**CITC-1301 Introduction to Programming**

**Final Review**

## Chapter 6 – Files and Exceptions

1. A file that data is written to is known as a(n) \_\_\_\_\_\_\_\_\_\_.
2. input file
3. output file
4. sequential access file
5. binary file
6. A file that data is read from is known as a(n) \_\_\_\_\_\_\_\_\_\_.
7. input file
8. output file
9. sequential access file
10. binary file
11. Before a file can be used by a program, it must be \_\_\_\_\_\_\_\_\_\_.
12. formatted
13. encrypted
14. closed
15. opened
16. When a program is finished using a file, it should do this.
17. erase the file
18. open the file
19. close the file
20. encrypt the file
21. The contents of this type of file can be viewed in an editor such as Notepad.
22. text file
23. binary file
24. English file
25. human-readable file
26. This type of file contains data that has not been converted to text.
27. text file
28. binary file
29. Unicode file
30. symbolic file
31. When working with this type of file, you access its data from the beginning of the file to the end of the file.
32. ordered access
33. binary access
34. direct access
35. sequential access
36. When working with this type of file, you can jump directly to any piece of data in the file without reading the data that comes before it.
37. ordered access
38. binary access
39. direct access – or random access
40. sequential access
41. This is a small “holding section” in memory that many systems write data to before writing the data to a file.
42. buffer
43. variable
44. virtual file
45. temporary file
46. This marks the location of the next item that will be read from a file.
47. input position
48. delimiter
49. pointer
50. read position
51. When a file is opened in this mode, data will be written at the end of the file’s existing contents.
52. output mode
53. append mode
54. backup mode
55. read-only mode
56. This is a single piece of data within a record.
57. field
58. variable
59. delimiter
60. subrecord
61. When an exception is generated, it is said to have been \_\_\_\_\_\_\_\_\_\_.
62. built
63. raised
64. caught
65. killed
66. This is a section of code that gracefully responds to exceptions.
67. exception generator
68. exception manipulator
69. exception handler
70. exception monitor
71. You write this statement to respond to exceptions.
72. run/handle
73. try/except
74. try/handle
75. attempt/except
76. (True|False) When working with a sequential access file, you can jump directly to any piece of data in the file without reading the data that comes before it.
77. (True|False) When you open a file that file already exists on the disk using the 'w' mode, the contents of the existing file will be erased.
78. (True|False) The process of opening a file is only necessary with input files. Output files are automatically opened when data is written to them.
79. (True|False) When an input file is opened, its read position is initially set to the first item in the file.
80. (True|False) When a file that already exists is opened in append mode, the file’s existing contents are erased.
81. (True|False) If you do not handle an exception, it is ignored by the Python interpreter, and the program continues to execute.
82. (True|False) You can have more than one except clause in a try/except statement.
83. (True|False) The else suite in a try/except statement executes only if a statement in the try suite raises an exception.
84. (True|False) The finally suite in a try/except statement executes only if no exceptions are raised by statements in the try suite.
85. Describe the three steps that must be taken when a file is used by a program. Open file – pass name – read or write – close file
86. Why should a program close a file when it’s finished using it? Locked until program closes file
87. What is a file’s read position? Where is the read position when a file is first opened for reading?
88. Which mode should a file be opened in to allow data to be written to it, erasing any previous data? write
89. If a file does not exist and a program attempts to open it in append mode, what happens?python will create it

## Chapter 7 – Lists and Tuples

1. This term refers to an individual item in a list.
2. element
3. bin
4. cubbyhole
5. slot
6. This is a number that identifies an item in a list.
7. element
8. index
9. bookmark
10. identifier
11. This is the first index in a list.
12. −1
13. 1
14. 0
15. The size of the list minus one
16. This is the last index in a list.
17. 1
18. 99
19. 0
20. The size of the list minus one
21. This will happen if you try to use an index that is out of range for a list.
22. A ValueError exception will occur.
23. An IndexError exception will occur.
24. The list will be erased, and the program will continue to run.
25. Nothing—the invalid index will be ignored.
26. This function returns the length of a list.
27. length()
28. size()
29. len()
30. lengthof()
31. When the \* operator’s left operand is a list and its right operand is an integer, the operator becomes this.
32. The multiplication operator
33. The repetition operator
34. The initialization operator
35. Nothing—the operator does not support those types of operands.
36. This list method adds an item to the end of an existing list.
37. add()
38. add\_to()
39. increase()
40. append()
41. This removes an item at a specific index in a list.
42. remove() method
43. delete() method
44. del statement
45. kill() method
46. Assume the following statement appears in a program:

mylist = []

Which of the following statements would you use to add the string 'Labrador' to the list at index 0?

