**Question 1.**

1. Running the Python code given below will tell you the type of each of these Python expressions. For each expression, briefly explain why the resulting type arises. *The first one has been done for you as an example.*

print( type( 1+1 ) ) # Example: int + int always gives int

#---

print( type( 1/1 ) )

print( type( 1\*\*1 ) )

print( type( '1'\*1 ) )

print( type( hash(float('inf')) ) )

print( type( {bool('0'): None}[True] ) )

|  |
| --- |
| print( type( 1+1 ) ) # Example: int + int always gives int  print( type( 1/1 ) ) # int / int always gives float  print( type( 1\*\*1 ) ) # int \*\* int always gives int  print( type( '1'\*1 ) ) # str \* int always gives str  print( type( hash(float('inf')) ) ) # hash(float('inf')) -float(‘inf’) gives inf and hash(inf) results 314159,which type is an int  print( type( {bool('0'): None}[True] ) ) #{bool('0'): None}[True] results none,which type is NoneType |

1. The Python code given below checks whether or not a given positive integer is the sum of two integer squares. Modify the Python code below to check whether or not a given positive integer is the sum of three integer squares. Use this to find a positive integer that is not a sum of three integer squares.

def sum\_of\_two\_squares(target):

upper = int(target\*\*0.5) + 1

found = False

for a in range(0,upper):

for b in range(a,upper):

if (a\*\*2+b\*\*2==target):

print(target,'=',a,'\*\*2 +',b,'\*\*2')

found = True

return(found)

print(sum\_of\_two\_squares(1))

print(sum\_of\_two\_squares(2450))

