Python (Programming Language)

Q1. What is an abstract class?

- [ ] An abstract class is the name for any class from which you can instantiate an object.
- [ ] Abstract classes must be redefined any time an object is instantiated from them.
- [ ] Abstract classes must inherit from concrete classes.
- [x] An abstract class exists only so that other "concrete" classes can inherit from the abstract class.

Q2. What happens when you use the build-in function `any()` on a list?

- [ ] The `any()` function will randomly return any item from the list.
- [x] The `any()` function returns True if any item in the list evaluates to True. Otherwise, it returns False.
- [ ] The `any()` function takes as arguments the list to check inside, and the item to check for. If "any" of the items in the list match the item to check for, the function returns True.
- [ ] The `any()` function returns a Boolean value that answers the question "Are there any items in this list?"

Q3. What data structure does a binary tree degenerate to if it isn't balanced properly?

- [x] linked list
- [ ] queue
- [ ] set
- [ ] OrderedDict

Q4. What statement about static methods is true?

- [ ] Static methods are called static because they always return `None`.
- [ ] Static methods can be bound to either a class or an instance of a class.
- [x] Static methods serve mostly as utility methods or helper methods, since they can't access or modify a class's state.
- [ ] Static methods can access and modify the state of a class or an instance of a class.

Q5. What are attributes?

- [ ] Attributes are long-form version of an `if/else` statement, used when testing for equality between objects.
- [x] Attributes are a way to hold data or describe a state for a class or an instance of a class.
- [ ] Attributes are strings that describe characteristics of a class.
- [ ] Function arguments are called "attributes" in the context of class methods and instance methods.

Q6. What is the term to describe this code?

```
count, fruit, price = (2, 'apple', 3.5)
```

- [ ] `tuple assignment`
- [x] `tuple unpacking`
- [ ] `tuple matching`
- [ ] `tuple duplication`

Q7. What built-in list method would you use to remove items from a list?

- [ ] `.delete()` method
- [ ] `pop(my_list)`
- [ ] `del(my_list)`
- [x] `.pop()` method

Q8. What is one of the most common use of Python's sys library?

- [x] to capture command-line arguments given at a file's runtime
- [ ] to connect various systems, such as connecting a web front end, an API service, a database, and a mobile app
- [ ] to take a snapshot of all the packages and libraries in your virtual environment
- [ ] to scan the health of your Python ecosystem while inside a virtual environment

Q9. What is the runtime of accessing a value in a dictionary by using its key?

- [ ] O(n), also called linear time.
- [ ] O(log n), also called logarithmic time.
- [ ] O(n^2), also called quadratic time.
- [x] O(1), also called constant time.

Q10. What is the correct syntax for defining a class called Game?

- [x] `class Game: pass`
- [ ] `def Game(): pass`
- [ ] `def Game: pass`
- [ ] `class Game(): pass`

Q11. What is the correct way to write a doctest?

- [] A

```
def sum(a, b):
    """
    sum(4, 3)
    7

    sum(-4, 5)
    1
    """
    return a + b
```

- [x] B

```
def sum(a, b):
    """
    >>> sum(4, 3)
    7

    >>> sum(-4, 5)
    1
    """
    return a + b
```

- [] C

```
def sum(a, b):
    """
    # >>> sum(4, 3)
    # 7

    # >>> sum(-4, 5)
    # 1
    """
    return a + b
```

- [] D

```
def sum(a, b):
    ###
    >>> sum(4, 3)
    7

    >>> sum(-4, 5)
    1
    ###
    return a + b
```

Q12. What built-in Python data type is commonly used to represent a stack?

- [] `set`
- [x] `list`
- [] `None`
- [] `dictionary`

`. You can only build a stack from scratch.`

Q13. What would this expression return?

```
college_years = ['Freshman', 'Sophomore', 'Junior', 'Senior']
return list(enumerate(college_years, 2019))
```

- [] `[('Freshman', 2019), ('Sophomore', 2020), ('Junior', 2021), ('Senior', 2022)]`

- [] `[(2019, 2020, 2021, 2022), ('Freshman', 'Sophomore', 'Junior', 'Senior')]`
- [] `[('Freshman', 'Sophomore', 'Junior', 'Senior'), (2019, 2020, 2021, 2022)]`
- [x] `[(2019, 'Freshman'), (2020, 'Sophomore'), (2021, 'Junior'), (2022, 'Senior')]`

Q14. How does `defaultdict` work?

