y)
    if x + y >= 13 :
        x = x + y - 2
        print(x+y)
    else:
        print(x+y)
elif x > 2:
    y = y+2
    print(x+y)
    if x + y > 13:
        y = y*2
        print(x+y)
    else:
        print(x+y)
else:
    print(x+y)
print(x, y)
	13
	18
	11 7

Q4. When the following Python program is evaluated, it may assign the variables a, b, c, and d different values, depending on the object that x is bound to.

```
# An assignment statement goes here
a = 0
b = 0
c = 0
d = 0
if x > 0 and x <= 5:
    a = 10
elif x > 5 and x < 10:
    b = 10
elif x > 8 and x <= 12:
    c = 10
else:
    d = 10
```

Select all of the assignment statements from below, that result in the following statement to be True after the above program is evaluated:

(a == 0 and b == 0 and c == 10 and d == 0)

x = 7
x = 8
x = 9
x = 10
x = 11
x = 12
x = 13
x = 14

Q5. The following two Python programs are missing a pair of assignments for x and z. Pairs of assignment statements for x and z are given in the question below. For each given pair of assignments for x and z, the value of the variable color after the execution of the programs will either be identical or different for the two programs. Select the correct option for each pair of assignments:

Program 1

```
# A pair of assignment statements goes here
color = 'white'
if x == 'red':
    if z == 'yellow':
        color = 'orange'
    elif z == 'blue':
        color = 'violet'
    else:
        color = 'red'
```

Program 2

```
# A pair of assignment statements goes here
color = 'white'
if x == 'red' and z == 'yellow':
    color = 'orange'
if x == 'red' and z == 'blue':
    color ='violet'
if z == 'red':
    color = z
```

`x = 'red'` `z = 'black'`	`value of color is different`
`x = 'blue'` `z = 'red'`	`value of color is different`
`x = 'blue'` `z = 'yellow'`	`value of color is identical`
`x = 'red'` `z = 'blue'`	`value of color is identical`

Q6 Consider the Python program

```
if x == 'red':
    if z == 'yellow':
        print('orange')
    elif z == 'blue':
        print('violet')
    else:
        print(z)
```

Select all of the following Python programs that produce the same output as the above program for all possible assignments of the variables x and z.

```
if x == 'red' and z == 'yellow':
    print('orange')
if x == 'red' and z == 'blue':
    print('violet')
if z == 'red and x == z':
    print(z)
```

```
if x == 'red' and z == 'yellow':
    print('orange')
if x == 'red' and z == 'blue':
    print('violet')
else:
    print(z)
```

```
if x == 'red' and z == 'yellow':
    print('orange')
else:
    print(z)
if x == 'red' and z == 'blue':
    print('violet')
else:
    print(z)
```

```
if x == 'red' and z == 'yellow':
    print('orange')
if x == 'red' and z == 'blue':
    print('violet')
if x == 'red':
    print(z)
```

```
if x == 'red' and z == 'yellow':
    print('orange')
elif x == 'red' and z == 'blue':
    print('violet')
elif x == 'red':
    print(z)
```

Section C Iterative Statements and Sequences

Q1. When each of the following Python programs is evaluated there will be either an error or no error. Select the correct option for each program:

`for anInt in [1,2,3,4] :` `    print(anInt)`	**No Error**
`for anInt in '647':` `    anInt = anInt + 1`	**Error**
`for anInt in ['1','2','3']:` `    print(anInt)`	**No Error**
`for anInt in 748:` `    anInt = anInt + 1`	**Error**

Q2. When each of the following Python programs is evaluated, there will be either an error, or no error. Select the correct option for each program:

`things = ['a', 'b', 'c']` `things.append(1)`	No error
`things = [4]` `print(things[1])`	Error

`things = ['*']`
`for thing in things:`
`       print(things)`
	No Error

`things = ['Hello',2,4,3.142]`
`while (len(things) >= 0):`
`     print(things.append('bye')`
	Error

Q3. Assume there is a function is_ruler(x) that when called with an argument x returns True if the argument is equal to the string 'ruler', False otherwise.

Some of the following Python programs produce the following single line output:
'*'
Select all of the programs that produce this output.

```
words = ['pencil','eraser','marker','ruler']
length = len(words)-1
if is_ruler(len(words)):
    print('*')
```

```
words = ['pencil','eraser','marker','ruler']
length = len(words)
if is_ruler(words[length] - 1):
    print('*')
```

```
words = ['pencil', 'eraser', 'marker', 'ruler']
length = len(words)
if is_ruler(words[length - 1]):
    print('*')
```

```
words = ['pencil', 'eraser', 'marker', 'ruler']
last = len(words - 1)
if is_ruler(words[last]):
    print('*')
```

```
words = ['pencil', 'eraser', 'marker', 'ruler']
last = len(words) - 1
if is_ruler(words[last]):
    print('*')
```

```
words = ['pencil', 'eraser', 'marker', 'ruler']
last = len(words) - 1
if is_ruler(words[last - 1]):
    print('*')
```

Q4. For each of the following code segments either write the output produced by the code segment or write ‘Error’ if there is a runtime error.