|  |
| --- |
| code below to check whether or not a given positive integer is the sum of three integer squares:  **def** **sum\_of\_three\_squares**(target):  upper = int(target\*\***0.5**) + **1**  found = False  **for** a **in** range(**0**,upper):  **for** b **in** range(a,upper):  **for** c **in** range(b,upper):  **if** (a\*\***2**+b\*\***2**+c\*\***2**==target):  **print**(target,'=',a,'\*\*2 +',b,'\*\*2 +',c,'\*\*2')  found = True  **return**(found)  **print**(sum\_of\_three\_squares(**1**))  **print**(sum\_of\_three\_squares(**2450**))  output:  1 = 0 \*\*2 + 0 \*\*2 + 1 \*\*2  True  2450 = 0 \*\*2 + 7 \*\*2 + 49 \*\*2  2450 = 0 \*\*2 + 35 \*\*2 + 35 \*\*2  2450 = 3 \*\*2 + 29 \*\*2 + 40 \*\*2  2450 = 4 \*\*2 + 15 \*\*2 + 47 \*\*2  2450 = 5 \*\*2 + 11 \*\*2 + 48 \*\*2  2450 = 5 \*\*2 + 20 \*\*2 + 45 \*\*2  2450 = 5 \*\*2 + 24 \*\*2 + 43 \*\*2  2450 = 8 \*\*2 + 19 \*\*2 + 45 \*\*2  2450 = 11 \*\*2 + 27 \*\*2 + 40 \*\*2  2450 = 12 \*\*2 + 25 \*\*2 + 41 \*\*2  2450 = 13 \*\*2 + 16 \*\*2 + 45 \*\*2  2450 = 15 \*\*2 + 17 \*\*2 + 44 \*\*2  2450 = 15 \*\*2 + 25 \*\*2 + 40 \*\*2  2450 = 20 \*\*2 + 23 \*\*2 + 39 \*\*2  2450 = 20 \*\*2 + 31 \*\*2 + 33 \*\*2  2450 = 21 \*\*2 + 28 \*\*2 + 35 \*\*2  2450 = 23 \*\*2 + 25 \*\*2 + 36 \*\*2  True  find a positive integer that is not a sum of three integer squares:  given 0 to 20 numbers in a for loop to give input to the function which find the sum of three squares and returns the output of each.  In 0 to 19 numbers – 7,15 are not a sum of three integer squares  Code:  **def** **sum\_of\_three\_squares**(target):  upper = int(target\*\***0.5**) + **1**  found = False  **for** a **in** range(**0**,upper):  **for** b **in** range(a,upper):  **for** c **in** range(b,upper):  **if** (a\*\***2**+b\*\***2**+c\*\***2**==target):  **print**(target,'=',a,'\*\*2 +',b,'\*\*2 +',c,'\*\*2')  found = True  **return**(found)  **for** i **in** range(**20**):  result = sum\_of\_three\_squares(i)  **if**(result == True):  **print**(i, ' is positive integer that is a sum of three integer squares')  **else**:  **print**(i, ' is positive integer that is not a sum of three integer squares')  output:  0 = 0 \*\*2 + 0 \*\*2 + 0 \*\*2  0 is positive integer that is a sum of three integer squares  1 = 0 \*\*2 + 0 \*\*2 + 1 \*\*2  1 is positive integer that is a sum of three integer squares  2 = 0 \*\*2 + 1 \*\*2 + 1 \*\*2  2 is positive integer that is a sum of three integer squares  3 = 1 \*\*2 + 1 \*\*2 + 1 \*\*2  3 is positive integer that is a sum of three integer squares  4 = 0 \*\*2 + 0 \*\*2 + 2 \*\*2  4 is positive integer that is a sum of three integer squares  5 = 0 \*\*2 + 1 \*\*2 + 2 \*\*2  5 is positive integer that is a sum of three integer squares  6 = 1 \*\*2 + 1 \*\*2 + 2 \*\*2  6 is positive integer that is a sum of three integer squares  7 is positive integer that is not a sum of three integer squares  8 = 0 \*\*2 + 2 \*\*2 + 2 \*\*2  8 is positive integer that is a sum of three integer squares  9 = 0 \*\*2 + 0 \*\*2 + 3 \*\*2  9 = 1 \*\*2 + 2 \*\*2 + 2 \*\*2  9 is positive integer that is a sum of three integer squares  10 = 0 \*\*2 + 1 \*\*2 + 3 \*\*2  10 is positive integer that is a sum of three integer squares  11 = 1 \*\*2 + 1 \*\*2 + 3 \*\*2  11 is positive integer that is a sum of three integer squares  12 = 2 \*\*2 + 2 \*\*2 + 2 \*\*2  12 is positive integer that is a sum of three integer squares  13 = 0 \*\*2 + 2 \*\*2 + 3 \*\*2  13 is positive integer that is a sum of three integer squares  14 = 1 \*\*2 + 2 \*\*2 + 3 \*\*2  14 is positive integer that is a sum of three integer squares  15 is positive integer that is not a sum of three integer squares  16 = 0 \*\*2 + 0 \*\*2 + 4 \*\*2  16 is positive integer that is a sum of three integer squares  17 = 0 \*\*2 + 1 \*\*2 + 4 \*\*2  17 = 2 \*\*2 + 2 \*\*2 + 3 \*\*2  17 is positive integer that is a sum of three integer squares  18 = 0 \*\*2 + 3 \*\*2 + 3 \*\*2  18 = 1 \*\*2 + 1 \*\*2 + 4 \*\*2  18 is positive integer that is a sum of three integer squares  19 = 1 \*\*2 + 3 \*\*2 + 3 \*\*2  19 is positive integer that is a sum of three integer squares |

1. The Python functions int() and str() can be used to convert between integers and strings. Here are some examples.

A = str(123)

B = int('123')

