Data Science C - Data Science C - UT Invitational 2020 - Division C - 11-08-2020

This exam has two parts: 1) A set of coding challenges hosted on Google Colab. 2) A test on Scilympiad.

For the coding challenges, be sure to **read all instructions carefully** (failure to do so may result in losing points).

This trial event was originated by Dhruva Karkada in the 2019-2020 season and was held for the first time at the UT Austin Invitational. We're excited to be running it again!

1. (40.00 pts)
The Google Colab notebook containing the coding challenges can be found here: https://colab.research.google.com/drive/1iWSIJ4LJx4IyPNMNihc_ApdvcIbl--ef?usp=sharing (https://colab.research.google.com/drive/1iWSIJ4LJx4IyPNMNihc_ApdvcIbl--ef?usp=sharing)

This is the master copy. It will not let you save the file if you edit it. To complete the challenges, **click the "Copy to Drive" button in the upper left**. This will create a private copy of this file in your google drive, which you can edit and save. Make only one copy per team -- your partner should be able to edit the notebook simultaneously, as long as you both have access to the document.

To submit your notebook, make sure to enable link sharing. This is done by clicking the "Share" button in the upper right, clicking "Change to anyone with the link" at the bottom of the popup, and then clicking "Copy link." **Make sure that you share with the setting "Anyone on the internet with this link can EDIT." This is extremely important, as it allows us to check if you edited your code after the submission time. If you do not do share your Colab Notebook with this setting, your coding challenges will not be scored.**

Before your test ends on Scilympiad, you should submit your Google Colab link through this google form: FORM LINK (https://forms.gle/Nq7Qs9e71QEsVFzNA)

Expected Answer:

```python
from enum import Enum

class Role(Enum):
    TAMK = 0
    DAMAGE = 1
    SUPPORT = 2

class Hero:
    ult_charge = 0

    def __init__(self):
        pass

    def add_ult_charge(self, amount):
        self.ult_charge += amount;

    def print_info(self):
        print("I'm a hero in the game Overwatch!");


class Overwatch(Hero):
    def __init__(self):
        pass

    def print_info(self):
        print("I'm part of Overwatch");

class Talon(Hero):
    def __init__(self):
        pass

    def print_info(self):
        print("I'm part of Talon!");

class Human(Hero):
    def __init__(self):
        pass

    def print_info(self):
        print("I'm a human!");

class Tracer(Overwatch, Human):

    hp = 150
    role = Role.DAMAGE
    needed_charge = 25

    def __init__(self):
        pass

    def cast_ult(self, stick=False):
        if(self.ult_charge):
            if(stick):
                print("You need a time out!")
            else:
                print("Bombs away")
        else:
            print("My ultimate is charging")

class Moira(Talon, Human):
    hp = 200
    role = Role.SUPPORT
    needed_charge = 100

    def __init__(self):
        pass

    def cast_ult(self):
        if(self.ult_charge):
            print("Géill do mo thoil!")
        else:
            print("My ultimate is charging")


def main():
    t = Tracer();
    m = Moira();

if __name__ == "__main__":
    main()
```

Use the above code to answer the following 8 questions.

2. (1.00 pts) Which of the following lines exhibits variable instantiation?

A) 23
[✓ B) 9]
C) 40
D) 52

3. (1.00 pts) Which of the following lines exhibits multiple inheritance?

[✓ A) 42]
B) 35
C) 76
D) 1

4. (1.00 pts) Which of the following lines exhibits singular inheritance?

A) 25
B) 15
C) 60
[✓ D) 21]

5. (1.00 pts) Which of the following lines exhibits assignment of a default argument?

A) 48
[✓ B) 51]
C) 15
D) 5

6. (1.00 pts) Which of the following lines exhibits method overriding?

A) 77
B) 14
C) 11
[✓ D) 25]

7. (1.00 pts) Which of the following lines exhibits creation of an enum?

A) 80
B) 77
[✓ C) 3]
D) 1

8. (1.00 pts) Which of the following lines exhibits instantiation of an object?

[✓ A) 76]
B) 3
C) 1

D) 6

9. (1.00 pts) Suppose we created a Tracer object and called print info method. What would print out?

[✓ A) I'm part of Overwatch]
B) I'm a human!
C) I'm part of Talon!
D) My ultimate is charging

10. (1.00 pts) What's the main difference between the methods __str__ and __repr__?

A) __repr__ is the human-readable format, while __str__ is a more technical format
B) There is no difference between the two, __repr__ is a legacy feature from Python 2
C) __repr__ should only be called by the system, while __str__ can be called by users
[✓ D) __str__ returns a user friendly representation the object, while __repr__ generally returns the raw object format]

11. (1.00 pts) Suppose I had a Python Statement such as the following:

```
[x for x in range(0, 10) if x % 2 == 0]
```

This would be an example of what python language concept?