- [] `defaultdict` will automatically create a dictionary for you that has keys which are the integers 0-10.
- [] `defaultdict` forces a dictionary to only accept keys that are of the types specified when you created the `defaultdict` (such as string or integers).
- [x] If you try to access a key in a dictionary that doesn't exist, `defaultdict` will create a new key for you instead of throwing a `KeyError`.
- [] `defaultdict` stores a copy of a dictionary in memory that you can default to if the original gets unintentionally modified.

Q15. What is the correct syntax for defining a class called "Game", if it inherits from a parent class called "LogicGame"?

- [] `class Game.LogicGame(): pass`
- [] `def Game(LogicGame): pass`
- [x] `class Game(LogicGame): pass`
- [] `def Game.LogicGame(): pass`

Q16. What is the purpose of the "self" keyword when defining or calling instance methods?

- [] `self` means that no other arguments are required to be passed into the method.
- [] There is no real purpose for the `self` method; it's just historic computer science jargon that Python keeps to stay consistent with other programming languages.
- [x] `self` refers to the instance whose method was called.
- [] `self` refers to the class that was inherited from to create the object using `self`.

Q17. Which of these is NOT a characteristic of namedtuples?

- [] You can assign a name to each of the `namedtuple` members and refer to them that way, similarly to how you would access keys in `dictionary`.
- [] Each member of a namedtuple object can be indexed to directly, just like in a regular `tuple`.
- [] `namedtuples` are just as memory efficient as regular `tuples`.
- [x] No import is needed to use `namedtuples` because they are available in the standard library.

Q18. What is an instance method?

- [x] Instance methods can modify the state of an instance or the state of its parent class.
- [] Instance methods hold data related to the instance.
- [] An instance method is any class method that doesn't take any arguments.
- [] An instance method is a regular function that belongs to a class, but it must return `None`.

Q19. Which choice is the most syntactically correct example of the conditional branching?

- []

```
num_people = 5

if num_people > 10:
    print("There is a lot of people in the pool.")
elif num_people > 4;
    print("There are some people in the pool.")
elif num_people > 0;
    print("There are a few people in the pool.")
else:
    print("There is no one in the pool.")
```

- []

```
num_people = 5

if num_people > 10:
    print("There is a lot of people in the pool.")
if num_people > 4:
    print("There are some people in the pool.")
if num_people > 0:
    print("There are a few people in the pool.")
else:
    print("There is no one in the pool.")
```

- [x]

```
num_people = 5

if num_people > 10:
    print("There is a lot of people in the pool.")
elif num_people > 4:
    print("There are some people in the pool.")
elif num_people > 0:
    print("There are a few people in the pool.")
else:
    print("There is no one in the pool.")
```

- [ ]

```
if num_people > 10;
    print("There is a lot of people in the pool.")
if num_people > 4:
    print("There are some people in the pool.")
if num_people > 0:
    print("There are a few people in the pool.")
else:
    print("There is no one in the pool.")
```

Also see Question 85 for the same question with different answers.

Q20. Which statement does NOT describe the object-oriented programming concept of encapsulation?

- [ ] It protects the data from outside interference.
- [ ] A parent class is encapsulated and no data from the parent class passes on to the child class.
- [ ] It keeps data and the methods that can manipulate that data in one place.
- [x] It only allows the data to be changed by methods.

Q21. What is the purpose of an if/else statement?

- [ ] It tells the computer which chunk of code to run if the instructions you coded are incorrect.
- [ ] It runs one chunk of code if all the imports were successful, and another chunk of code if the imports were not successful.
- [x] It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false.
- [ ] It tells the computer which chunk of code to run if the is enough memory to handle it, and which chunk of code to run if there is not enough memory to handle it.