	Code Segment	**What is the output?**

1.	`name='Barney'`
	`print(len(name))`
	`print(name[1])`
		`6`
		`a`

2.	`astring = 'hello'` `astring[0] = 'j'`	`Error`

3.	`someList=['Hello',23,47.8,'Goodbye',None,True,[67, 45]]`
	`someList[0] = 'Jello'`
	`print(someList[0][0])`
	`print(someList[4])`
	`print(type(someList[5]))`
	`print(someList[len(someList)-1])`
	`print(type(someList[len(someList)-5]))`
		`J`
		`None`
		`<class 'bool'>`
		`[67, 45]`
		`<class 'float'>`

4.	`alist = [2,4,6]`
	`for item in alist:`
	`  print(item)`
		`2`
		`4`
		`6`

5.	`nameList = ['apples','oranges','grapes']`
	`for aFruit in nameList:`
	`  if len(aFruit) == 6 and aFruit[0]=='g':`
	`  print(aFruit)`
		`grapes`

6.	`aString = 'bananas'`
	`count = 0`
	`for achar in aString:`
	`  if achar=='a' or achar=='n':`
	`  count = count+1`
	`print(count)`
		`5`

7.	`i=0`
	`while(i < 2):`
	`  print(i)`
	`  i=i+1`
		`0`
		`1`

8.	`i=2`
	`total = 0`
	`while(i > 0):`
	`  total = total+i`
	`  i=i-1`
	`print(i,total)`
		`0 3`

9.	word = 'chair'
	vowels = ['a','e','i','o','u']
	count = 0
	i = 0
	while (i < len(word)):
	    if word[i] in vowels:
	        count = count+1
	    i = i + 1
	print(count)
		2

10.	alist = [2,4,6,8,0,10,20,0,72] target = 0 location = 0 while (location < len(alist) and alist[location] != target): location = location +1 print(location)	4

11.	word_list = ['orange','chair','mouse','sandwich']
	for index in range(len(word_list)):
	    if index == 1:
	        print(word_list[index])
		chair

12	some_string = 'hello'
	for index in range(0,len(some_string)):
	    if index == 1:
	        print(some_string[index])
		e

13.	for index in range(3,7): print(index)	3 4 5 6
14.	for index in range(3): print(index)	0 1 2
15	alist = [2,4,6,8] for index in range(0,len(alist)): alist[index] = alist[index] *2 print(alist)	[4, 8, 12, 16]

Section D User Defined Functions

Q1 Replace the if statement in the given code segment with a single return statement so that the two code segments produce the same result.

Given Code Segment	**Equivalent Code Segment**

def isCold(temp):
    if temp <= 10:
        return True
    else:
        return False
	def isCold(temp):
	  return temp <= 10

Q2. For each of the following code segment write the output for the code segment:

	Code	**Write the output**

1.	`def main():`
	`  max=0`
	`  getMax(1,2,max)`
	`  print(max)`
	`def getMax(v1,v2,max):`
	`  if (v1>v2):`
	`  max=v1`
	`  else:`
	`  max=v2`
	`main()`
		`0`

2.	`def main():`
	`  times=3`
	`  print("Before function call : ",times)`
	`  nPrint("Welcome",times)`
	`  print("After function call : ",times)`
	`def nPrint(message,n):`
	`  while(n>0):`
	`  print(message)`
	`  n=n-1`
	`main()`
		`Before function call :  3`
		`Welcome`
		`Welcome`
		`Welcome`
		`After function call :  3`

3.	`def change(aList):`
	`  aList.append(4)`
	`  aList.append(5)`
	`def main():`
	`  mylist=[1,2,3]`
	`  change(mylist)`
	`  print(mylist)`
	`main()`
		`[1, 2, 3, 4, 5]`

4.	`def change(aList):`
	`  aList=[]`
	`  aList.append(4)`
	`  aList.append(5)`
	`  print(aList)`
	`def main():`
	`  mylist=[1,2,3]`
	`  change(mylist)`
	`  print(mylist)`
	`main()`
		`[4, 5]`
		`[1, 2, 3]`

5.	`def change(aString):`
	`  aString.upper()`
	`  return aString`
	`def main():`
	`  myString="hello"`
	`  result=change(myString)`
	`  print(result)`
	`main()`
		`hello`

Section E Code Completion Questions

Q1 Write a program that repeatedly prompts the user to input a word, sleeps for 2 seconds, print the word until the user enters either the word 'stop' or enters a word whose length is greater than 5. Some parts the program is given. Complete the missing parts:

```
import time
def main():
   word = input('Input word :')
   while word != 'stop' and len(word) <=5:
       time.sleep(2)
       print(word)
       word = input('Input word :')
main()
```

Q2 Write a function called banner that takes in a word in its parameter and prints each letter of the word on a new line. For example if word is blue a call to banner(word) would produce the following output:

```
b
l
u
e
```

Part of the function is given below. Complete the missing parts:

```
def banner(word):
    for letter in word:
        print(letter)
```

Q3 Write a function called fancy_banner that takes in two parameters.