1. mylist[0] = 'Labrador'
2. mylist.insert(0, 'Labrador')
3. mylist.append('Labrador')
4. mylist.insert('Labrador', 0)
5. If you call the index method to locate an item in a list and the item is not found, this happens.
6. A ValueError exception is raised.
7. An InvalidIndex exception is raised.
8. The method returns −1.
9. Nothing happens. The program continues running at the next statement.
10. This built-in function returns the highest value in a list.
11. highest()
12. max()
13. greatest()
14. best\_of()
15. This file object method returns a list containing the file’s contents.
16. to\_list()
17. getlist()
18. readline()
19. readlines()
20. Which of the following statements creates a tuple?
21. values = [1, 2, 3, 4]
22. values = {1, 2, 3, 4}
23. values = (1)
24. values = (1,)
25. (True|False) Lists in Python are immutable.
26. (True|False) Tuples in Python are immutable.
27. (True|False) The del statement deletes an item at a specified index in a list.
28. (True|False) Assume list1 references a list. After the following statement executes, list1 and list2 will reference two identical but separate lists in memory: list2 = list1
29. (True|False) A file object’s writelines() method automatically writes a newline ('\n') after writing each list item to the file.
30. (True|False) You can use the + operator to concatenate two lists.
31. (True|False) A list can be an element in another list.
32. (True|False) You can remove an element from a tuple by calling the tuple’s remove() method.
33. Look at the following statement:

numbers = [10, 20, 30, 40, 50]

1. How many elements does the list have?5
2. What is the index of the first element in the list?0
3. What is the index of the last element in the list?4 or -1
4. Look at the following statement:

letters = ['A', 'B', 'C', 'D']

1. What value is stored in letters[1]?b
2. What value is stored in letters[3]?d
3. What value is stored in letters[−2]?c
4. What will the following code display?

values = [2, 4, 6, 8, 10]

print(values[1:3])

1. What does the following code display?

numbers = [5, 4, 3, 2, 1, 0]

print(numbers[:3])

1. What does the following code display?

numbers = [1, 2, 3, 4, 5, 6, 7, 8]

print(numbers[−4:])

1. What does the following code display?

values = [2] \* 5

print(values)

## Chapter 8 – More About Strings

1. This is the first index in a string.
2. −1
3. 1
4. 0
5. The size of the string minus one
6. This is the last index in a string.
7. 1
8. 99
9. 0
10. The size of the string minus one
11. This will happen if you try to use an index that is out of range for a string.
12. A ValueError exception will occur.
13. An IndexError exception will occur.
14. The string will be erased, and the program will continue to run.
15. Nothing—the invalid index will be ignored.
16. This function returns the length of a string.
17. length()
18. size()
19. len()
20. lengthof()
21. This string method returns a copy of the string with all leading whitespace characters removed.
22. lstrip()
23. rstrip()
24. remove()
25. strip\_leading()
26. This string method returns the lowest index in the string where a specified substring is found.
27. first\_index\_of()
28. locate()
29. find()
30. index\_of()
31. This operator determines whether one string is contained inside another string.
32. contains
33. is\_in
34. ==
35. in
36. This string method returns true if a string contains only alphabetic characters and is at least one character in length.
37. isalpha() method
38. alpha() method
39. alphabetic() method
40. isletters() method
41. This string method returns true if a string contains only numeric digits and is at least one character in length.
42. digit() method
43. isdigit() method
44. numeric() method
45. isnumber() method
46. This string method returns a copy of the string with all leading and trailing whitespace characters removed.
47. clean()
48. strip()
49. remove\_whitespace()
50. rstrip()
51. (True|False) Once a string is created, it cannot be changed.
52. (True|False) You can use the for loop to iterate over the individual characters in a string.
53. (True|False) The isupper() method converts a string to all uppercase characters.
54. (True|False) The repetition operator (\*) works with strings as well as with lists.
55. (True|False) When you call a string’s split() method, the method divides the string into two substrings.
56. What does the following code display?

mystr = 'abc'

mystr2 = '123'

mystr += mystr2

print(mystr) abc123

1. What does the following code display?

mystr = 'abc' \* 3

print(mystr)abcabcabc

1. What will the following code display?

mystr = 'abracadabra'

print(mystr[6:9])dab

1. What does the following code display?

numbers = [1, 2, 3, 4, 5, 6, 7]

print(numbers[4:6])