Complete the Python function given below to return the sum of the squares of the digits of a given positive integer. For example, 389 should give . What conclusion can you draw from the output of the completed Python code (particularly the for loop given below)? *Include your Python code in your answer.*

def sum\_squared\_digits(n):

sum = 0

# -- add your Python code here --

return(sum)

a = 389

for i in range(12):

print(a,end=', ')

a = sum\_squared\_digits(a)

print(a)

|  |
| --- |
| **def** **sum\_squared\_digits**(n):  sum = **0**  N = str(n)  **for** i **in** range(len(N)):  m = int(N[i])  sum = sum + m\*\***2**  **return**(sum)    a = **389**  **for** i **in** range(**12**):  **print**(a,end=', ')  a = sum\_squared\_digits(a)  **print**(a)    output:  389, 154, 42, 20, 4, 16, 37, 58, 89, 145, 42, 20, 4  In the given code, there is a for loop with range(12),which means the program starts from i=0 and continue till i=12.In the for loop the function call continues for 13 times, the input of the 1st exection of function is a=389 and output is 154 ( , and the output 154 is the input of 2nd execution of the function, and the output of 2nd function call is the input of 3rd function call and soon. |

**Question 2.**

1. Consider the Python function given below. Write a suitable docstring that explains what the function does. *Give the docstring only. Do not attempt to explain how this code works.*

import sympy

def lucky():

keep\_going = True

x = 1

while (keep\_going):

p = x\*\*2 - x + 41

if (sympy.isprime(p)):

print(x,p,True)

else:

print(x,p,False)

keep\_going = False

x += 1

|  |
| --- |
| """keep printing the prime numbers using a 2nd order linear equation X\*\*2-X+41 with respect to X=1(incrementing by 1 for every itteration)  untill the result of the equation is not prime """ |
|  |

1. Consider the following variant of the dice game “Crag”. A single player starts with £10. After rolling three fair 6-sided dice, the player wins or loses money depending on the outcome (see below). The player continues to roll dice (three at a time) until they have no money left (they are “ruined”).

Win £10 Crag: any combination containing a pair (any two dice the same)

and totalling 13, e.g., 6-6-1

Win £3 Thirteen: any other combination totalling 13, e.g., 6-4-3

Win £2 Triple: all dice the same, e.g., 5-5-5

Win £1 Straight: low straight (1-2-3), high straight (4-5-6), or odd straight (1-3-5)

Lose £1 Any other combination

Write a Python function to implement the dice game described above. Your Python code should follow the partial code outline given below. The only output should be a count of the number of dice rolls and a count of the number of Crags rolled. *In your answer please include your Python code and the output (on one line) from one run.*

def play\_crag(money):

keep\_going = True

while (keep\_going):

# roll the three dice

if ( *\*\*something\*\** ):

money = money + 10

elif ( *\*\*something\*\** ):

money = money + 3

elif ( *\*\*something\*\** ):

money = money + 2

elif ( *\*\*something\*\** ):

money = money + 1

else:

money = money - 1

if (money <= 0):

keep\_going = False

|  |
| --- |
| **import** **random**  **def** **play\_crag**(money):  keep\_going = True  no\_of\_rolls = **0**  crags\_count = **0**  **while** (keep\_going):  # roll the three dice  p = random.randint(**0**,**6**)  q = random.randint(**0**,**6**)  r = random.randint(**0**,**6**)  no\_of\_rolls += **1**  **if** ((p+q+r==**13**) **and** (p==q **or** q==r **or** r==p)):  money = money + **10**  crags\_count += **1**  **elif** (p+q+r == **13** ):  money = money + **3**  **elif** ( p==q==r ):  money = money + **2**  **elif** ((p==**1** **and** q==**2** **and** r==**3**) **or** (p==**4** **and** q==**5** **and** r==**6**) **or** (p==**1** **and** q==**3** **and** r==**5**)):  money = money + **1**  **else**:  money = money - **1**  **if** (money <= **0**):  keep\_going = False  **print**('number of times dice rolled' , no\_of\_rolls ,', number of crags collected ',crags\_count )  play\_crag(**10**)  output:  number of times dice rolled 112 , number of crags collected 8 |

**Question 3.**

1. Run Length Encoding (RLE) is a simple method for data compression which does not lose any information. Suppose we are given a string in which the same character is often repeated several times. We can replace each “run” of a character by that character followed by the length of that run. For example, if then applying RLE gives .