Q22. What built-in Python data type is commonly used to represent a queue?

- [ ] `dictionary`
- [ ] `set`
- [ ] `None`
- [x] `list`

`You can only build a stack from scratch.`

Q23. What is the correct syntax for instantiating a new object of the type Game?

- [ ] `my_game = class.Game()`
- [ ] `my_game = class(Game)`
- [x] `my_game = Game()`
- [ ] `my_game = Game.create()`

Q24. What does the built-in `map()` function do?

- [ ] It creates a path from multiple values in an iterable to a single value.
- [x] It applies a function to each item in an iterable and returns the value of that function.
- [ ] It converts a complex value type into simpler value types.
- [ ] It creates a mapping between two different elements of different iterables.

Q25. If you don't explicitly return a value from a function, what happens?

- [ ] The function will return a RuntimeError if you don't return a value.
- [x] If the return keyword is absent, the function will return `None`.
- [ ] If the return keyword is absent, the function will return `True`.
- [ ] The function will enter an infinite loop because it won't know when to stop executing its code.

Q26. What is the purpose of the `pass` statement in Python?

- [] It is used to skip the `yield` statement of a generator and return a value of None.
- [x] It is a null operation used mainly as a placeholder in functions, classes, etc.
- [] It is used to pass control from one statement block to another.
- [] It is used to skip the rest of a `while` or `for loop` and return to the start of the loop.

Q27. What is the term used to describe items that may be passed into a function?

- [x] arguments
- [] paradigms
- [] attributes
- [] decorators

Q28. Which collection type is used to associate values with unique keys?

- [] `slot`
- [x] `dictionary`
- [] `queue`
- [] `sorted list`

Q29. When does a for loop stop iterating?

- [] when it encounters an infinite loop
- [] when it encounters an if/else statement that contains a break keyword
- [x] when it has assessed each item in the iterable it is working on or a break keyword is encountered
- [] when the runtime for the loop exceeds O(n^2)

Q30. Assuming the node is in a singly linked list, what is the runtime complexity of searching for a specific node within a singly linked list?

- [x] The runtime is O(n) because in the worst case, the node you are searching for is the last node, and every node in the linked list must be visited.
- [] The runtime is O(nk), with n representing the number of nodes and k representing the amount of time it takes to access each node in memory.
- [] The runtime cannot be determined unless you know how many nodes are in the singly linked list.
- [] The runtime is O(1) because you can index directly to a node in a singly linked list.

Q31. Given the following three list, how would you create a new list that matches the desired output printed below?

```
fruits = ['Apples', 'Oranges', 'Bananas']
quantities = [5, 3, 4]
prices = [1.50, 2.25, 0.89]

#Desired output
[('Apples', 5, 1.50),
('Oranges', 3, 2.25),
('Bananas', 4, 0.89)]
```

- []

```
output = []

fruit_tuple_0 = (first[0], quantities[0], price[0])
output.append(fruit_tuple)

fruit_tuple_1 = (first[1], quantities[1], price[1])
output.append(fruit_tuple)

fruit_tuple_2 = (first[2], quantities[2], price[2])
output.append(fruit_tuple)

return output
```

- [x]

```
i = 0
output = []
for fruit in fruits:
    temp_qty = quantities[i]
    temp_price = prices[i]
    output.append((fruit, temp_qty, temp_price))
    i += 1
return output
```

- []

```
groceries = zip(fruits, quantities, prices)
return groceries

>>> [
('Apples', 5, 1.50),
('Oranges', 3, 2.25),
('Bananas', 4, 0.89)
]
```

- []

```
i = 0
output = []
for fruit in fruits:
    for qty in quantities:
        for price in prices:
            output.append((fruit, qty, price))
    i += 1
return output
```

Q32. What happens when you use the built-in function all() on a list?

- [] The `all()` function returns a Boolean value that answers the question "Are all the items in this list the same?
- [] The `all()` function returns True if all the items in the list can be converted to strings. Otherwise, it returns False.
- [] The `all()` function will return all the values in the list.
- [x] The `all()` function returns True if all items in the list evaluate to True. Otherwise, it returns False.

