1. What is the type of btz after the following executes?
id = 1
btz = [{“name” : “Coleen”},{“id”:id}]
btz = btz{id}
YOU CANNOT INDEX A DICTIONARY
2. write a for loop to display all the elements of a tuple
friends = {‘Joseph’, ‘Sarah’, ‘Alice’]

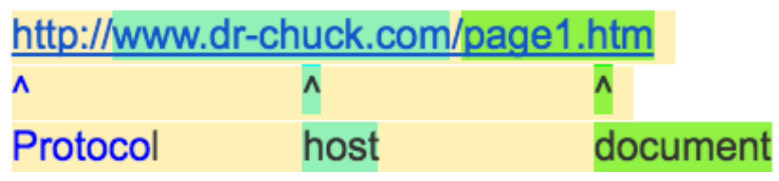
```
    for friend in friends:
    print (‘Happy New Year’:, friend)
```

3. Define and initialize an example list, dictionary, n tuple
lst = [] dict = { } tuple = (A, B, C)
4. Write a regular expression that describes a string with no digits in it that is atleast four characters long
re.findall([0^-9]\s\s\s\s+)
5. What happens if you forget the -m in a git commit
a new text editor window opens in terminal
6. what is the git command to see your activity /git status
7. what does it mean to stage a file
git add . Name the data structure that have a method called .items( ) — dictionaries
8. do you use json.loads( ) or json.dumps( ) to create a str
dumps creates a string json.loads( ) loads the string
9. whats the difference between urllib.urlopen(url) and urllib.urlopen(url).read( ) // open and read as a string
10. what is the output?
var = “I love SI206”
print (var[2:4])
print (var[-1])

Name 4 common functions used on lists: s = s.strip( ) s.upper( ) s.replace( ) s.lower ( )

Name 4 common list methods
list.append( ) list. remove( ) list.sort( ) list.reverse( )
Give an example of when it would be best to utilize a regular expression, Beautiful Soup, and API
Beautiful soup = HTML API = SQL
Regex = string inputs and emails, small things
Fill in the code for function test1 to check if two variables (of your choice) are of the same type:

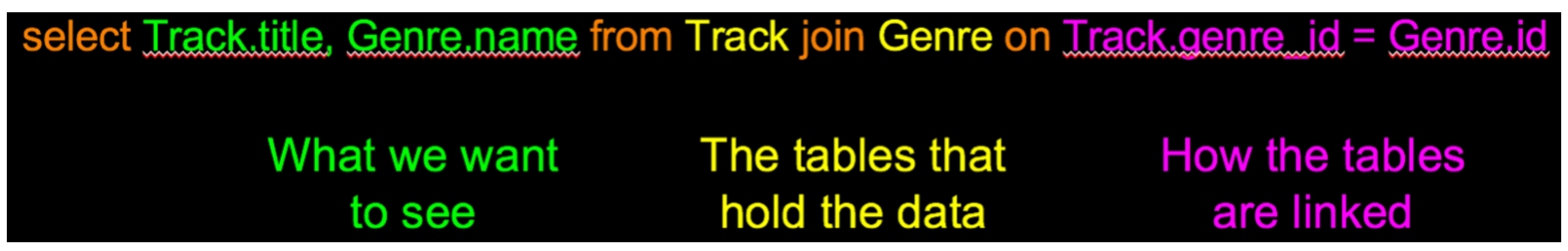
```
class Problem1(unittest.TestCase)
    def test1(self):
        i = 1
        j = 1
        self.assertEqual(i,j)
```


http://www.dr-chuck.com/page1.htm
^ ^ ^
Protocol host document

Given the following code, you want to sort lst1 on the third element of each list. Write the code for three different options for replacing ___???__:
    using lambda, key = lambda = i[2]
    using itemgetter( ) and key = operator.itemgetter(2)
    using a seperate function three different
lst1 = [[1,2,3,6], [4,5,1,6], [1,1,1,]
sorted_lst1 = sorted(lst1, key =__???__)
write a valid SQL statement to insert a new tuple with the name Dorian and age 17. INSERT INTO Users (name, age) VALUES (Dorian, 17)
write a valid SQL statement to return just the names of everyone with an age greater than 25. SELECT name FROM Users WHERE age > 25

Write a valid Python statement to insert a new tuple with the name Dorian and age 17 - query = “INSERT INTO Users (name.age) VALUES (‘Dorian’, ‘17’

JSON - Javascript Object Notation
- No tags, We construct our JSON by nesting dictionaries (objects) and lists as needed.
- Can be used to retrieve data from web server like XML, Key value pairs
SQL - Structured Query Language
- CRUD: **C**reate (insert) - inserts two into a table, **R**etrieve, **U**pdate - Allows the updating of a field with a where clause ex. UPDATE Users SET name =‘Charles’ WHERE email = ‘csev@umich.edu’, **D**elete - Deletes a row in a table based on a selection criteria ex. DELETE FROM Users WHERE email = ‘ted@umich.edu’
- Keys
  - Primary key (Orange) - Genreally an integer quto-increment field
  - Logical key (Green) - What the outside world uses for lookup
  - Foreign key (Purple) - Generally an integer key pointing to a row in another table


select Track.title, Genre.name from Track join Genre on Track.genre_id = Genre.id
What we want to see
The tables that hold the data
How the tables are linked

- Rules
- Never use your logical key as the primary key
- logical keys can and do change, albeit slowly
- relationships that based on matching string fields, less effici than integers

API - Application Programming Interface - Use XML or JSON for APIs - When an application makes a set of services in its API Available over the web its a web service

INSERT INTO Users VALUES
DELETE FROM Users WHERE
UPDATE Users SET name = “Charles” WHERE
SELECT * FROM Users
SELECT * FROM Users WHERE
SELECT * FROM Users ORDER BY email

- HTML was designed to display data and to focus on how data looks. You must use *predefined* tags. HTML is case insensitive. When HTML is usd to display data, the data is stored inside your HTML. Is content unaware. Does not check syntax

- XML was designed to describe data and to focus on what the data is. You have to define your own tags in XML. It is case sensitive. Can separate data from HTML. with you data is stored outside your HTML. Content aware. Does a syntax check

```
def get_tweet(search_term):
  if search_term in Cache_diction: #looks for term in cached file
    print("using cache") #If term has been cached, the data will return that
    return Cache_diction[search_term]
  else:
    print("fetching") #If not, it pulls from Twitter
    results = api.search(search_term)
    try:
      Cache_diction[search_term] = json.dumps(results)
      dumped_json_cache = json.dumps(Cache_diction)
      fw = open(cache_fname, 'w')
      fw.write(dumped_json_cache)
      fw.close()
      return Cache_diction[search_term]
    except:
      print("Wasn't in cache and wasn't in search either") #If there was
                                                 nothing in cached or twitter
      return None
```