1. What does the following code display?

name = 'joe'

print(name.lower())

print(name.upper())

print(name)joe

## Chapter 9 – Dictionaries and Sets

1. You can use the \_\_\_\_\_\_\_\_\_ operator to determine whether a key exists in a dictionary.
2. &
3. in
4. ˆ
5. ?
6. You use \_\_\_\_\_\_\_\_\_ to delete an element from a dictionary.
7. remove() method
8. erase() method
9. delete() method
10. del statement
11. The \_\_\_\_\_\_\_\_\_ function returns the number of elements in a dictionary:
12. size()
13. len()
14. elements()
15. count()
16. You can use \_\_\_\_\_\_\_\_\_ to create an empty dictionary.
17. {} or dict
18. ()
19. []
20. empty()
21. The \_\_\_\_\_\_\_\_\_ method returns a randomly selected key-value pair from a dictionary.
22. pop()
23. random()
24. popitem()
25. rand\_pop()
26. The \_\_\_\_\_\_\_\_\_ method returns the value associated with a specified key and removes that key-value pair from the dictionary.
27. pop()
28. random()
29. popitem()
30. rand\_pop()
31. The \_\_\_\_\_\_\_\_\_ dictionary method returns the value associated with a specified key. If the key is not found, it returns a default value.
32. pop()
33. key()
34. value()
35. get()
36. The \_\_\_\_\_\_\_\_\_ method returns all of a dictionary’s keys and their associated values as a sequence of tuples.
37. keys\_values()
38. values()
39. items()
40. get()
41. The following function returns the number of elements in a set:
42. size()
43. len()
44. elements()
45. count()
46. You can add one element to a set with this method.
47. append()
48. add()
49. update()
50. merge()
51. You can add a group of elements to a set with this method.
52. append()
53. add()
54. update()
55. merge()
56. This set method removes an element but does not raise an exception if the element is not found.
57. remove()
58. discard()
59. delete()
60. erase()
61. This set method removes an element and raises an exception if the element is not found.
62. remove()
63. discard()
64. delete()
65. erase()
66. This operator can be used to find the union of two sets.
67. |
68. &
69. −
70. ˆ
71. This operator can be used to find the difference of two sets.
72. |
73. &
74. −
75. ˆ
76. This operator can be used to find the intersection of two sets.
77. |
78. &
79. −
80. ˆ
81. This operator can be used to find the symmetric difference of two sets.
82. |
83. &
84. −
85. ˆ
86. (True|False) The keys in a dictionary must be mutable objects.
87. (True|False) Dictionaries are not sequences.
88. (True|False) A tuple can be a dictionary key.
89. (True|False) A list can be a dictionary key.
90. (True|False) The dictionary method popitem() does not raise an exception if it is called on an empty dictionary.
91. (True|False) The following statement creates an empty dictionary: mydct = {}
92. (True|False) The following statement creates an empty set: myset = ()
93. (True|False) Sets store their elements in an unordered fashion.
94. (True|False) You can store duplicate elements in a set.
95. (True|False) The remove() method raises an exception if the specified element is not found in the set.
96. What will the following code display?

dct = {'Monday':1, 'Tuesday':2, 'Wednesday':3}

print(dct['Tuesday'])

1. What will the following code display?

dct = {'Monday':1, 'Tuesday':2, 'Wednesday':3}

print(dct.get('Monday', 'Not found'))

1. What will the following code display?

dct = {'Monday':1, 'Tuesday':2, 'Wednesday':3}

print(dct.get('Friday', 'Not found'))

1. What will occur when the following code is executed?

months = {'Jan':1, 'Feb':2, 'Mar':3}

months[2]

1. How do you delete an element from a dictionary?
2. How do you determine the number of elements that are stored in a dictionary?
3. What will the following code display?

dct = {0:[1, 2, [3, 4], 5]}

print(dct[0][2][1])

1. What values will the following code display? (Don’t worry about the order in which they will be displayed.)

dct = {1:[0, 1], 2:[2, 3], 3:[4, 5]}

for k in dct:

print(k)

1. After the following statement executes, what elements will be stored in the myset set?

myset = set('Saturn')

1. After the following statement executes, what elements will be stored in the myset set?

myset = set(10)

1. After the following statement executes, what elements will be stored in the myset set?

myset = set('a bb ccc dddd')

1. After the following statement executes, what elements will be stored in the myset set?

myset = set([2, 4, 4, 6, 6, 6, 6])

1. After the following statement executes, what elements will be stored in the myset set?

myset = set(['a', 'bb', 'ccc', 'dddd'])

1. What will the following code display?

myset = set('1 2 3')

print(len(myset))

1. After the following code executes, what elements will be members of set3?

set1 = set([10, 20, 30, 40])

set2 = set([40, 50, 60])

set3 = set1.union(set2)