Q33. What is the correct syntax for calling an instance method on a class named Game?

(Answer format may vary. Game and roll (or dice_roll) should each be called with no parameters.)

- [x]

```
>>> dice = Game()
>>> dice.roll()
```

- []

```
>>> dice = Game(self)
>>> dice.roll(self)
```

- []

```
>>> dice = Game()
>>> dice.roll(self)
```

- []

```
>>> dice = Game(self)
>>> dice.roll()
```

Q34. What is the algorithmic paradigm of quick sort?

- [] backtracking
- [] dynamic programming
- [] decrease and conquer
- [x] divide and conquer

Q35. What is runtime complexity of the list's built-in `.append()` method?

- [x] O(1), also called constant time
- [] O(log n), also called logarithmic time
- [] O(n^2), also called quadratic time
- [] O(n), also called linear time

Q36. What is key difference between a `set` and a `list`?

- [ ] A set is an ordered collection unique items. A list is an unordered collection of non-unique items.
- [ ] Elements can be retrieved from a list but they cannot be retrieved from a set.
- [ ] A set is an ordered collection of non-unique items. A list is an unordered collection of unique items.
- [x] A set is an unordered collection unique items. A list is an ordered collection of non-unique items.

Q37. What is the definition of abstraction as applied to object-oriented Python?

- [ ] Abstraction means that a different style of code can be used, since many details are already known to the program behind the scenes.
- [x] Abstraction means the implementation is hidden from the user, and only the relevant data or information is shown.
- [ ] Abstraction means that the data and the functionality of a class are combined into one entity.
- [ ] Abstraction means that a class can inherit from more than one parent class.

Q38. What does this function print?

```
def print_alpha_nums(abc_list, num_list):
    for char in abc_list:
        for num in num_list:
            print(char, num)
    return

print_alpha_nums(['a', 'b', 'c'], [1, 2, 3])
```

- [x]

```
a 1
a 2
a 3
b 1
b 2
b 3
c 1
c 2
c 3
```

- [ ]

```
['a', 'b', 'c'], [1, 2, 3]
```

- [ ]

```
aaa
bbb
ccc
111
222
333
```

- [ ]

```
a 1 2 3
b 1 2 3
c 1 2 3
```

Q39. Correct representation of doctest for function in Python

- [ ]

```
def sum(a, b):
    # a = 1
    # b = 2
    # sum(a, b) = 3

    return a + b
```

- []

```
def sum(a, b):
    """
    a = 1
    b = 2
    sum(a, b) = 3
    """

    return a + b
```

- [x]

```
def sum(a, b):
    """
    >>> a = 1
    >>> b = 2
    >>> sum(a, b)
    3
    """

    return a + b
```

- []

```
def sum(a, b):
    '''
    a = 1
    b = 2
    sum(a, b) = 3
    '''
    return a + b
```

Q40. Suppose a Game class inherits from two parent classes: BoardGame and LogicGame. Which statement is true about the methods of an object instantiated from the Game class?

- [] When instantiating an object, the object doesn't inherit any of the parent class's methods.
- [] When instantiating an object, the object will inherit the methods of whichever parent class has more methods.
- [] When instantiating an object, the programmer must specify which parent class to inherit methods from.
- [x] An instance of the Game class will inherit whatever methods the BoardGame and LogicGame classes have.

Q41. What does calling namedtuple on a collection type return?

- [] a generic object class with iterable parameter fields
- [] a generic object class with non-iterable named fields
- [] a tuple subclass with non-iterable parameter fields
- [x] a tuple subclass with iterable named fields

Q42. What symbol(s) do you use to assess equality between two elements?

- [] `&&`
- [] `=`
- [x] `==`
- [] `||`

Q43. Review the code below. What is the correct syntax for changing the price to 1.5?

```
fruit_info = {
  'fruit': 'apple',
  'count': 2,
  'price': 3.5
}
```

- [x] `fruit_info ['price'] = 1.5`
- [] `my_list [3.5] = 1.5`
- [] `1.5 = fruit_info ['price]`
- [] `my_list['price'] == 1.5`

Q44. What value would be returned by this check for equality?

