|  |  |
| --- | --- |
| Module Code | B8IT105 |
| Module Name | Programming for Big Data |
| Date | 02/06/2020 |
| Student number | 10544739 |
| Student name | Thomas Higgins |

*By uploading this exam from my Moodle account I Thomas am confirming that this document is all my own work.*

*I understand that DBS will carry out checks such as text-matching (via Urkund), bench-marking and viva voce exams in order to verify the authenticity of submissions.*

Please input your answers below. You may answer the questions in any order but you must ensure they are clearly labelled.

Question 1

# a

attach(ToothGrowth)

ToothGrowth

?ToothGrowth

# 60 Treatment observations in the dataset

# b

str(ToothGrowth)

# data is structured as a dataframe conbtaing 60 observations of 3 variables

# Dose variable is measured in mg/day

# c

mean(ToothGrowth$len)

# Mean tooth length is 18.81333

sd(ToothGrowth$len)

# standard deviation of tooth length is 7.649315

# d

# e

ToothGrowth$treat=with(ToothGrowth,interaction(supp,dose))

model=lm(len~treat,data=ToothGrowth)

anova(model)

# f

Question 2

# Q2

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# A

#load dataset

dataset = pd.read\_csv("C:/Users/GBKXN/OneDrive - Bayer/Code/github/B8IT105/exam/mtcars.csv")

#examine dataset

dataset.shape

# 32 Variables & 12 observations

# B

# Produce a scatterplot of the horse power (on the x-axis) against the miles per gallon variable (y-axis).

dataset.plot(kind='scatter',x='hp',y='mpg')

#C

# Use the type='n' argument (or otherwise) to help you to create a

# scatterplot of the horse-power variable against the miles per gallon

# variable where there are three distinct groups in the plot, depending

# on the cyl (number of cylinders) type of the cars.

# You should:

# ● Select a different plotting character than the default pch

# ● Colour the three groups differently

# ● Include a legend to explain these groups

# ● Include sensible x-axis and y-axis labels and a main title

# ● Rotate the numbers on the y-axis so they appear horizontal

# Comment on the resulting graph.

dataset.plot(kind='scatter',x='hp',y='mpg')

# We were not shown how to do this

Question 3

class Employee(object):

def \_\_init\_\_(self,name,salary=0):

self.\_\_name = name

self.\_\_salary = salary

def get\_name(self):

return self.\_\_name

def get\_salary(self):

return self.\_\_salary

def displayEmployee(self):

print('Name: {0}, Salary: {1}'.format(self.\_\_name,self.\_\_salary))

def salary\_increase(self,increase):

additional = (self.\_\_salary/100)\*increase

self.\_\_salary += additional

print('{0} salary increased by {1}%'.format(self.\_\_name,increase))

class ContractEmployee(object):

def \_\_init\_\_(self,name,hourly):

Employee.\_\_init\_\_(self,name)

self.\_\_hourly = hourly

def get\_hourly(self):

return self.\_\_hourly

Bob = Employee('Bob',10000)

Bob.displayEmployee()

Bob.salary\_increase(10)

Bob.displayEmployee()

import unittest

from Employee import \*

class TestQ3(unittest.TestCase):

def test\_employee(self):

test = Employee('Test',25000)

self.assertIsNone(test.displayEmployee())

self.assertEqual('Test',test.get\_name())

self.assertEqual(25000,test.get\_salary())

test.salary\_increase(10)

self.assertEqual(27500,test.get\_salary())

def test\_contract\_employee(self):

test = ContractEmployee('Test',15)

self.assertEqual(15,test.get\_hourly())

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

Question 4

A

class StringUtility(object):

def count\_vowels(words):

vowels = ['a','e','i','o','u']

count = 0

for letter in words:

if letter in vowels:

count+=1

return count

def unique\_vowels(words):

vowels = ['a','e','i','o','u']

count = 0

unique = []

for letter in words:

if letter in vowels:

if letter not in unique:

unique.append(letter)

return unique

def count\_spaces(words):

count = 0

for letter in words:

if letter ==' ':

count+=1

return count

def main():

test = StringUtility

text = 'How now a brown cow'

print(test.count\_vowels(text))

print(test.unique\_vowels(text))

print(test.count\_spaces(text))

main()

B

Beautiful soup

import requests

from bs4 import BeautifulSoup

headers = {

'authority': 'en.wikipedia.org',

'cache-control': 'max-age=0',

'upgrade-insecure-requests': '1',

'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.61 Safari/537.36',

'accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9',

'sec-fetch-site': 'none',

'sec-fetch-mode': 'navigate',

'sec-fetch-user': '?1',

'sec-fetch-dest': 'document',

'accept-language': 'en-GB,en;q=0.9',

'cookie': 'GeoIP=:::47.00:8.00:v4; forceHTTPS=true; enwikimwuser-sessionId=f0dabff1601beea9f1cf; WMF-Last-Access=02-Jun-2020; WMF-Last-Access-Global=02-Jun-2020; enwikiSession=cc7fm862kdf5jhi879d0io5k7756qhms',

'if-modified-since': 'Tue, 02 Jun 2020 18:14:27 GMT',

}

response = requests.get('https://en.wikipedia.org/wiki/COVID-19\_pandemic\_in\_the\_Republic\_of\_Ireland', headers=headers)

soup = BeautifulSoup(response.content, features="html.parser")

table\_count = 0

for table in soup.find\_all('table'):

# print(table)

# print(table.caption)

table\_count+=1

print(table\_count)

C

Evaluate the Big-O classification for the following functions.

i. O(n^3)

ii. O(log n)

iii. O(n^5)

iv. O(n^2)

v. O(log n)

vi. O(n)