Greek Mythology

Consider the following code for extracting the two names of a Mythological Figure:

```
def mythlogy_intro(name, realm='earth'):
    _____(a)_____
    _____(b)_____
    intro_print(name, greek_name, roman_name, realm)
def intro_print(name, get_greek_name_func, get_roman_name_func,
realm='earth'):
    print("Introducing the Great", name)
    print(" Greek name is", get_greek_name_func(name))
    print(" Roman name is", get_roman_name_func(name))
    print(" Rules over", realm)

mythlogy_intro('Athena Minerva', 'wisdom and war')
mythlogy_intro('Poseidon Neptune', 'sea and waters')
```

Output from the above code is:

```
Introducing the Great Athena Minerva
 Greek name is Athena
 Roman name is Minerva
 Rules over wisdom and war
Introducing the Great Poseidon Neptune
 Greek name is Poseidon
 Roman name is Neptune
 Rules over sea and waters
```

1.What lines of code could go in blank **(a)** ? — **choose all options that could work**

A.
```
def greek_name(name):
    return name.split(' ')[0]
```

B.
```
def greek_name(name):
    return name.split()[1]
```

C.
```
greek_name = lambda full_name : full_name.split()[0]
```

D.
```
greek_name = lambda full_name : full_name.split()[1]
```

E.
```
greek_name = lambda full_name : full_name.split(" ")[1]
```

2.What lines of code could go in blank **(b)**? — **choose all options that could work**

A.
```
def roman_name(name):
    return name.split(' ')[1]
```

B.
```
def roman_name(name):
```

```
        return name.split()[1]
```

C.
```
def roman_name(name):
    return name.split()[0]
```

D.
```
roman_name = lambda full_name : full_name.split()[0]
```

E.
```
roman_name = lambda full_name : full_name.split()[1]
```

Period of a Pendulum

The time for a pendulum to complete a cycle is T= 2π(√(I / mgL))

I being the inertia of the center of mass

m being the mass

L being the distance between the center of mass and the pivot g being the acceleration of gravity

The Python `math` library has both the variable `pi` and the function `sqrt` which will be needed. Recall the Python built-in function `pow` that raises the first parameter to the second parameter. Consider the code:

```
import math

EARTH_GRAVITY = 9.81
MOON_GRAVITY = 1.62

def period(inertia, mass, length, gravity=_____(a)):
  pi = _____(b)
  def division(x, y):
    return ____(c)
  return 2 * pi * math.sqrt(division(inertia,(___(d))

print(period(2*(3**2), 2, 3))
```

3.What line of code could go in blank **(a)** assuming we want the Earth's gravity to be the default?

A. `EARTH_GRAVITY`

B. `MOON_GRAVITY`

A. `math.pi`

C. `math.sqrt`

D. `x / y`

E. `mass * length * gravity`
F. `inertia / (mass * length * gravity)`

4.What line of code could go in blank **(b)** ?

A. `EARTH_GRAVITY`
B. `MOON_GRAVITY`
C. `math.pi`
D. `math.sqrt`
E. `x / y`
F. `mass * length * gravity`
G. `inertia / (mass * length * gravity)`

5.What line of code could go in blank **(c)** ?

A. `EARTH_GRAVITY`
B. `MOON_GRAVITY`
C. `math.pi`
D. `math.sqrt`
E. `x / y`
F. `mass * length * gravity`
G. `inertia / (mass * length * gravity)`

6.What line of code could go in blank **(d)** ?

A. `EARTH_GRAVITY`
B. `MOON_GRAVITY`
C. `math.pi`
D. `math.sqrt`
E. `x / y`
F. `mass * length * gravity`
G. `inertia / (mass * length * gravity)`

Let's Draw

Consider the `Pencil` class and fill in the missing code:

```
class Pencil:
  def __init__(self, color, weight):
    self.color = color
    self.weight = weight

  def __str__(self):
    return f'{___(a)} Pencil with the color {___(b)}'
```

7.What line of code could go in blank **(a)**?