`5 != 6`

- [] `yes`
- [] `False`
- [x] `True`
- [] `None`

Q45. What does a class's `init()` method do?

- [] The `__init__` method makes classes aware of each other if more than one class is defined in a single code file.
- [] The `__init__` method is included to preserve backwards compatibility from Python 3 to Python 2, but no longer needs to be used in Python 3.
- [x] The `__init__` method is a constructor method that is called automatically whenever a new object is created from a class. It sets the initial state of a new object.
- [] The `__init__` method initializes any imports you may have included at the top of your file.

Q46. What is meant by the phrase "space complexity"?

- [] `How many microprocessors it would take to run your code in less than one second`
- [] `How many lines of code are in your code file`
- [x] `The amount of space taken up in memory as a function of the input size`
- [] `How many copies of the code file could fit in 1 GB of memory`

Q47. What is the correct syntax for creating a variable that is bound to a dictionary?

- [x] `fruit_info = {'fruit': 'apple', 'count': 2, 'price': 3.5}`
- [] `fruit_info =('fruit': 'apple', 'count': 2,'price': 3.5 ).dict()`
- [] `fruit_info = ['fruit': 'apple', 'count': 2,'price': 3.5 ].dict()`
- [] `fruit_info = to_dict('fruit': 'apple', 'count': 2, 'price': 3.5)`

Q48. What is the proper way to write a list comprehension that represents all the keys in this dictionary?

`fruits = {'Apples': 5, 'Oranges': 3, 'Bananas': 4}`

- [] `fruit_names = [x in fruits.keys() for x]`
- [] `fruit_names = for x in fruits.keys() *`
- [x] `fruit_names = [x for x in fruits.keys()]`
- [] `fruit_names = x for x in fruits.keys()`

Q49. What is the purpose of the `self` keyword when defining or calling methods on an instance of an object?

- [] `self` refers to the class that was inherited from to create the object using `self`.
- [] There is no real purpose for the `self` method. It's just legacy computer science jargon that Python keeps to stay consistent with other programming languages.
- [] `self` means that no other arguments are required to be passed into the method.
- [x] `self` refers to the instance whose method was called.

Q50. What statement about the class methods is true?

- [] A class method is a regular function that belongs to a class, but it must return None.
- [x] A class method can modify the state of the class, but they can't directly modify the state of an instance that inherits from that class.
- [] A class method is similar to a regular function, but a class method doesn't take any arguments.
- [] A class method hold all of the data for a particular class.

Q51. What does it mean for a function to have linear runtime?

- [] You did not use very many advanced computer programming concepts in your code.
- [] The difficulty level your code is written at is not that high.
- [] It will take your program less than half a second to run.
- [x] The amount of time it takes the function to complete grows linearly as the input size increases.

Q52. What is the proper way to define a function?

- [] `def getMaxNum(list_of_nums): # body of function goes here`
- [] `func get_max_num(list_of_nums): # body of function goes here`
- [] `func getMaxNum(list_of_nums): # body of function goes here`
- [x] `def get_max_num(list_of_nums): # body of function goes here`

explanation

Q53. According to the PEP 8 coding style guidelines, how should constant values be named in Python?

- [] in camel case without using underscores to separate words -- e.g. `maxValue = 255`
- [] in lowercase with underscores to separate words -- e.g. `max_value = 255`

- [x] in all caps with underscores separating words -- e.g. `MAX_VALUE = 255`
- [ ] in mixed case without using underscores to separate words -- e.g. `MaxValue = 255`

Q54. Describe the functionality of a deque.

- [ ] A deque adds items to one side and remove items from the other side.
- [ ] A deque adds items to either or both sides, but only removes items from the top.
- [x] A deque adds items at either or both ends, and remove items at either or both ends.
- [ ] A deque adds items only to the top, but remove from either or both sides.

Q55. What is the correct syntax for creating a variable that is bound to a set?