- a word of type str
- how_many of type int

The function is required to print each letter of the word in a new line with how_many asterisks before and after each letter.
For example if word is blue and how_many is 3, call to banner(word,how_many) would produce the following output:

```
***b***
***l***
***u***
***e***
```

Part of the function is given below. Complete the missing parts:

```
def fancy_banner(word,how_many):
    for letter in word:
        padding = how_many * '*'
        print(padding + letter + padding)
```

Q4 Write a function called diff that takes in two parameters:

- x of type int
- y of type int

The function subtracts the smaller number from the larger number and returns the result. If the two numbers are the same, the function returns 0. For example if we make the following calls to diff:

```
print(diff(4,9)) it would output 5
print(diff(9,4)) it would output 5
print(diff(4,4)) it would output 0
```

Part of the function is given below. Complete the missing parts:

```
def diff(x,y):
    if x == y:
        result = 0
    elif x > y:
        result = x - y
    else:
        result = y - x
    return result
```

Q5 Write a function called compute_coordinates that takes the following parameters:

- location of type list that holds the coordinates x and y

The function modifies the given list such that the x coordinate is incremented by 2 and the y coordinate is incremented by 4.

Part of the function is given below. Complete the missing parts**:**

```
def compute_coordinates(location):
    location[0] = location[0] + 2
    location[1] = location[1] + 4
```

Q6 Complete the missing code segment so that it will calculate the length of the longest line of text in a .txt file, using a for loop.

```
    max_length = 0
    filename = 'random_file.txt'
    file = open(filename, 'r')

    data = file.read()
    lines = data.splitlines()

    for line in lines:
        if len(line) > max_length:
            max_length = len(line)

    print('The longest line was ' + str(max_length) + ' characters long.')
```

Q7 Complete this function, which takes a list of words called words as an argument and returns the longest word in the list.

```
def longest_word(words):
    longest_len = 0
    longest_word = None
    for word in words:
        if len(word) >= longest_len:
            longest_len = len(word)
            longest_word = word
    return longest_word
```

Q8 A list of names that are in a file called names.txt. Read the file and print the names to the console.

Write the function here:

```
def print_file(filename):
```

```
infile=open(filename,'r')
astring = infile.read()
alist = astring.splitlines()
for item in alist:
        print(item)
```

Q9 Write a function that creates a new list that has all the numbers that are in the given list except zeros.

For example if the given list is:

[78,0,90,67,5,34,0,45,98,0,78,0]

then the new list returned by the function would be:

[78,90,67,5,34,45,98.78]

Write the body of the following function:

```
def remove_zeros(alist):
```

```
newList = [ ]
for item in aList:
   if item != 0:
       newList.append(item)
return newList
```

Q10 Write a function that creates and returns a new list that contains all the words as orig_list, EXCEPT the words whose i th letter is the i th letter of the alphabet (i.e., 1st letter cannot be a, 2nd cannot be b, 3rd cannot be c, etc…):

```
alphabet  = 'abcdefghijklmnopqrstuvwxyz'
orig_list = [ 'bat', 'act', 'cat', 'rat', 'abs' ]

def remove_words(orig_list):
    new_list = [ ]
    for word in orig_list:
        valid_word = True
        for i in range(len(word)):
            if word[i] == alphabet[i]:
                valid_word = False
        if valid_word:
            new_list.append(word)
    return new_list

new_list = remove_words(orig_list)
```

Section F Classes and Objects Questions

Q1 Select True or False for each of the following statements:

To initialize a new object you must call the __init__ method explicitly in your code after you create the new object.	False
Every method call must include a dot delimiter.	True
The __init__ method is called once, automatically, on each newly created instance object.	True
The __init__ method is called automatically when the class definition statement is evaluated.	False
The __init__ method must have at least one parameter.	True
When __init__ is called, its first parameter is automatically bound to a newly created object.	True
A method call must have the same number of arguments as there are parameters in the definition of that method.	False
Every instance method definition has one parameter that is never specified as a normal argument when the method is called.	True

Q2 Consider the following Rectangle class:

```
class Rectangle:
    def __init__(self,x,y,width,height):
        # initializes the Rectangle object
        # - self is the Rectangle object
        # - x is the top left x coordinate of type int
        # - y is the top left y coordinate of type int
        # - width is the width of the Rectangle object of type int
        # - height is the height of the Rectangle object of type int
        self.x = x
        self.y = y
        self.width = width
        self.height = height

    def get_corners(self):
        # - self is the Rectangle object
        top_left = (self.x,self.y)
        top_right = (self.x +self.width, self.y)
        bottom_left =  (self.x, self.y + self.height)
        bottom_right = (self.x + self.width, self.y + self.height)

        corners = (top_left,top_right,bottom_left,bottom_right)
        return corners


    def collide_point(self,point):
        # returns True if point is in the rectangle
        # False otherwise
        # - self is of type Rectangle
        # - point is a tuple and it holds x and y coordinates of a point
        within_x_range = point[0] >= self.x and point[0] <= (self.x +
self.width)
        within_y_range = point[1] >= self.y and point[1] <= (self.y +
self.height)
        if within_x_range and within_y_range:
            return True
        else:
            return False
```

2a. Which one of the following Python statements would create a Rectangle object called shape at location x = 25 and y = 200 with width 400 and height 350.

`shape = Rectangle(self,25,200,400,350)`
`shape = Rectangle((25,200),400,350)`
`shape = Rectangle(25,200,400,350)`
`Rectangle = shape(25,200,400,350)`

`shape = Rectangle.init(25,200,400,350)`

2b. Which one of the following Python statements gets and prints the four corners of the shape that is of type Rectangle.