A. `Pencil`
B. `self.weight`
C. `self.color`
D. `nextPencil = nextPencil.next`
E. `n += 1`
F. `currentPencil = currentPencil.next`

8.What line of code could go in blank **(b)**?

A. `Pencil`
B. `self.weight`
C. `self.color`
D. `nextPencil = nextPencil.next`
E. `n += 1`
F. `currentPencil = currentPencil.next`

Consider the following linked-list like class `PencilBox` that stores a linked list of `Pencils`:

```
class PencilBox:
   def __init__(self, currPencil, nextPencil):
    assert issubclass(type(currPencil), ___(c)) or type(currPencil) ==
___(c)
    assert issubclass(type(nextPencil), ___(c)) or type(nextPencil) ==
___(c)
    self.current = currPencil
    self.next = nextPencil
```

9.What line of code could go in both blanks **(c)** ?

A. `Pencil`
B. `self.weight`
C. `self.color`
D. `nextPencil = nextPencil.next`
E. `n += 1`
F. `currentPencil = currentPencil.next`

The following function is for the `PencilBox` class to compute the length of the list.

```
  def __len__(self):
    n = 1
    nextPencil = self.nextPencil
    while isinstance(nextPencil, Pencil):
        ___(d1)
        ___(d2)
    if nextPencil is not None:
        raise TypeError('Length attempted on mixed pencil box')
    return n
```

10.What two lines of code could go in blanks **(d1) and (d2)** ? **— choose two that work together**

A. `Pencil`
B. `self.weight`
C. `self.color`
D. `nextPencil = nextPencil.next`
E. `n += 1`
F. `currentPencil = currentPencil.next`

11.Suppose you want to use inheritance to create classes of specific types of pencil. Would the following be a valid Python class that inherits from the Pencil class? (True/False)

```
class Crayola(Pencil):
```

```
name = 'red'
weight = 5
def __init__(self):
     super().__init__(Crayola.name, Crayola.weight)
```

Decode the Message

```
import re

""" The list is each letter attached to the corresponding index, a = 0, z = 25 """

super_secret_list = [a, b, ... , z]
def decode_letter(n):
      return super_secret_list[n]
def decode_number(n):
      return (n + 3)
def decoded_message(message):
    decoded_message = []
    for letter in ____(a)____:
        if ____(b)____:
            decoded_message += decode_letter(letter)
        elif ____(c)____:
            decoded_message += decode_number(letter)
        else:
            print("Message could not be decoded")
            return
    return ____(d)____
```

12. Which of the following would go in blank **(a)** above?
    A. message
    B. decoded_message
    C. decode_letter(message)
    D. decode_message(my_str)
    E. super_secret_list
13. Which of the following would go in blank **(b)** above?
    A. re.match(r"[A-Za-z]", letter)
    B. re.match(r"\d", letter)
    C. True
    D. letter is type(str)
    E. is_prime
14. Which of the following would go in blank **(c)** above?
    A. re.match(r"[A-Za-z]", letter)
    B. re.match(r"\d", letter)
    C. True
    D. letter is type(str)
    E. is_prime
15. Which of the following would go in blank **(d)** above?
    A. message
    B. decode_letter(message)

C. ndigits
D. decoded_message
E. super_secret_list

Trees

```
class FamilyTree:
    def __init__(____(a)____):
        self.name  = name
        self.mother  = None
        self.father = None

    def set_mother(self, branch):
        self.mother = branch

    def set_father(self, branch):
        self.father = branch

def find_names( tree ):
     """List the names of all the tree family members"""
    if ____(b)____:
        return []
    else:
        names = []
        names += ____(c)____
        names += ____(d)____
    return ____(e)____  + names
```

16. What expression should go in blank **(a)** above?
    A. `self, name=None`
    B. `self, name, mother=None, father=None`
    C. `self`

17. What expression should go in blank **(b)** above? — **choose all options that could work**
    A. `not tree`
    B. `tree == None`
    C. `isinstance(tree, Tree)`