- [x] `myset = {0, 'apple', 3.5}`
- [] `myset = to_set(0, 'apple', 3.5)`
- [] `myset = (0, 'apple', 3.5).to_set()`
- [] `myset = (0, 'apple', 3.5).set()`

Q56. What is the correct syntax for defining an `__init__()` method that takes no parameters?

- []

```
class __init__(self):
    pass
```

- []

```
def __init__():
    pass
```

- []

```
class __init__():
    pass
```

- [x]

```
def __init__(self):
    pass
```

Q57. Which of the following is TRUE About how numeric data would be organised in a binary Search tree?

- [x] For any given Node in a binary Search Tree, the child node to the left is less than the value of the given node and the child node to its right is greater than the given node.
- [ ] Binary Search Tree cannot be used to organize and search through numeric data, given the complication that arise with very deep trees.
- [ ] The top node of the binary search tree would be an arbitrary number. All the nodes to the left of the top node need to be less than the top node's number, but they don't need to ordered in any particular way.
- [ ] The smallest numeric value would go in the top most node. The next highest number would go in its left child node, the the next highest number after that would go in its right child node. This pattern would continue until all numeric values were in their own node.

Q58. Why would you use a decorator?

- [ ] A decorator is similar to a class and should be used if you are doing functional programming instead of object oriented programming.
- [ ] A decorator is a visual indicator to someone reading your code that a portion of your code is critical and should not be changed.
- [x] You use the decorator to alter the functionality of a function without having to modify the functions code.
- [ ] An import statement is preceded by a decorator, python knows to import the most recent version of whatever package or library is being imported.

Q59. When would you use a for loop?

- [ ] Only in some situations, as loops are used only for certain type of programming.
- [x] When you need to check every element in an iterable of known length.
- [ ] When you want to minimize the use of strings in your code.
- [ ] When you want to run code in one file for a function in another file.

Q60. What is the most self-descriptive way to define a function that calculates sales tax on a purchase?

- [ ]

```
def tax(my_float):
    '''Calculates the sales tax of a purchase. Takes in a float representing the subtotal as an argument and returns a float represe
    pass
```

- []

```
def tx(amt):
    '''Gets the tax on an amount.'''
```

- []

```
def sales_tax(amount):
    '''Calculates the sales tax of a purchase. Takes in a float representing the subtotal as an argument and returns a float represe
```

- [x]

```
def calculate_sales_tax(subtotal):
    pass
```

Q61. What would happen if you did not alter the state of the element that an algorithm is operating on recursively?

- [] You do not have to alter the state of the element the algorithm is recursing on.
- [] You would eventually get a KeyError when the recursive portion of the code ran out of items to recurse on.
- [x] You would get a RuntimeError: maximum recursion depth exceeded.
- [] The function using recursion would return None.

Q62. What is the runtime complexity of searching for an item in a binary search tree?

- [] The runtime for searching in a binary search tree is O(1) because each node acts as a key, similar to a dictionary.
- [] The runtime for searching in a binary search tree is O(n!) because every node must be compared to every other node.
- [x] The runtime for searching in a binary search tree is generally O(h), where h is the height of the tree.
- [] The runtime for searching in a binary search tree is O(n) because every node in the tree must be visited.

Q63. Why would you use `mixin`?

- [] You use a `mixin` to force a function to accept an argument at runtime even if the argument wasn't included in the function's definition.
- [] You use a `mixin` to allow a decorator to accept keyword arguments.
- [] You use a `mixin` to make sure that a class's attributes and methods don't interfere with global variables and functions.
- [x] If you have many classes that all need to have the same functionality, you'd use a `mixin` to define that functionality.

Q64. What is the runtime complexity of adding an item to a stack and removing an item from a stack?

- [] Add items to a stack in O(1) time and remove items from a stack on O(n) time.
- [x] Add items to a stack in O(1) time and remove items from a stack in O(1) time.
- [] Add items to a stack in O(n) time and remove items from a stack on O(1) time.
- [] Add items to a stack in O(n) time and remove items from a stack on O(n) time.

Q65. Which statement accurately describes how items are added to and removed from a stack?