`shape.get_corners()`
`print(shape.get_corners())`
`print(shape.get_corners(self))`
`print(Rectangle.shape.get_corners())`

2c. Assume the following objects already exist:

- `point is bound to an object of type tuple whose value is (50,100)`
- `shape is an object of type Rectangle of width 10 pixels and height 10 pixels located at x = 30 and y = 50.`

Which one of the following boolean expression checks if the `point` is inside or on the borders of `shape` :

`(50 <= 30 and 50 <= 40) and (100 <= 50 and 100 <= 60)`
`(50 >= 40 and 50 <= 30) or (100 >= 60 and 100 <= 50)`
`(50 >= 30 and 50 <= 40) and (100 >= 50 and 100 <= 60)`
`(50 >= 30 or 50 <= 40) and (100 >= 50 or 100 <= 60)`

2d.Which one of the following statements would need to be added to the `collide_point` method in the Rectangle class if we had to call the `get_corners` method inside the `collide_point` method?

`get_corners()`
`Rectangle.get_corners()`
`self.get_corners()`
`a method cannot be called inside another method`

Section G Pre-Poke Framework Questions

Q1 Refer to the [Pre-Poke framework] to answer the following questions:

1a. What is the type of object the identifier `self` in the Game class is bound to ?

function
Game
surface
pygame

1b. Refer to line 46 and 47. How many arguments need to be passed to the init method of the Dot class to create the small_dot and big_dot objects?

8
7
6
5

1c The number of methods called in the play method of the Game class varies based upon what objects self.continue_game and self.close_clicked are bound to. Write the number of methods and the names of method for each of the following cases:

`self.close_clicked`	`self.continue_game`	Number of methods	Names of methods
True	True	0	
True	False	0	
False	True	5	handle_events,draw,update, decide_continue,tick
False	False	3	handle_events,draw,tick

1d. Which instance attribute in the Game class is updated if the user closes the graphical window i.e. a `pygame.QUIT` event occurs?

`self.close_clicked`
`self.continue_game`
`self.small_dot`
`no instance attribute is updated`
`self.big_dot`

1e. Which method in the Game class, when called inside the play method of the Game class, causes the Dot objects to change position?

`draw`
`update`
`decide_continue`
`handle_events`

1f Under what condition is the identifier `self.continue_game` bound to False? Write the condition.

`self.frame_counter > self.max_frames`

1g How many instance attributes are there in the Dot class?

5

1h. The move method in the Dot class can be written without using the for loop. Which one of the following code segments would be the same as the existing body of the move method?

`self.center[0] = (self.center[0] + self.velocity[0])` `self.center[1] = (self.center[1] + self.velocity[1])`
`self.center[i] = (self.center[i] + self.velocity[i])` `self.center[i] = (self.center[i] + self.velocity[i])`

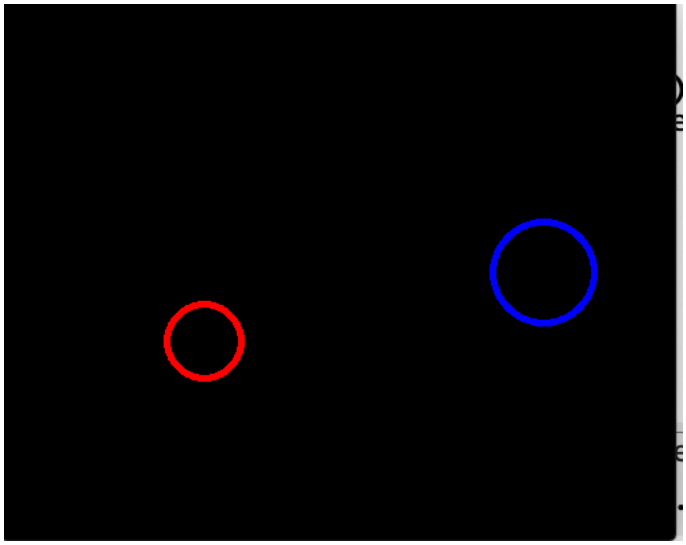
`self.center[0] = (self.center[0] + self.velocity[0])`
`self.center[1] = (self.center[1] + self.velocity[1])`
`self.center[2] = (self.center[2] + self.velocity[2])`

`(self.center[0] + self.velocity[0])` `(self.center[1] + self.velocity[1])`

1i. Refer to the draw method of the Dot class to write the name of the pygame function to draw the Dot object.

`pygame.draw.circle`

1j. Refer to the pygame documentation on pygame.draw.circle. Add a 5th parameter in the call to `pygame.draw.circle` such that the draw method in the Dot class draws a ring with 5 pixels line thickness instead of a dot. After the modification, the program should display red and blue rings as shown in the following picture:



```
pygame.draw.circle(self.surface, self.color, self.center, self.radius,5)
```

Section H Poke The Dots Reflection

Q1 Use the following code fragment from Poke The Dots to answer the questions.

```
def play(self):
        # Play the game until the player presses the close box.
        # - self is the Game that should be continued or not.

        while not self.close_clicked:
            # play frame
            self.handle_events()
            self.draw()
            if self.continue_game:
               self.update()
               self.decide_continue()
            self.game_Clock.tick(self.FPS) # run at most with FPS
```

1.1	What object is self.close_clicked bound to when the play method is called from the main function for the first time?
	It is bound to False.
1.2	For what condition (write the identifier and its value) will the body of the while loop not be evaluated?
	If self.close_clicked is bound to True then the body of the while loop will not be evaluated.
1.3	How does the handle_event method in the Game class impact the evaluation of the condition of the while loop in the play method?