[✓ A) List comprehension]
B) List generation
C) List reduction
D) For loop transformation

12. (1.00 pts) The list in the previous question would contain what?

A) [0, 2, 4, 6]
[✓ B) [0, 2, 4, 6, 8]]
C) [1, 3, 5, 7]
D) [1, 3, 5, 7, 9]

13. (1.00 pts) What is the average time complexity of prepending to a python list?

A) O(1)
[✓ B) O(n)]
C) O(n^2)
D) O(log n)

14. (1.00 pts) What is the average time complexity of retrieving an element from a dictionary?

[✓ A) O(1)]
B) O(log n)
C) O(n)
D) O(n^2)

15. (1.00 pts) Suppose I had an ordered list of numbers of size *n* and I wanted to insert them into a Binary Search Tree? What would be the big-O of this operation?

A) O(log n)

B) O(n log n)
[✓ C) O(n)]
D) O(n^2)

16. (1.00 pts) Suppose I converted the Binary Search Tree in the previous problem into a Red-Black Tree. What would be big O of this operation?

A) O(1)
[✓ B) O(log n)]
C) O(n^2)
D) O(2^n)

17. (1.00 pts) Assume x is a list. What does the following line of python do?

```
y = x[:]
```

[✓ A) Creates a deep copy of x and stores it in y]
B) Creates a shallow copy of x and stores it in y
C) Nothing, this is an example of a python noop
D) None of the above

18. (1.00 pts) If I had a program written in C and a program written in Python, both of which did the exact same thing, which program would have better performance and why?

[✓ A) C, because it is a compiled language which runs much faster than Python]
B) Python, because it is an interpreted language and runs faster than C
C) They would have the exact same performance
D) C, because C is a low-level language

19. (1.00 pts)

```
l = []
t = ()
s = set()
f = frozenset()
d = {}
i = 1
```

Check all the statements where the assertion fails.

(Mark **ALL** correct answers)
[✓ A) assert l is []]
B) assert t is ()
[✓ C) assert s is set()]
D) assert f is frozenset()
[✓ E) assert d is {}]
F) assert i is 1

20. (1.00 pts) Given:

```
# numpy.shape(x) == (1, 10)
# numpy.shape(M) == (20, 10)
# k == 50
```

What is the output of the following? Enter your answer in the form "(a, b)", where a and b are integers. If there is an error, type "ERROR".

```
print(numpy.shape(k * numpy.matmul(M,x.T))
```

(20, 1)

21. (1.00 pts) Given:

```
# numpy.shape(x) == (10, 1)
```

What is the output of the following? Enter your answer in the form "(a, b)", where a and b are integers. If there is an error, type "ERROR".

```
print(numpy.shape(numpy.matmul(x,x.T))
```

(10, 10)

22. (1.00 pts)

```
l = []
t = ()
s = set()
f = frozenset()
d = {}
i = 1
```

```
s.add(t)
s.add(f)
s.add(d)
s.add(i)
```

Are the above statements legal?

True [✓ False]

23. (1.00 pts)

```
#Child is a subclass of Parent
def f (num) :
        if (num == 0)
                raise Parent(...)
        if (num == 1)
                raise Child(...)

def g (num) :
        try :
                a = f(num)
        except Parent as e :
                print(“hello”)
        except Child as e :
                print(“world”)
```

What is the result of calling g(1)?

[✓ A)]
```
hello
```

B)
```
world
```

C)
```
hello
world
```

D)
```
<error>
```

24. (1.00 pts) Consider the following generator function myzip().

myzip() takes in any iterable (list, set, etc.) and has the following behavior

```
Input:
a = [1,2,3]
b = [4,5,6]
c = [7,8,9]

Code:
x = myzip(a, b, c)
list(x)

output: [(1, 4, 7), (2, 5, 8), (3, 6, 9)]
```

Here is the implementation of myzip():

```
#assume iterables are all equal length
def myzip(*iterables):
    iterators = [MISSING for iterable in iterables]
    while iterators:
        result = []
        for it in iterators:
            temp = next(it)
            result.append(temp)
        yield tuple(result)
```

What goes in the place of MISSING?

iter(iterable)

25. (1.00 pts)

```
Input:
a = [1,2,3]
b = [4,5,6]
c = [7,8,9]

Code:
x = myzip(a, b, c)
list(x)

output: [(1, 4, 7), (2, 5, 8), (3, 6, 9)]
```

What is the additional output if we call list(x) again?

A) `[(1, 4, 7), (2, 5, 8), (3, 6, 9)]`

[✓ B)] `[]`

C) `<syntax error>`

D) `<runtime error>`

26. (1.00 pts) Consider f() as an implementation of a factorial calculation. For example 4! = 24 and f(4) = 24.