- [] a stacks adds items to one side and removes items from the other side.
- [x] a stacks adds items to the top and removes items from the top.
- [] a stacks adds items to the top and removes items from anywhere in the stack.
- [] a stacks adds items to either end and removes items from either end.

Q66. What is a base case in a recursive function?

- [x] A base case is the condition that allows the algorithm to stop recursing. It is usually a problem that is small enough to solve directly.
- [] The base case is summary of the overall problem that needs to be solved.
- [] The base case is passed in as an argument to a function whose body makes use of recursion.
- [] The base case is similar to a base class, in that it can be inherited by another object.

Q67. Why is it considered good practice to open a file from within a Python script by using the `with` keyword?

- [] The `with` keyword lets you choose which application to open the file in.
- [] The `with` keyword acts like a `for` loop, and lets you access each line in the file one by one.
- [] There is no benefit to using the `with` keyword for opening a file in Python.
- [x] When you open a file using the `with` keyword in Python, Python will make sure the file gets closed, even if an exception or error is thrown.

Q68. Why would you use a virtual environment?

- [x] Virtual environments create a "bubble" around your project so that any libraries or packages you install within it don't affect your entire machine.
- [ ] Teams with remote employees use virtual environments so they can share code, do code reviews, and collaborate remotely.
- [ ] Virtual environments were common in Python 2 because they augmented missing features in the language. Virtual environments are not necessary in Python 3 due to advancements in the language.
- [ ] Virtual environments are tied to your GitHub or Bitbucket account, allowing you to access any of your repos virtually from any machine.

Q69. What is the correct way to run all the doctests in a given file from the command line?

- [ ] python3 -m doctest <_filename_>
- [x] python3 <_filename_>
- [ ] python3 <_filename_> rundoctests
- [ ] python3 doctest

Q70. What is a lambda function ?

- [ ] any function that makes use of scientific or mathematical constants, often represented by Greek letters in academic writing
- [ ] a function that get executed when decorators are used
- [ ] any function whose definition is contained within five lines of code or fewer
- [x] a small, anonymous function that can take any number of arguments but has only expression to evaluate

Reference

Explanation: `the lambda notation is basically an anonymous function that can take any number of arguments with only single expression (i.e, cannot be overloaded). It has been introducted in other programming languages, such as C++ and Java. The lambda notation allows programmers to "bypass" function declaration.`

Q71. What is the primary difference between lists and tuples?

- [ ] You can access a specifc element in a list by indexing to its position, but you cannot access a specific element in a tuple unless you iterate through the tuple
- [x] Lists are mutable, meaning you can change the data that is inside them at any time. Tuples are immutable, meaning you cannot change the data that is inside them once you have created the tuple.
- [ ] Lists are immutable, meaning you cannot change the data that is inside them once you have created the list. Tuples are mutable, meaning you can change the data that is inside them at any time.
- [ ] Lists can hold several data types inside them at once, but tuples can only hold the same data type if multiple elements are present.

Q72. Which statement about static method is true?

- [ ] Static methods can be bound to either a class or an instance of a class.
- [ ] Static methods can access and modify the state of a class or an instance of a class.
- [x] Static methods serve mostly as utility or helper methods, since they cannot access or modify a class's state.
- [ ] Static methods are called static because they always return None.

Q73. What does a generator return?

- [ ] None
- [x] An iterable object
- [ ] A linked list data structure from a non-empty list
- [ ] All the keys of the given dictionary

Q74. What is the difference between class attributes and instance attributes?

- [ ] Instance attributes can be changed, but class attributes cannot be changed
- [x] Class attributes are shared by all instances of the class. Instance attributes may be unique to just that instance
- [ ] There is no difference between class attributes and instance attributes
- [ ] Class attributes belong just to the class, not to instance of that class. Instance attributes are shared among all instances of a class

Q75. What is the correct syntax of creating an instance method?

- [ ]

```
def get_next_card():
  # method body goes here
```

- [x]

```
def get_next_card(self):
  # method body goes here
```

- [ ]

```
def self.get_next_card():
  # method body goes here
```

- [ ]

```
def self.get_next_card(self):
  # method body goes here
```

Q76. What is the correct way to call a function?