	If the close box of the window is clicked (a QUIT event) then the handle_event method bounds self.close_clicked to True else self.close_clicked remains bound to False.
1.4	What object is self.continue_game bound to when the play method is called in the main function for the first time?
	It is bound to True.
1.5	What effect does the call to the update method have on the Dot objects?
	The update method in the Game class moves the big dot and the small dot.
1.6	What single assignment statement can be included in the decide_continue method of the Game class so that the condition of the if statement becomes False?
	self.contine_game = False
1.7	Why is the tick method in pygame.time.Clock class called in each iteration of the while loop?
	This method is called once per frame to limit the runtime speed of a game. By calling self.game_clock.tick(60) once per frame, the program will never run at more than 60 frames per second.

Q2 Refer to the pre-poke framework. List all of the instance attributes of the Game class. For each attribute, identify what type that identifier is bound to.

Instance Attributes	Type
surface	pygame.Surface
bg_color	pygame.Color
FPS	int
game_Clock	pygame.time.Clock
close_clicked	bool
continue_game	bool
small_dot	Dot
big_dot	Dot

Q3 For each of the following classes, listed in the Class column, choose and match a line of code, listed in the Initialization Statement column, that is used to obtain a new instance of that class.

	Class	Initialization Statement
1	**Game**	`w_surface = pygame.display.set_mode((500, 400))` **3**
2	**pygame.Color**	`pygame.draw.circle(self.surface, self.color, self.center, self.radius)` **6**
3	**pygame.Surface**	`game = Game(w_surface)` **1**
4	**list**	`self.bg_color = pygame.Color('black')` **2**

5	**tuple**	`events = pygame.event.get()` 4
6	**pygame.Rect**	`size = self.surface.get_size()` 5

Q4 The code for the randomize_dot method in the Dot class is given. Identify the problem that may occur when we use the given code to randomize the dot. Assuming the Dot class has a radius attribute, rewrite the given code to fix the problem.

Given Code

```
class Dot:
...
   def randomize(self):
        size = self.surface.get_size()
        for coord in range(0, 2):
            self.center[coord] = random.randint(0, size[coord])
```

Identify the problem

When coord is 0, the x coordinate of the dot center could be assigned a random number between 0 to width of the window (both endpoints inclusive) and when coord is 1, the y coordinate of the dot center could be assigned a random number between 0 to height of the window (both endpoints inclusive).

Since the dot is drawn around the center, the above code could cause up to half of the dot to be outside the window surface.

Fix the problem - rewrite the given code

```
class Dot:
...
   def randomize(self):
        size = self.surface.get_size()
        for coord in range(0, 2):
            self.center[coord] = random.randint(self.radius, size[coord] -
self.radius)
```

Q5 The following table gives a list of expressions used in the move method of the Dot class and the purpose they are being used for. Match each expression (in the Expression column) to its purpose (in the Purpose column).

	Expression	**Purpose**
1	`self.center[0] < self.radius`	check if dot has moved past bottom edge (4)
2	`self.center[0] +self.radius > size[0]`	check if dot has moved past left edge (1)

3	`self.center[1] < self.radius`	reverses horizontal direction of the dot (5)
4	`self.center[1] + self.radius > size[1]`	reverse vertical direction of the dot (6)
5	`self.velocity[0] = -self.velocity[0]`	check if dot has moved past right edge (2)
6	`self.velocity[1] = -self.velocity[1]`	check if dot has moved past top edge (3)

Q6 Modify the given code for the draw_score method in the Game class such that the score is displayed at the top right corner of the window, in font size 100, in green foreground color on a black background.

Given Code

```
def draw_score(self):
      # Draw the time since the game began as a score
      # in white on the window's background.
      # - self is the Game to draw for.

      text_string = 'Score:' + str(self.score)
      text_fg_color = pygame.Color('white')
      text_font = pygame.font.SysFont('', 70)
      text_image = text_font.render(text_string, True, text_fg_color,
self.bg_color)
      text_top_left_corner = (0, 0)
      self.surface.blit(text_image, text_top_left_corner)
```

Modified Code

```
def draw_score(self):
      # Draw the time since the game began as a score
      # in white on the window's background.
      # - self is the Game to draw for.

      text_string = 'Score:' + str(self.score)
      text_fg_color = pygame.Color('green')
      text_font = pygame.font.SysFont('', 100)
      text_image = text_font.render(text_string, True, text_fg_color,
self.bg_color)
      text_top_right_corner = (self.surface.get_width() - text_image.get_width(),0)
      self.surface.blit(text_image, text_top_right_corner)
```

Q7 Write the Python code for drawing Hello in red 70 font size on green background at the bottom left corner of the window whose surface is bound to an identifier w_surface.

```
text_string = 'Hello'
text_fg_color = pygame.Color('red')
text_bg_color = pygame.Color('green')
text_font = pygame.font.SysFont('', 70)
text_image = text_font.render(text_string, True, text_fg_color, text_bg_color)
text_bottom_left_corner=(0,w_surface.get_height()- text_image.get_height())
w_surface.blit(text_image, text_bottom_left_corner)
```

Q8 What are the different kinds of events that are handled in Poke Version 3?

