**1. Python theory questions**

1. What is the program?

A program is a specific set of ordered operations for a computer to perform. And the program is put into a storage area accessible to the computer.

1. What is the process?

A process is an instance of a program running in computer. It is close in meaning to task, a term used in some operating systems.

1. What is Cache?

Caches are used to store temporary files, using hardware and software components.

1. What is Thread and Multithreading?

A thread is a small set of instructions designed to be scheduled and executed by the CPU independently of the parent process. Multithreading is the ability of a program or an operating system to enable more than one user at a time without requiring multiple copies of the program running on the computer.

1. What is GIL in Python and how does it work?

GIL is the Python Global Interpreter Lock. It is a mutex (or a lock) that allows only one thread to hold the control of the Python interpreter. This means that only one thread can be in a state of execution at any point in time.

1. What is Concurrency and Parallelism and what are the differences?

Concurrency is an application that is processing more than one task at the same time. It is used for decreasing the response time of the system by using single processing units.

Parallelism is related to an application where the tasks are divided into smaller sub-tasks that are processed seemingly simultaneously or in parallel. It is used for increasing the throughput and computational speed of the system by using multiple processors.

Differences:

1. Concurrency: the task of running and managing multiple computations at the same time.

Parallelism: the task of running multiple computations simultaneously.

2. Concurrency: is achieved through the interleaving operation of processes on the central processing unit (CPU) or in other words by the context switching.

Parallelism: is achieved by through multiple central processing units(CPUs).

3. Concurrency can be done by using a single processing unit.

Parallelism can’t be done by using a single processing unit. it needs multiple processing units.

4. Concurrency increases the amount of work finished at a time.

Parallelism improves the throughput and computational speed of the system.

5. Concurrency deals a lot of things simultaneously.

Parallelism does a lot of things simultaneously.

6. Concurrency: the non-deterministic control flow approach.

Parallelism: the deterministic control flow approach.

7. In concurrency debugging is very hard. But in Parallelism debugging is simplier.

1. What do these stand for in programming: DRY, KISS, BDUF

DRY (don't repeat yourself): software engineers to do something once, and only once.

KISS (Keep it simple, stupid): designs and systems should be as simple as possible.

BDUF (Big Design Up Front): the program's design is to be completed and perfected before that program's implementation is started.

1. What is Garbage collector? How does it work?

The garbage collector is keeping track of all objects in memory. It helps you avoid memory leaks by detecting circular references and destroy objects appropriately.

How it works:

Keep your Python objects being referenced, or they will be released in memory. The Python garbage collector initiates its execution with the program and is activated if the reference count falls to zero.

When we assign the new name or place it in containers such as a dictionary, the reference count increases its value. If we reassign the reference to an object, the reference counts decreases its value.

1. What are ‘deadlock’ and ‘livelock’ in a relational database?

‘deadlock’ is a situation that occurs when processes block each other with resource acquisition and makes no further progress. ‘livelock’ is similar, processes block each other with a repeat state change, but hasn’t made any progress.

1. What is Flask and what can we use it for?

Flask is a small Python web framework that provides useful tools and features to create web applications. It is more flexible, useful and accessible for new developers to build a web application quickly by only using one Python file.

**2. Discuss the difference between Python 2 and Python 3**

Python 3 syntax is simpler and easily understandable and Python 2 syntax is more difficult to understand.

Python 3 offers Range() function to perform iterations, but in Python 2, the xrange() is used for iterations.

Python 3 default storing of strings is Unicode but Python 2 stores need to define Unicode string value with “u.”

Python 3 value of variables never changes whereas in Python 2 value of the global variable will be changed while using it inside the for-loop.

Python 3 rules of ordering comparisons are simplified whereas Python 2 rules of ordering comparison are complex.

Python 3 exceptions should be enclosed in parenthesis while Python 2 exceptions should be enclosed in notations.

**3. Write a function that can define whether a word is a Palindrome or**

**not (a word, phrase, or sequence that reads the same backwards**

**as forwards, e.g. *madam*)**

def if\_it\_is\_palindrome(word1):  
 if word1 == word1[::-1]:  
 return True  
 else:  
 return False  
  
x1 = input("Please input your word:")  
  
if if\_it\_is\_palindrome(x1) == True:  
 print("Yes, it is a Palindrome.")  
else:  
 print("No, it is not a Palindrome")

**4. Write tests for the newly created Palindrome function. Provide a**

**brief explanation for your test case options.**

class TestPalindromeFunction(TestCase):  
 def It\_is\_right(self):  
 expected = True  
 result = if\_it\_is\_palindrome(word1='madam')  
 self.assertEqual(expected, result)  
  
 def It\_is\_wrong(self):  
 expected = False  
 result = if\_it\_is\_palindrome(word1='sir')  
 self.assertEqual(expected, result)  
   
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

My explanation:

Because for my function, there are only two situations (yes or no), so I just need a case of yes(true) and another case of no(false), and those could include all the possible situations.