- [x] get_max_num([57, 99, 31, 18])
- [ ] call.(get_max_num)
- [ ] def get_max_num([57, 99, 31, 18])
- [ ] call.get_max_num([57, 99, 31, 18])

Q77. How is comment created?

- [] `-- This is a comment`
- [x] `# This is a comment`
- [] `/_ This is a comment _\`
- [] `// This is a comment`

Q78. What is the correct syntax for replacing the string apple in the list with the string orange?

- [ ] orange = my_list[1]
- [x] my_list[1] = 'orange'
- [ ] my_list['orange'] = 1
- [ ] my_list[1] == orange

Q79. What will happen if you use a while loop and forget to include logic that eventually causes the while loop to stop?

- [ ] Nothing will happen; your computer knows when to stop running the code in the while loop.
- [ ] You will get a KeyError.
- [x] Your code will get stuck in an infinite loop.
- [ ] You will get a WhileLoopError.

Q80. Describe the functionality of a queue?

- [x] A queue adds items to either end and removes items from either end.
- [ ] A queue adds items to the top and removes items from the top.
- [ ] A queue adds items to the top, and removes items from anywhere in, a list.
- [ ] A queue adds items to the top and removes items from anywhere in the queue.

Q81. Which choice is the most syntactically correct example of the conditional branching?

- [x]

```
 num_people = 5

if num_people > 10:
    print("There is a lot of people in the pool.")
elif num_people > 4:
    print("There are some people in the pool.")
else:
    print("There is no one in the pool.")
```

- []

```
 num_people = 5

if num_people > 10:
    print("There is a lot of people in the pool.")
if num_people > 4:
    print("There are some people in the pool.")
else:
    print("There is no one in the pool.")
```

- []

```
num_people = 5

if num_people > 10;
    print("There is a lot of people in the pool.")
elif num_people > 4;
    print("There are some people in the pool.")
else;
    print("There is no one in the pool.")
```

- []

```
if num_people > 10;
    print("There is a lot of people in the pool.")
if num_people > 4;
    print("There are some people in the pool.")
else;
    print("There is no one in the pool.")
```

This question seems to be an updated version of Question 19.

Q82. How does `defaultdict` work?

- [] `defaultdict` will automatically create a dictionary for you that has keys which are the integers 0-10.
- [] `defaultdict` forces a dictionary to only accept keys that are of the types specified when you created the `defaultdict` (such as strings or integers).
- [x] If you try to read from a `defaultdict` with a nonexistent key, a new default key-value pair will be created for you instead of throwing a `KeyError`.
- [] `defaultdict` stores a copy of a dictionary in memory that you can default to if the original gets unintentionally modified.

Updated version of Question 14.

Q83. What is the correct syntax for adding a key called `variety` to the `fruit_info` dictionary that has a value of `Red Delicious`?

- [] `fruit_info['variety'] == 'Red Delicious'`
- [x] `fruit_info['variety'] = 'Red Delicious'`
- [] `red_delicious = fruit_info['variety']`
- [] `red_delicious == fruit_info['variety']`

Q84. When would you use a `while` loop?

- [] when you want to minimize the use of strings in your code
- [] when you want to run code in one file while code in another file is also running
- [x] when you want some code to continue running as long as some condition is true
- [] when you need to run two or more chunks of code at once within the same file

Q85. What is the correct syntax for defining an `__init__()` method that sets instance-specific attributes upon creation of a new class instance?

- []

```
def __init__(self, attr1, attr2):
    attr1 = attr1
    attr2 = attr2
```

- []

```
def __init__(attr1, attr2):
    attr1 = attr1
    attr2 = attr2
```

- [x]

```
def __init__(self, attr1, attr2):
    self.attr1 = attr1
    self.attr2 = attr2
```

- []

```
def __init__(attr1, attr2):
    self.attr1 = attr1
    self.attr2 = attr2
```

Explanation: When instantiating a new object from a given class, the `__init__()` method will take both `attr1` and `attr2`, and set its values to their corresponding object attribute, that's why the need of using `self.attr1 = attr1` instead of `attr1 = attr1`.