QUIT, MOUSEBUTTONUP

Q9 Refer to pygame documentation and list at least four different kinds of events that are not handled in Poke Version 3:

MOUSEBUTTONDOWN, MOUSEMOTION, KEYUP, KEYDOWN

Q10 Which one of the following methods (Choice 1 or Choice 2) would you choose to handle the events in the game? Explain your answer,

Choice 1

```
def handle_event(self):
    event = pygame.event.poll()
    if event.type == QUIT:
         self.close_clicked = True
    elif event.type == MOUSEBUTTONUP
         self.handle_mouse_up()
```

Choice 2

```
def handle_event(self):
   event = pygame.event.poll()
   if event.type == QUIT:
      self.close_clicked = True
   elif event.type == MOUSEBUTTONUP and self.continue_game:
      self.handle_mouse_up()
```

Note that handle_mouse_up may or may not have an event parameter since it is not needed.

Choice 2 code is correct. Choice 2 code will handle the MOUSEBUTTONUP event provided there was a mouse click AND the dots have not collided. Choice 1 code does not check the identifier self.continue_game. Therefore, the game continues to move the dots to random locations in

response to mouse clicks even when the dots have collided and the game is over. The event parameter in handle_mouse_up is optional since it is not needed in this game.

Q11 Which one of the following methods (Choice 1 or Choice 2) would you choose to create and randomize dots such that they are not touching or intersecting each other at the start of the game.

Choice 1

```
def create_dots(self):
     # create and randomize the dots at the start of the game such that
     # the dots are not touching each other at the start of the game
     # - self is the Dot
     self.small_dot = Dot('red', 30, [50, 50], [1, 2], self.surface)
     self.big_dot = Dot('blue', 40, [200, 100], [2, 1], self.surface)
     # Randomize the dots
     self.small_dot.randomize()
     self.big_dot.randomize()
     while self.small_dot.intersects(self.big_dot):
        self.small_dot.randomize()
        self.big_dot.randomize()
```

Choice 2

```
def create_dots(self):
     # create and randomize the dots at the start of the game such that
     # the dots are not touching each other at the start of the game
     # - self is the Dot
     self.small_dot = Dot('red', 30, [50, 50], [1, 2], self.surface)
     self.big_dot = Dot('blue', 40, [200, 100], [2, 1], self.surface)
     while self.small_dot.intersects(self.big_dot):
        self.small_dot.randomize()
        self.big_dot.randomize()
```

Choice 1 is the correct answer.

Choice 2 creates the dots at locations [50,50] and [200,200]. The dots at these locations will not be intersecting or touching each other. The while loop condition will be false. Hence this method, will not place the dots at random locations at the start of the game.

Section I Additional Miscellaneous Questions

Q1 Trace through the following code three times, one for each set of values for x, and y, and record the results in the table below:

```
c = 5
k = (x + y) ** 2
z = 0
if k < c ** 2:
 x = c
 y = x + c
 z = x * y
elif k == c ** 2:
 z = k
else:
 x = x - c
 y = y - (y - c)
 z = x * y
print (z)
```

Variable assignment	Output
x = 2 y = 3	**25**
x = 3 y = 3	**-10**
x = 1 y = 3	**50**

Q2 Trace through the following program and fill in the table below:

```
print("Enter a number between 50 and 100: ")
number = int(input())
if number >= 50:
    if number <= 100:
         print('Correct.')
else:
    print('Incorrect!')
```

Input number	Printed output
38	**Incorrect!**
50	**Correct.**
72	**Correct.**
104	**No output (not handled by the code)**

Which scenarios are not handled by the if – else statements above? Modify the code to handle these scenarios.

The code does not display any output for numbers greater than 100. We can fix this problem by writing the code in the following manner:

```
print("Enter a number between 50 and 100: ")
number = int(input())
if number >= 50:
    if number <= 100:
        print('Correct.')
    else:
        print('Incorrect!')
else:
    print('Incorrect!')
```

Q3 Trace through the nested while loops and write the values of k,l and m as they change in the given table. What's the output?

```
k = 7
l = 8
m = 10
while l < m:
    while l > k:
        l = l - 1
    print(k, l, m)
    l = l + 3
```

k	l	m
7	8	10
7	7	10
7	10	10

The output is :

```
7 7 10
```

Q4 For each of the code examples, determine their output:

a)

```
word = "Pumpkin"
for item in word:
    print(item)
```

```
P
u
m
p
k
i
n
```

b)

```
word = "Pasta"
for item in range(len(word)):
    print(str(item) + " : " + word[item])
```

```
0 : P
1 : a
2 : s
3 : t
4 : a
```

c)

```
word = ["Plum", "Peanut", "Pear", "Pans"]
i = 0
for item in word:
    print(item[i])
    i = i + 1
```

```
P
e
a
s
```

Q5 What's the output of the following code? The operator % is the modulo operator, which evaluates to the remainder of a division, e.g., 9 % 3 evaluates to 0 since the division 9 / 3 has no remainder, 10 % 3 evaluates to 1 since 10 divided by 3 has a remainder of 1, etc.

```
fruits = ['apple', 'orange', 'banana', 'kiwi', 'cherry', 'strawberry',
'pear', 'plum']
for fruit in range(len(fruits)):
      if fruit % 3 != 0:
         print(fruits[fruit])
```

```
orange
banana
cherry
strawberry
```

```
plum
```

