Foundation Assessment 1

Robyn Seymour-Jones

1. Python theory questions

1. Python is a popular programming language. These are languages that humans can read and write and that can be translated into a language that computers can understand. Python uses an interpreter to do this, which means it doesn't require a compiling stage. It is object-orientated. It is dynamically typed, which means that you don't have to explicitly declare the type of an object. It is strongly typed, which means that it does little implicit type conversion. It is open source so it is freely available for anyone to download and use and it also means that anyone can see the code base.

2. It counts how many occurrences there are of a specified value with your string.
   mystring = 'this is my string'
   letter_h_count = mystring.count('h')
   print(letter_h_count)
   1

3. ord() - returns a string representing a character from ASCII
   ord('r')
   114

   chr() - returns an ASCII character from a number
   chr(114)
   'r'

4. mystring = ' 2020-11-10_sales.csv'
   sliced_string = mystring[0:-4]

5. A tuple is immutable, which means the values in it can't be changed. You can modify the values in a list as it is mutable. A tuple is denoted by round brackets—(item1, item2). A list is denoted by square brackets—[item1, item2].

6. append() adds a single item to the end of a list
   extend() adds multiple items to the end of a list

7. An infinite loop occurs when you use a while loop and the code keeps on running. You can make sure you avoid it by making sure the condition is correct.

8. Arguments are values that you pass to a function.
9. FINISH IF TIME
10. Bool - immutable
    Int - immutable
    List - mutable
    Tuple - immutable
    Str - immutable
    Set - mutable
    Dict - mutable

Python coding questions

2.

```
with open ('my_file.txt') as fh:
text = fh.read()

count = 0

for character in text:
        if character.isupper()
        count = count + 1

print(count)
```

3.

```
my_string = 'apple'
my_reversed_string = my_string[::-1]
print(my_reversed_string)
```

4.

```
my_list = ['cat', 'horse', 'elephant', 'dog']

longest_word = 0

for word in my_list:
        if len(word)
```

FINISH IF TIME

5.

```
print ( "Please select operation - \n "
"1. Add \n "
"2. Subtract \n "
"3. Multiply \n "
"4. Divide \n " )

# Take input from the user
select = int ( input ( "Select operations form 1, 2, 3, 4 :" ))
number_1 = int ( input ( "Enter first number: " ))
number_2 = int ( input ( "Enter second number: " ))
```

```
### YOUR CODE GOES HERE ###

# Function to add two numbers
def sum(n1, n2):
        return n1 + n2

# Function to subtract two numbers
def subtract(n1, n2):
        return n1 – n2

# Function to multiply two numbers
def multiply(n1, n2):
        return n1 * n2

# Function to divide two numbers
def divide(n1, n2):
        return n1 / n2

# Calculator logic
if select == 1:
        result = sum(number_1, number_2)
        print(f'{number_1} + {number_2} = {result}')
elif select == 2:
        result = subtract(number_1, number_2)
        print(f'{number_1} – {number_2} = {result}')
elif select == 3:
        result = multiply(number_1, number_2)
        print(f'{number_1} * {number_2} = {result}')
elif select == 4:
        result = divide(number_1, number_2)
        print(f'{number_1} / {number_2} = {result}')
else:
        print('Sorry, I don’t understand.')
```

6. SQL theory questions

1. MySQL is a relational database management system that uses the SQL languages to manage and retrieve data from a relational database.

2. CHAR is used for a fixed number of characters. White trailing space is added if the string is not as long as the specified length or it is truncated if the string you are trying to insert is too long. VARCHAR is for variable character strings so each entry in a VARCHAR column will have different numbers of characters in it.

3. % is a wildcard used to indicate 0, 1 or any number of characters.
   _ is a wildcard used to indicate precisely 1 character.
   They can be used to return records where a string contains a particular pattern of characters. e.g. LIKE 'J%' would return records where the string starts with J and has any number of other characters after it.

4. WHERE is used to add a condition to a SELECT statement based on column values. HAVING is used to add a condition to a SELECT statement based on a values returned by aggregate functions (e.g. COUNT()) for a group of rows.

5. The CHECK constraint makes the server check that the values being entered into a column conform to the condition provided.
   E.g. The following would make sure someone was born before 1990.
   ```
   CREATE TABLE customers(
           cust_id INT PRIMARY KEY,
           fname VARCHAR(20),
           lname (VARCHAR(20),
           dob DATE
                   CHECK dob < 1990-01-01
           )
   ;
   ```

6. Joins merge two or more tables together horizontally, i.e. columns are added on.
   INNER JOIN - keeps only records present in both tables
   LEFT JOIN - keeps all records in the first table listed
   RIGHT JOIN - keeps all records in the last table listed

7. COMMIT and ROLLBACK are used in transactions. COMMIT saves the changes to the db, ROLLBACK undos the changes.

8. COUNT() returns the number of records (rows).
   SUM() adds the values in the records for that column.

9. A stored procedure is a block of code you write and can call on later and reuse. It differs from a function as it can return more than one value.

10. A clustered index sorts and stores the full records in the actual table based on their key values (the indexed column). There is only one clustered index per table. It is usually the primary key. In MySQL, the server automatically creates a clustered index from the primary key. An unclustered or secondary index is when a column or columns are stored in an order in a special table alongside values pointing like a map to the location of that record in the original table.

SQL coding questions

7.

a)

```
SELECT
        E.Candidate,
        SUM(E.Votes) AS Total_Votes
FROM
        ELECTIONS AS E
GROUP BY
        E.Candidate;
```

b)

```
SELECT
        E.Candidate,
        SUM(E.Votes) AS Total_Votes
FROM
        ELECTIONS AS E
GROUP BY
        E.Candidate
HAVING
        SUM(E.Votes) > 50;
```

c)

```
DELETE FROM ELECTIONS AS E
WHERE E.RecordID = 13;
```

d)

We can add a new index. It is currently in primary key order so this is probably the clustered index and is useful to keep this. We could add a secondary (non-clustered) index. This could be based on the Votes column as we seem to want to query this column a fair bit (although this would be a problem if the number of votes was changing as the index would need to be updated every time the number of votes is updated in the original table.

E.g.

```
CREATE INDEX vote_idx
ON ELECTIONS(Votes);
```

8.
```
SELECT
	E.Department_Name,
	AVG(S.Salary) AS Average_Departmental_Salary
FROM
	EMPLOYEES AS E
JOIN
	SALARIES AS S
	ON E.Employee_id = S.Employee_id
GROUP BY
	E.Department_Name
HAVING
	AVG(S.Salary) < 500;
```

9.
```
CREATE TABLE shopping(
	Number INT AUTO_INCREMENT PRIMARY KEY,
	Item VARCHAR(30),
	Price DECIMAL(5, 2)
	)
;
```

10.
```
INSERT INTO shopping
(Item, Price)
VALUES
('bread', 1.50),
('milk', 0.80),
('chocolate', 0.59),
('wine', 7.99);
```

Note - didn’t need to include Number in list of columns as it auto increments.