The following Python code has been suggested to implement RLE. Indicate what is wrong with the output produced by this code. Then modify the Python code so that it implements RLE correctly. *Include both Python code and output in your answer.*

def encode(S):

R = ''

previous = ''

count = 0

for i in range(len(S)):

current = S[i]

if (current==previous):

count = count+1

else:

R = R+previous+str(count)

count = 0

previous = current

print(R)

print(encode('aaaaabbbbcc'))

|  |
| --- |
| Output produced with the given code:  0a4b3  None  As we are giving “aaaaabbbbcc“ as input to encode function , we have to get a5b4c2 as output.  As we are not assigning empty value to previous we are unable to go count 1st value of a and b and also C. using print(R) we can print the value in R but At the end of the function we need to return R to get the output returned to the function call, as it is not returning anything we got None printed at the function call.  Corrected code:  **def** **encode**(S):  R = ''  previous = S[**0**]  count = **0**    **for** i **in** range(len(S)):  current = S[i]  **if** (current==previous):  count = count+**1**    **else**:  R = R+previous+str(count)  count = **1**  previous = current  R = R+previous+str(count)  **return** R  **print**(encode('aaaaabbbbcc'))  output:  a5b4c2 |

1. In set theory, the “Universal set” is often written as . If is any subset of then the “complement of ” (written as ) is all the elements in that are not in . In the Python code given below, we are given the particular sets , and and we use Python set operators and set methods to produce , , (union) and (intersection).

# Given universal set U and subsets A and B

U = set('abcdefghijklmnopqrstuvwxyz')

A = {'a','b','c','d','e','f','g','h','i'}

B = {'a','e','i','o','u'}

# Complements of A and B

Acomp = U-A

Bcomp = U-B

# Union and intersection of A and B

AunionB = set.union(A,B)

AintB = set.intersection(A,B)

Write Python code to show that is a subset of . Also write Python code to investigate whether (the complement of “”) is related to (and make a clear conclusion). *Include both Python code and output in your answer.*

|  |
| --- |
| #A∩B is a subset of A∪B  check= set.issubset(AintB, AunionB)  **print**('A∩B is a subset of A∪B is :',check)  output:  A∩B is a subset of A∪B is : True  # whether ¯(A∪B) (the complement of “A∪B”) is related to ¯A∩¯B  AUB\_c = U-AunionB  **print**("AUB whole complement",AUB\_c)  A\_cintB\_c = set.intersection(Acomp, Bcomp)  **print**("A Complement int B complement : ",A\_cintB\_c)  **if** (AUB\_c==A\_cintB\_c):  **print**( "A∪B\_c (the complement of “A∪B”) is equal to ¯A∩¯B ")  output:  AUB whole complement {'w', 'k', 's', 'v', 'x', 'j', 't', 'y', 'l', 'm', 'r', 'p', 'z', 'q', 'n'}  A Complement int B complement : {'w', 'x', 's', 'v', 'j', 't', 'y', 'l', 'm', 'r', 'p', 'z', 'n', 'q', 'k'}  A∪B\_c (the complement of “A∪B”) is equal to ¯A∩¯B |

1. The two Python dictionaries given below show the highest revenue films of all time (adjusted for inflation). The first dictionary gives the worldwide revenue (adjusted to 2020 US dollars). The second dictionary gives the name of the director of each film.