**5. Agile methodology, Scrum: name at least 3 types of meetings that**

**are exercised by Agile teams and describe the objective of each**

**meeting.**

1. Sprint planning

In order to ensure that everyone is on the same page before work begins.

The meeting is conducted by the product owner at the start of every sprint, after the Sprint review and the Sprint retrospective for the prior sprint.

The owner introduces the product backlog items, then identifies or reiterates the company’s goals and desired outcomes.

2. Daily standup

It gives the team a chance to review goals and address any potential bottlenecks.

The daily standup happens every day at an agreed-upon time and is done standing up.

3. Sprint review

It happens at the end of every sprint. The development team leads it and presents the work they accomplished during their sprint. It’s important for the work presented during this ceremony to be fully demonstrable.

**6. Exception handling in Python, explain what each of the following blocks means in the program flow:**

Try, except, else, finally

Those blocks are for checking possible bugs in a program.

Try: run to test a block of code to see if it would show the bugs.

Except: to see if it would output the expecting results.

Else: run this if there was no expectation, it includes other possible expecting results.

Finally: the Python would execute it no matter what happened.

**7. How can we connect a Python program (process) with a database? Explain how it works and how do we fetch / insert data into DB tables from a python program.**

We need a python file called config.py to storage the HOST, USER and PASSWORD of our MySql.

And we should create another python file and using these codes:

import mysql.connector  
from config import USER, PASSWORD, HOST  
  
def \_connect\_to\_db(db\_name):  
 cnx = mysql.connector.connect(  
 host=HOST,  
 user=USER,  
 password=PASSWORD,  
 auth\_plugin='mysql\_native\_password',  
 database=db\_name  
 )  
 return cnx

def get\_all\_records():  
 try:  
 db\_name = 'tests'   
 db\_connection = \_connect\_to\_db(db\_name)  
 cur = db\_connection.cursor()  
 print("Connected to DB: {}".format(db\_name))  
  
 query = """SELECT \* FROM abcdefgh"""  
 cur.execute(query)  
 results = cur.fetchall()  
  
 for i in results:  
 print(i)  
  
 cur.close()  
  
 except Exception:  
 raise DbConnectionError('Failed to read data')  
  
 finally:  
 if cur:  
 cur.close()  
  
 if db\_connection:  
 db\_connection.close()  
 print("DB connection closed")

def main():  
 pass  
 get\_all\_records()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Before we run the program in Python, we need to run that DB in MySQL first, and then run these Python codes.

Fetch / Insert data into DB tables from a python program is similar: (I will take an saling DB as an example):

import pandas as pd  
import pymysql  
  
filepath = 'C:\ProgramData.xls'  
data = pd.read\_excel(filepath)  
db = pymysql.connect('localhost','root','1234','python\_analysis')  
cursor = db.cursor()  
try:  
 cursor.execute('create table ProgramData(num int primary key,date datetime, sale float )')  
except:  
 print('The DB has been created')  
  
query = """insert into catering\_sale (num, date, sale) values (%s,%s,%s)"""  
  
for r in range(0, len(data)):  
 num = data.ix[r,0]  
 date = data.ix[r,1]  
 sale = data.ix[r,2]  
 values = (int(num), str(date), float(sale))  
 cursor.execute(query, values)  
  
cursor.close()  
db.commit()  
db.close()  
  
db = pymysql.connect('localhost','root','1234','python\_anylysis')  
cursor = db.cursor()  
cursor.execute('''select \* from PythonData''')  
results = cursor.fetchall()  
  
for row in results:  
 print(row)  
  
cursor.close()  
db.commit()  
db.close()

**8. Given two SQL tables below: authors and books.**

**● The authors dataset has 1M+ rows**

**● The books dataset also has 1M+ rows**

Create an SQL query that shows the TOP 3 authors who sold the

most books in total!

SELECT BOOKS.book\_name, BOOKS.sold\_copies, AUTHORS.author\_name, sum(s.sold\_copies)

FROM AUTHORS

right join BOOKS on BOOKS.book\_name=AUTHORS.book\_name

group by s.author\_name

order by sum(s.sold\_copies)

where rank < 3