Q6 Determine the output for the following program

```
def foo(alist):
    blist = []
    for index in range(len(alist)):
        blist.append(alist[index])
    blist.reverse()
    print(blist)

def main():
    alist = ['a','b','c','d']
    foo(alist)
    print(alist)
main()
```

```
['d', 'c', 'b', 'a']
['a', 'b', 'c', 'd']
```

Q7 Determine the output for the following program:

```
def foo(alist):
    blist = []
    for index in range(len(alist)):
        blist.append(alist[index])
    blist.reverse()
    return blist

def main():
    alist = ['a','b','c','d']
    alist = foo(alist)
    print(alist)
main()
```

```
['d', 'c', 'b', 'a']
```

Q8 Determine the output for the following program:

```
def foo(alist):
    for item in alist:
        item.upper()

def main():
    alist = ['a','b','c','d']
    result = foo(alist)
    print(alist)
    print(result)
main()
```

```
['a', 'b', 'c', 'd']
None
```

Q9 Determine the output for the following program:

```
def foo(astring,old_char,new_char):
    result = ''
    for char in astring:
        if char == old_char:
            result = result + new_char
        else:
            result = result + char
    return result

def main():
    alist = ['a','#','b','c','#','d']
    astring = ''.join(alist)
    print(astring)
    # result = foo(astring,'*','#') - this is a typo - the old and new
    # are not in the correct order
    result = foo(astring,'#','*') # typo corrected
    print(alist)
    print(result)
main()
```

```
a#bc#d
['a', '#', 'b', 'c', '#', 'd']
a*bc*d
```

Q10 Determine the output for the following program:

```
def foo(astring):
    index = 0
    result = ''
    for index in range(len(astring)):
        result = astring[index] + result
        print(result)
    return result
def main():
    astring = 'oranges'
    result = foo(astring)
    print(result)
main()
```

```
o
ro
aro
naro
gnaro
egnaro
segnaro
segnaro
```

Q11 Determine the output for the following program:

```
def function_a(a_list, high_num):
    for i in range(len(a_list)):
        if a_list[i] > high_num:
            a_list[i] = a_list[i] + 1
        else:
            a_list[i] = 0
    high_num = 0

def main():
    list_1 = [1,3, 6, 4, 1, 2, 8]
    my_num = 4
    function_a(list_1, my_num)
    print(list_1)
    print(my_num)

main()
```

```
[0, 0, 7, 0, 0, 0, 9]
4
```

Q12 Determine the output for the following program:

```
def function_b(b_list, high_num):
    c_list = [0, 0, 0, 0, 0, 0,0]
    i = 0
    for num in b_list:
        if num > high_num:
            c_list[i] = num
        i = i + 1
    b_list = c_list

def main():
    b_list = [1,3, 6, 4, 1, 2, 8]
    high_num = 4
    function_b(b_list, high_num)
    print(b_list)

main()
```

```
[1, 3, 6, 4, 1, 2, 8]
```

Q13 Write a function that generates a sequence of squares of numbers for all integers from a given smallest number to a given largest number. For instance, the function call `square(3, 6)` should return a list containing the numbers 3^2, 4^2, 5^2, 6^2, i.e., the list `[9, 16, 25, 36]`.

The function definition and return value are given as follows:

```
def squares(a,b):
# computes the squares of all integers between a and b
# - a is the first int number to square
# - b is the last int number to square
# returns a list of the squares from a² to b²
```

```
def squares(a,b):
      sqrs = []
      for i in range(a,b+1):
         sqrs.append(i * i)
      return sqrs
```

Q14 The given code changes the entries in the dimensions list object by using the entries in the change list object. Modify the given code such that it satisfies the requirement of Replacing all adjacent duplicate line groups with iteration as specified in Section 5 of the Software Quality Test Document.

Given Code

```
dimensions = [200,700,500,900]
change = [2,4,6,8]
dimensions[0] = dimensions[0] + change[0]
dimensions[1] = dimensions[1] + change[1]
dimensions[2] = dimensions[2] + change[2]
dimensions[3] = dimensions[3] + change[3]
print(dimensions)
```

Modified Code

```
dimensions = [200,700,500,900]
change = [2,4,6,8]
for index in range(0,4):
   dimensions[index] = dimensions[index]+change[index]
print(dimensions)
```

Section J Nested For Loops

Q1 What is the output for each of the given program segments? If there is no output for the given program segment indicate 'no output' or if there is an error in executing the program segment, indicate 'error is reported by the program'.