F = {'Avatar': 3273000000,  
 'Avengers - Endgame': 2811000000,

'Doctor Zhivago': 2244000000,

'E.T. the Extra-Terrestrial': 2501000000,

'Gone with the Wind': 3724000000,

'Star Wars - A New Hope': 3059000000,

'Star Wars - The Force Awakens': 2213000000,

'The Sound of Music': 2562000000,

'The Ten Commandments': 2368000000,

'Titanic': 3096000000}

D = {'Avatar': 'James Cameron',

'Avengers - Endgame': 'Joe and Anthony Russo',

'Doctor Zhivago': 'David Lean',

'E.T. the Extra-Terrestrial': 'Steven Spielberg',

'Gone with the Wind': 'Victor Fleming',

'Star Wars - A New Hope': 'George Lucas',

'Star Wars - The Force Awakens': 'J.J. Abrams',

'The Sound of Music': 'Robert Wise',

'The Ten Commandments': 'Cecil B. DeMille',

'Titanic': 'James Cameron'}

Write Python code to print out the name of each film, the director that film and the worldwide revenue of that film in US$billions (one line per film), in the format below. *Include both Python code and output in your answer.*

Film: Avatar Director: James Cameron Revenue: 3.273

|  |
| --- |
| **for** key,value **in** D.items():  revenue = F[key]/**1000000000**  **print**("Filim: ",key,"**\t**Director: ",value,"Revenue: ",revenue)  output:  Film: Avatar Director: James Cameron Revenue: 3.273  Film: Avengers - Endgame Director: Joe and Anthony Russo Revenue: 2.811  Film: Doctor Zhivago Director: David Lean Revenue: 2.244  Film: E.T. the Extra-Terrestrial Director: Steven Spielberg Revenue: 2.501  Film: Gone with the Wind Director: Victor Fleming Revenue: 3.724  Film: Star Wars - A New Hope Director: George Lucas Revenue: 3.059  Film: Star Wars - The Force Awakens Director: J.J. Abrams Revenue: 2.213  Film: The Sound of Music Director: Robert Wise Revenue: 2.562  Film: The Ten Commandments Director: Cecil B. DeMille Revenue: 2.368  Film: Titanic Director: James Cameron Revenue: 3.096 |

1. Considering the same two dictionaries in part (c), write Python code to build (and print) a new dictionary, where the key is name of the director and the corresponding value is the total film revenue of films where that person what the director. *Include both Python code and output in your answer. You must not simply copy-and-paste values from the existing dictionaries.*

|  |
| --- |
| D\_R ={}  **for** key,values **in** D.items():  D\_R[D[key]] =F[key]  **print**(D\_R)  output:  {'James Cameron': 3096000000, 'Joe and Anthony Russo': 2811000000, 'David Lean': 2244000000, 'Steven Spielberg': 2501000000, 'Victor Fleming': 3724000000, 'George Lucas': 3059000000, 'J.J. Abrams': 2213000000, 'Robert Wise': 2562000000, 'Cecil B. DeMille': 2368000000} |

**Question 4.**

Use a small example to clearly explain the difference between the numpy functions numpy.tile() and numpy.repeat(). *Include both Python code and output in your answer.*

|  |
| --- |
| numpy.tile():The elements from the output are copied as ordered in input given and are repeated in the array.  Sample = [**7**,**3**,**8**]  np.tile(Sample,reps=**3**)  output:  array([7, 3, 8, 7, 3, 8, 7, 3, 8])  numpy.repeat():NumPy "repeat" function repeats element by element of an array .  np.repeat(a=Sample,repeats = **2**)  output:  array([7, 7, 3, 3, 8, 8]) |

1. The dataset *price* given below is extracted from the Road Fuel Prices statistics published weekly (<https://www.gov.uk/government/statistics/weekly-road-fuel-prices>) by the Department for Business, Energy and Industrial Strategy. It gives the price of petrol (pence per litre) at the pump recorded on one day every week for the two year period from 2 March 2020 to 28 February 2022.