1. After the following code executes, what elements will be members of set3?

set1 = set(['o', 'p', 's', 'v'])

set2 = set(['a', 'p', 'r', 's'])

set3 = set1.intersection(set2)

1. After the following code executes, what elements will be members of set3?

set1 = set(['d', 'e', 'f'])

set2 = set(['a', 'b', 'c', 'd', 'e'])

set3 = set1.difference(set2)

1. After the following code executes, what elements will be members of set3?

set1 = set(['d', 'e', 'f'])

set2 = set(['a', 'b', 'c', 'd', 'e'])

set3 = set2.difference(set1)

1. After the following code executes, what elements will be members of set3?

set1 = set([1, 2, 3])

set2 = set([2, 3, 4])

set3 = set1.symmetric\_difference(set2)

1. Look at the following code:

set1 = set('abc')

set2 = set('def')

What would be the result of calculating the intersection of set1 and set2?

Which other set operation would produce the same result as the union of set1 and set2?

## Chapter 10 – Classes and Object-Oriented Programming

1. The \_\_\_\_\_\_\_\_\_\_ programming practice is centered on creating functions that are separate from the data that they work on.
2. modular
3. procedural
4. functional
5. object-oriented
6. The \_\_\_\_\_\_\_\_\_\_ programming practice is centered on creating objects.
7. object-centric
8. objective
9. procedural
10. object-oriented
11. A(n) \_\_\_\_\_\_\_\_\_\_ is a component of a class that references data.
12. method
13. instance
14. data attribute
15. module
16. An object is a(n) \_\_\_\_\_\_\_\_\_\_.
17. blueprint
18. cookie cutter
19. variable
20. instance
21. By doing this, you can hide a class’s attribute from code outside the class.
22. avoid using the self parameter to create the attribute
23. begin the attribute’s name with two underscores
24. begin the name of the attribute with private\_\_
25. begin the name of the attribute with the @ symbol
26. A(n) \_\_\_\_\_\_\_\_\_\_ method gets the value of a data attribute but does not change it.
27. retriever
28. constructor
29. mutator
30. accessor
31. A(n) \_\_\_\_\_\_\_\_\_\_ method stores a value in a data attribute or changes its value in some other way.
32. modifier
33. constructor
34. mutator
35. accessor
36. The \_\_\_\_\_\_\_\_\_\_ method is automatically called when an object is created.
37. \_\_init\_\_
38. init
39. \_\_str\_\_
40. \_\_object\_\_
41. If a class has a method named \_\_str\_\_(), which of these is a way to call the method?
42. you call it like any other method: *object*.\_\_str\_\_()
43. by passing an instance of the class to the built in str() function
44. the method is automatically called when the object is created
45. by passing an instance of the class to the built-in state() function
46. A set of standard diagrams for graphically depicting object-oriented systems is provided by \_\_\_\_\_\_\_\_\_\_.
47. the Unified Modeling Language
48. flowcharts
49. pseudocode
50. the Object Hierarchy System
51. In one approach to identifying the classes in a problem, the programmer identifies the \_\_\_\_\_\_\_\_\_\_ in a description of the problem domain.
52. verbs
53. adjectives
54. adverbs
55. nouns
56. In one approach to identifying a class’s data attributes and methods, the programmer identifies the class’s \_\_\_\_\_\_\_\_\_\_.
57. responsibilities
58. name
59. synonyms
60. nouns
61. (True|False) The practice of procedural programming is centered on the creation of objects.
62. (True|False) Object reusability has been a factor in the increased use of object-oriented programming.
63. (True|False) It is a common practice in object-oriented programming to make all of a class’s data attributes accessible to statements outside the class.
64. (True|False) A class method does not have to have a self parameter.
65. (True|False) Starting an attribute name with two underscores will hide the attribute from code outside the class.
66. (True|False) You cannot directly call the \_\_str\_\_() method.
67. (True|False) One way to find the classes needed for an object-oriented program is to identify all of the verbs in a description of the problem domain.
68. What is meant by data hiding?
69. How are an object’s data attributes made inaccessible to code outside the class definition?
70. What is the difference between a class and an instance of a class?
71. The following statement calls an object’s method. What is the name of the method? What is the name of the variable that references the object?

wallet.get\_dollar()

1. When the \_\_init\_\_() method executes, what does the self parameter reference?
2. In a Python class, how do you hide an attribute from code outside the class?
3. The \_\_str\_\_() method in Python returns a string representation of an object’s state. Describe how the \_\_str\_\_() method is used to print the state of an object.