18. What expression should go in blank **(c)** and **(d)** above?
    A. `tree.name`
    B. `find_names(tree.father)`
    C. `find_names(tree.mother)`
    D. `tree.father`

E. `tree.mother`

19. What expression should go in blank **(e)** above?

A. `tree.mother`
B. `tree.father`
C. `tree.name`
D. `self.name`

HTML

Consider the following HTML file

```
<html>
  <head>
    <title>A CS 111 Practice Final Doc</title>
  </head>
  <body>
  <h1 id="intro">Introduction</h1>
  <p id="intro">This page is from your <a href="/staff/#tas">TA's</a>
on the final exam!</p>
  <img src="/assets/images/staff/david_bauch.jpeg" alt="The best TA">
  <!-- Notice the '/' in src -->
  <p class="list-start">Let's remember some things:</p>
  <ul>
    <li class="odd">How to request a web page</li>
    <li class="even">How to find a tag</li>
    <li class="odd">How to construct URLs</li>
    <li class="even">How to print content on a page</li>
    <li class="odd">How to use regular expressions in <a

href="https://www.crummy.com/software/BeautifulSoup/bs4/doc/">
        BeautifulSoup</a></li>
  </ul>
  <h1 id="farewell">Goodbye now</h1>
  <p id="farewell">Hopefully this preps you for the final.</p>
  <img src="assets/images/staff/connor_nesbit.jpeg" alt="The 2nd best
TA">
  <!-- Notice the NO '/' in src -->
  </body>
</html>
```

Assume that this page is located at http://cs111.byu.edu/practice/final.html, and that you have created a BeautifulSoup object called **`soup`** from this HTML file.

20. Which of the following would return the paragraph tag with the 'id' attribute "farewell"?

A. `soup.find_all("p")`
B. `soup.find_all("p", id=True)`

C. `soup.find_all(True, "farewell")`
D. `soup.find_all(id="farewell")`
E. `soup.find_all("p", {"id": "farewell"})`
F. `soup.find_all(True, {"id": "farewell"})`

21. What is the full URL to the image after the first paragraph and the img after p#farewell?
A. `http://cs111.byu.edu/assets/images/staff/david_bauch.jpeg`
B. `http://byu.edu/assets/images/staff/david_bauch.jpeg`
C. `http://cs111.byu.edu/practice/david_bauch.jpeg`
D. `http://cs111.byu.edu/practice/assets/images/staff/connor_nesbit.jpeg`
E. `http://byu.edu/practice/assets/images/staff/david_bauch.jpeg`
F. `http://byu.edu/practice/assets/images/staff/david_bauch.jpeg`

22. What is the full URL to the staff page linked in the first paragraph?
A. `http://cs111.byu.edu/final/staff`
B. `http://byu.edu/staff`
C. `http://cs.byu.edu/staff/`
D. `http://cs111.byu.edu/staff/`

23. What code would you write to get the text of the link to staff page (not the link, but the text that is being linked)
A. `soup.find_all('a')`
B. `soup.find_all('href')`
C. `soup.find_all('href')[0].string`
D. `soup.find_all('a')[0]`
E. `soup.find_all('a')[0].string`
F. `soup.find_all('a')[0]['href']`

24. What does the following code print?

```
tags = soup.find_all('li')
for tag in tags:
    print(tag.string)
```

A.
```
How to request a web page
How to find a tag
How to construct URLs
How to print content on a page
How to use regular expressions in
```