The details of reduce() are described below:

```
def reduce(function, iterable, initializer=None):
    it = iter(iterable)
    if initializer is None:
        value = next(it)
    else:
        value = initializer
    for element in it:
        value = function(value, element)
    return value
```

```
def f(x):
    return reduce(lambda a,b : a*b, range(1, x + 1), 0)
```

There appears to be a bug in f(). f(4) returns ____ and to fix the bug, we must change the last argument in the function call to ____ or just remove it.

0

1

27. (1.00 pts)
In Python, you can pass functions as parameters and define functions inside functions. These nested functions cannot be accessed from the outside, but they allow us to build something called a decorator, which takes a function and modifies its behavior.

```
def mydecorator(f):
    def g():
        print("The result is:")
        f()
    return g

def myfunction():
    print("One")


myfunction = mydecorator(myfunction)
```

What is the result of a call to myfunction()?

[✓ A)]
```
The result is:
One
```

B)
```
One
```

C)
```
<syntax error>
```

D)
```
<runtime error>
```

28. (1.00 pts)

```
def mydecorator(f):
    def g():
        print("The result is:")
        f()
    return g
```

Let's redefine myfunction() to take in a parameter.

```
def myfunction(num):
    print(num * num)


myfunction = mydecorator(myfunction)
```

Look carefully at mydecorator() and g() again. What is the result of a call to myfunction(3)?

A)
```
The result is:
9
```

B)
```
9
```

[✓ C)]
```
<syntax error>
```

D)
```
<runtime error>
```

29. (1.00 pts)

```
class Foo():
    i = 1
    def bar(self):
        print (Foo.i)
        print (self.i)


f = Foo() # Create an instance of Foo
```

What is the output of f.bar()?

[✓ A)]
```
1
1
```

B)
```
1
2
```

C)
```
2
1
```

D)
```
2
2
```

E)
```
<error>
```

30. (1.00 pts)

```
class Foo():
    i = 1
    def bar(self):
        print (Foo.i)
        print (self.i)


f = Foo() # Create an instance of Foo
```

```
Foo.i = 2
```

What is the output of f.bar() after we execute the above statement?

- ○ A)
```
1
1
```
- ○ B)
```
1
2
```
- ○ C)
```
2
1
```
- [✓ D)]
```
2
2
```
- ○ E) `<error>`

31. (1.00 pts)

```
class Foo():
    i = 1
    def bar(self):
        print (Foo.i)
        print (self.i)


f = Foo() # Create an instance of Foo
```

```
Foo.i = 2
f.i = 1
```

What is the output of f.bar() after we execute the above statements?

- ○ A)
```
1
1
```
- ○ B)
```
1
2
```
- [✓ C)]
```
2
1
```
- ○ D)
```
2
2
```
- ○ E) `<error>`

32. (1.50 pts) Let A, and B be mutually exclusive events. What is P(A ∩ B)?

- ○ A) P(B)
- ○ B) P(A)P(B)
- ○ C) P(A)
- [✓ D) 0]

33. (1.50 pts) Let A, and B be independent events. What is P(A ∩ B)?

[✓ A) P(A)P(B)]
B) P(A)
C) P(B)
D) 0

34. (1.50 pts) A binomial distribution can be expressed as which of the following?

A) The Product of Identical Poisson Distributions
B) The Sum of Identical Hypergeometric Distributions
[✓ C) The Sum of Identical Poisson Distributions]
D) The Product of Identical Normal Distributions

35. (1.50 pts) Suppose the random variable X models a fair 6-sided dice roll. Determine its expected value E[X].

A) 3
[✓ B) 3.5]
C) 4
D) None of the above

36. (1.50 pts)
Suppose we knew the probability that a device was defective given the manufacturer. What could we use to find the probability that a certain manufacturer made the device, given that we have the probability that a device is defective?

A) Central Limit Theorem
B) Chebyshev inequality
C) Jensen inequality
[✓ D) Bayes' Theorem]

37. (1.50 pts) Suppose the random variable X modes a fair six-sided dice roll. Determine its variance, var(X).

A) 1.57
[✓ B) 2.92]
C) 3
D) 3.5

38. (1.50 pts)
Farmer John wants to make sure that his cows are eating an adequate amount of food. Suppose the random variable F represents the amount of food a cow on Farmer John's farm gets. F is a Gaussian random variable with mean 10.5 and variance 5.3. Suppose Daisy, a cow on Farmer John's farm, eats 12.2 kg of food. Determine Daisy's percentile, rounded to the nearest whole percentage. You can look up a z-score table for this problem.

[✓ A) 77th percentile]
B) 60th percentile
C) 5th percentile
D) 95th percentile

39. (1.50 pts) Consider a smooth probability distribution function P(x), defined for all real numbers x. The probability that x=3 is

[✓ A) 0]
B) P(3)

C) P(1/3)

D) The area under the curve between x=0 and x=3

40. (1.50 pts) Which of the following is NOT TRUE about the mean and sample mean?