	Given Program Segment	Output

1.1	`for i in range(1,4):`
	`  for j in range(1,4):`
	`  if(i+j)%2 == 0:`
	`  print('*')`
		`*`
		`*`
		`*`
		`*`
		`*`

1.2	`for x in ['ab','cd']:`
	`  for y in x:`
	`  print(y)`
		`a`
		`b`
		`c`
		`d`

1.3	`alist=[[0,1,2],['Fred','Barney','Wilma']]`
	`blist = alist[1]`
	`blist[0]='Sara'`
	`print(alist[1])`
	`print(alist[1][2][0])`
		`['Sara', 'Barney', 'Wilma']`
		`W`

1.4	`grid = [ ["a", "b", "c"], ["d", "e", "f"], ["g", "h", "i"] ]`
	`for i in range(len(grid)):`
	`  print(grid[len(grid)-i-1][i])`
		`g`
		`e`
		`c`

1.5	`grid = [ ["a", "b", "c"], ["d", "e", "f"], ["g", "h", "i"] ]`
	`for i in range(len(grid)):`
	`  if i % 2 == 0:`
	`  print(grid[i][i])`
		`a`
		`i`

1.6	grid = [["a", "b", "c"], ["d", "e", "f"], ["g", "h", "i"]] for i in range(len(grid)): print(grid[i][i%2])	a e g

1.7	grid = [[0, 1, 2], [1, 2, 4], [2, 4, 8]]
	for i in range(3):
	    for j in range(3):
	        if i+j == grid[i][j]:
	            print(i+j)
		0
		1
		2
		1
		2
		2

2) A classroom has three rows of three chairs. The teacher has created the following seating chart.

The first list in the seating_chart holds the names of the student seated in the first row, the second list holds the names of the students seated in the second row and so on.

```
seating_chart =
[['Amy','Sarah','Brian'],['Donald','Jacob','Zoey'],['Amanda','Bob','Dora']]
```

Choose a program segment that would work with the given seating chart to produce the following output:

```
List of students at position  0 : ['Amy', 'Donald', 'Amanda']
List of students at position  1 : ['Sarah', 'Jacob', 'Bob']
List of students at position  2 : ['Brian', 'Zoey', 'Dora']
```

a.	for col_index in range(3):
	    column = []
	    for row_index in range(3):
	        student = seating_chart[row_index][col_index]
	        column.append(student)
	    print('List of students at position ',col_index,':',column)

b.	for row_index in range(3):
	    column = []
	    for col_index in range(3):
	        student = seating_chart[row_index][col_index]
	        column.append(student)
	    print('List of students at position ',row_index,':',column)

c.	for row_index in range(3):
	    column = []
	    for col_index in range(3):
	        student = seating_chart[row_index][col_index]
	    column.append(student)
	    print('List of students at position ',row_index,':',column)

d.	for col_index in range(3):

	column = [] for row_index in range(3): student = seating_chart[row_index][col_index] column.append(student) print('List of students at position ',col_index,':',column)

Section K Dictionaries

Q1 Write the output for each of the following program segments ***without*** using Wing IDE. If an error occurs just write the name of the error.

	Code Segment	Write the output
1.	speeds={} speeds['B777']=896 speeds['A330']=840 speeds['E190']=811 print(speeds)	{'B777': 896, 'A330': 840, 'E190': 811}
2.	speeds = {'B777': 896, 'A330': 840, 'E190': 811} speeds['E190']=900 speeds['F16']=800 print(speeds)	{'B777': 896, 'A330': 840, 'E190': 900, 'F16': 800}

3.	speeds = {'B777': 896, 'A330': 840, 'E190': 811}
	for k,v in speeds.items():
	print(k,v)
	for values in speeds.values():
	print(values)
	for keys in speeds.keys():
	print(keys)
		B777 896
		A330 840
		E190 811
		896
		840
		811
		B777
		A330
		E190

4.	speeds = {'B777': 896, 'A330': 840, 'E190': 811} print(speeds['B769'])	KeyError
5,	speeds = {'B777': 896, 'A330': 840, 'E190': 811} print(speeds.get('B769',"Not in dictionary")) print(speeds.get('B769')) print(speeds.get('E190'))	Not in dictionary None 811
6.	speeds = {'B777': 896, 'A330': 840, 'E190': 811} print(speeds.pop('E194'))	KeyError
7.	speeds = {'B777': 896, 'A330': 840, 'E190': 811} print(speeds.pop('E194','Item not found'))	Item not found 840

	`print(speeds.pop('A330'))` `print(speeds)`	`{'B777': 896, 'E190': 811}`

8.	`speeds = {'B777': 896, 'A330': 840, 'E190': 900}`
	`print('E190' in speeds)`
	`print(900 in speeds)`
	`print(900 in speeds.values())`
		`True`
		`False`
		`True`

Q2 Write a function that reads in a file containing an unspecified number of integers, separated by spaces, and finds the number(s) that have the most occurrences. For example, if the file contains 2 2 3 3 7 3 4 2 then the program reports that 3 and 2 occur three times. The function should use a dictionary to determine the number of occurrences of each number in the file.

`def count_occurrences(filename):`

```
# read the numbers and add them to a list
fl = open('filename.txt', 'r')
numbers = fl.read().split()

# count the numbers
numbers_count = {}
for number in numbers:
   number = int(number)
   numbers_count[number] = numbers_count.get(number, 0) + 1

# collect the number(s) with the highest count in a list
maxcount = 0
for number in numbers_count:
   count = numbers_count[number]
   if count > maxcount:
      maxcount = count
      most_common_numbers = [number]
   elif count == maxcount:
      most_common_numbers.append(number)

# print the output
output = ('the most common numbers were '
          + str(most_common_numbers).strip('[]')
          + '. Each occurred ' + str(maxcount) + ' times.')
print(output)
```