B.
```
How to request a web page
```

How to construct URLs
How to use regular expressions in

C. How to find a tag
How to print content on a page

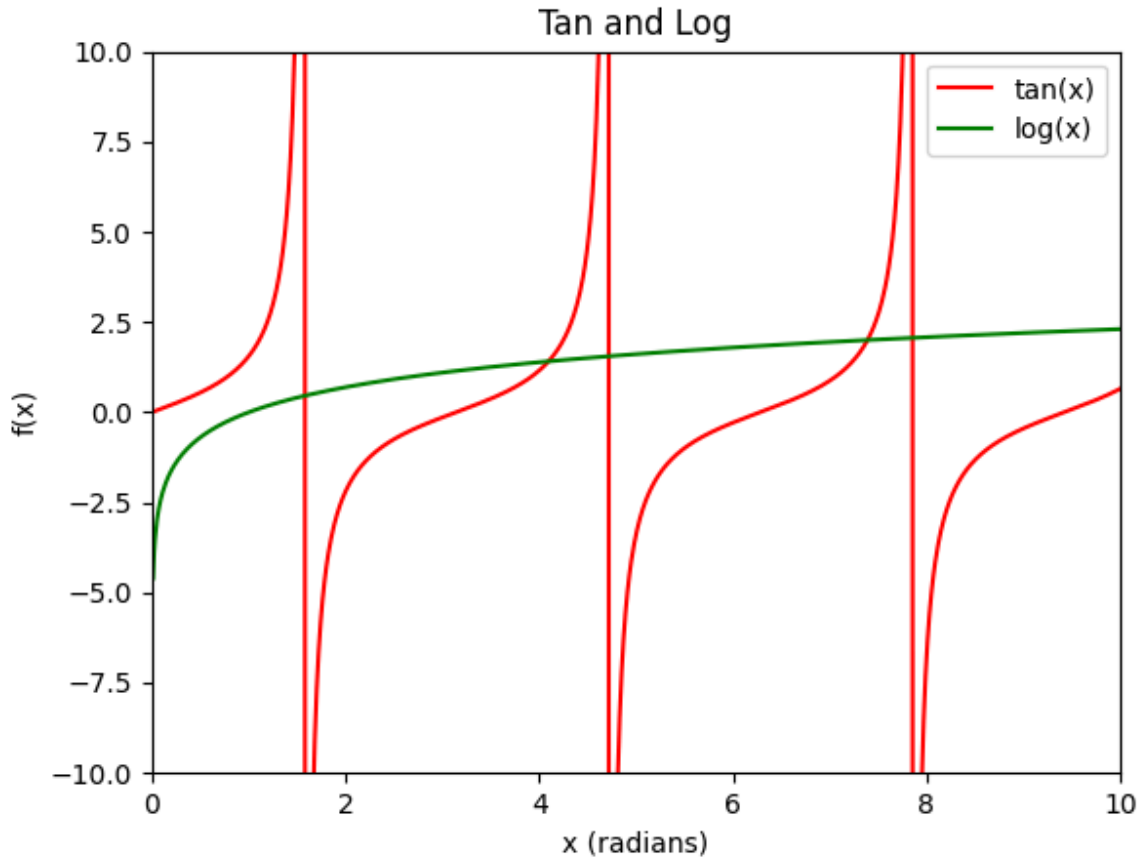
D. Hopefully you remember the following:

E. It prints nothing

(6 points) **Plotting**

Consider the following code that constructs the plot below it

```
1   import matplotlib.pyplot as plt
2   from math import tan, log
3
4   x = [i/100 for i in range(1, 1300)]
5   y1 = [tan(val) for val in x]
6   y2 = [log(val) for val in x]
7
8   plt.plot(x, y1, _____(a)_____, _____(c)_____="tan(x)")
9   plt.plot(x, y2, _____(b)_____, _____(c)_____="log(x)")
10  plt.xlim(0, 10)
11  plt.ylim(-10, 10)
12  plt.title("Tan and Natural Log")
13  plt._____(d)_____("x (radians)")
14  plt.ylabel("f(x)")
15  plt.legend()
16
17  plt.show()
```



25. Which of the following can go in positions (a) & (b) on lines 8 & 9 to produce the red and green lines?

A. ‘red’, ‘green’
B. ‘r’, ‘g’
C. ‘green’, ‘red’
D. ‘g’, ‘r’

26. What keyword goes in position (c) on lines 8 & 9 to give a title to the sets of points?

A. label
B. title
C. name
D. id
E. text

27. (2 points) What keyword goes in position (d) on line 11 to put the text at the bottom of the plot?

A. xlabel
B. ylabel
C. title
D. name
E. id