A) For Gaussian distributions, the sample mean is an unbiased estimator of the mean.

B) Given a distribution, the mean is a deterministic value whereas the sample mean is a random variable.

C) The variance of the sample mean depends on the number of samples.

[✓ D)] The mean is equal to the sample mean for Poisson distributions.

41. (1.50 pts)
A statistician tests a random population sample and finds that 3% of them test COVID-positive. So she estimates the true COVID infection rate to be 3%. After doing more analysis, she finds a 95% confidence interval of [0%, 13%]. Which of the following is a valid interpretation of this statistical measure?

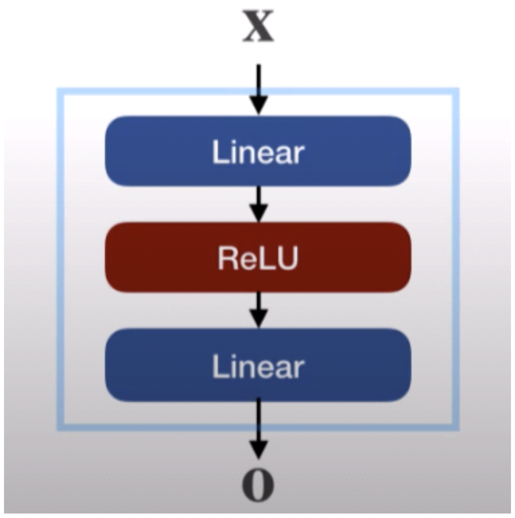
A) 95% of people will test COVID-negative

B) Between 0% and 13% of people have an 5% uncertainty of having COVID

[✓ C)] 95% of sample populations will have an infection rate between 0% and 13%

D) The true infection rate is 95% likely to be between 3% and 16%

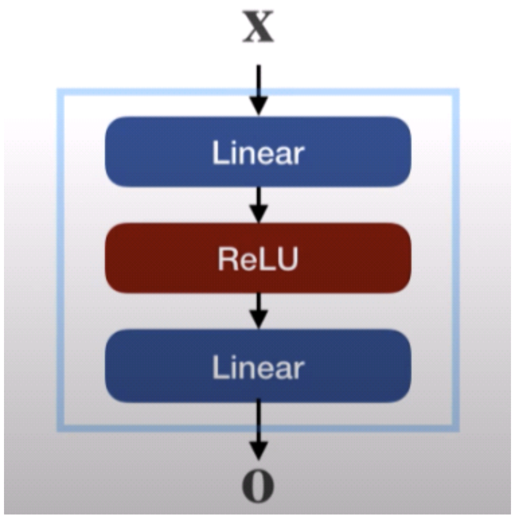
42. (4.00 pts)
The largest unit of computation that's structure is the same throughout neural network architectures is called a layer. Examples of layers include linear layers (fully connected layer, convolutional layer), activation functions (ReLU, Softmax, TanH, Leaky ReLU), pooling layers, loss functions, output transformations, etc.



What are linear layers? What type of functions can they represent? What do the innards of a linear layer look like given the input vector x? Why don't machine learning engineers put a linear layer directly after another linear layer?

Expected Answer: Linear layers are the fundamental building block of neural networks. Linear layers can only represent linear functions (or linearly separated functions). x*W+b, where W(eights) is a matrix and b(iases), is a vector. Putting a linear layer directly after another linear layer with no activation layer in between is just another linear layer.

43. (4.00 pts)
The largest unit of computation that's structure is the same throughout neural network architectures is called a layer. Examples of layers include linear layers (fully connected layer, convolutional layer), activation functions (ReLU, Softmax, TanH, Leaky ReLU), pooling layers, loss functions, output transformations, etc.



What are activation functions and why do we use them? Which of the following binary functions requires an activation function for a neural net to "learn" it: AND, OR, XOR, NOT, NAND, NOR?

Expected Answer: Activation functions are needed for neural networks to be able to learn non-linear functions. The XOR binary function is non-linear and thus would require an activation function.

44. (3.00 pts) When is it a good idea to use a clustering algorithm (i.e. k-means, DBSCAN, etc) vs a deep learning model?

Expected Answer: Clustering is useful for extracting structure from unlabeled data (unsupervised) Deep learning works best with labelled data (supervised)

45. (4.00 pts)
Principal component analysis is a linear transformation that decomposes a high-dimensional vector space into the directions that best explain the variance in a data set. Explain how this relates to dimensionality reduction (one of the most common applications of PCA).

Expected Answer: The high-variance directions (the first few principal components) tend to account for most of the structure in the data. So we can often transform the dataset using PCA and then project the data onto the hyperplane containing only the first few principal components. This effectively reduces the dimension of the dataset without losing too much structure.

Thank you for competing in Data Science! Good luck with your other events!