import numpy as np

price = np.array([122.43, 122.24, 120.33, 119.64, 112.45, 110.23,

109.27, 108.63, 107.88, 107.56, 107.45, 105.09,

104.87, 105.17, 105.74, 106.37, 107.13, 108.31,

109.43, 111.24, 112.23, 112.66, 112.91, 113.02,

113.04, 113.22, 113.29, 113.37, 113.32, 113.31,

113.30, 113.26, 113.19, 113.18, 113.14, 113.11,

112.50, 112.35, 112.42, 112.61, 113.17, 113.82,

114.43, 114.91, 115.39, 116.14, 116.93, 118.10,

119.14, 119.67, 120.53, 121.27, 122.17, 122.94,

123.90, 124.57, 125.13, 125.24, 125.40, 125.48,

125.80, 126.09, 126.53, 127.19, 127.89, 128.15,

128.69, 129.47, 130.10, 130.73, 131.70, 132.47,

133.20, 133.48, 134.21, 134.70, 134.77, 134.68,

134.66, 134.76, 134.75, 134.86, 135.19, 136.10,

137.17, 139.46, 141.81, 143.70, 144.90, 145.87,

146.89, 147.53, 146.89, 146.22, 145.16, 144.92,

145.04, 144.82, 144.80, 144.87, 145.74, 146.33,

146.95, 147.77, 149.22])

Use a Python plotting library to plot a *line graph* showing the price of petrol over this time period. Add a horizontal line showing the average price over this time period. Make sure your plot has a title and suitable axis labels. *Include both Python code and your plot in your answer.*

|  |
| --- |
| **import** **matplotlib.pyplot** **as** **plt**  plt.figure()  plt.plot(price,'b--')  index = np.arange(**0**,len(price),**7**)  plt.plot(index,price[index],'r.',markersize=**12**)  plt.axhline(y=np.average(price), color='r', linestyle='-')  plt.title("weekly petrol prices in two years period - 2 March 2020 to 28 February 2022")  plt.xlabel("recorded petrol price once a week")  plt.ylabel("price of perol")  plt.show() |

1. Consider the following Python code applied to the petrol *price* dataset from part (b). Explain what the Python code does line-by-line. What 5-letter word would you recommend to replace the “@@@@@”?

price\_split = np.split(price[5:],10)

for i in range(len(price\_split)):

block = price\_split[i]

print('Block',i,'has price @@@@@',np.around(np.ptp(block),2))

|  |
| --- |
| Line 1: Leaving first 5 elements ,from 6th element to last element splitting the array to 10 element array block each  Line 2: for loop to access all the values in the price\_split list  line3: using index i from 0 assigning each value of price\_split to the block in the for loop till the len(price\_split)  line 4: Increase of price form peak to peak using np.ptp.  Rounding the output of np.ptp to 2 decimal points only np.around(np.ptp(np.array([106.37, 107.13, 108.31, 109.43, 111.24, 112.23, 112.66, 112.91, 113.02, 113.04])),2)  “Range” is the 5-letter word would you recommend to replace the “@@@@@”. |

**Question 5.**

Consider the dataset in the file “animals.csv” on the number of animals of different types in Argentina, Egypt, France and the UK (2001–2020). You can download this dataset from the Moodle site for 7143CEM. *Download the CSV file and save it in the same location as your Python code for this question.*

The data comes from the Food and Agriculture Organization of the United Nations (FAO) statistics <http://www.fao.org/faostat/>.

import pandas as pd

animals = pd.read\_csv('animals.csv')

For this question, you may use any plotting libraries you wish, but make sure each plot has a suitable title. You may also find the following Python code useful.

AAA = animals.pivot(index= box A , columns= box B , values= box C )

1. Write Python code to find the names of the columns in this dataset and the unique values of the each animal *type* in the dataset. *Include both Python code and output in your answer.*

|  |
| --- |
| **import** **pandas** **as** **pd**  animals = pd.read\_csv('animals.csv')  **print**(animals.columns)  animals.type.unique()  outpt:  Index(['country', 'type', 'year', 'unit', 'value'], dtype='object')  Out[319]:  array(['Cattle', 'Chickens', 'Ducks', 'Geese and guinea fowls', 'Goats',  'Horses', 'Pigs', 'Rabbits and hares', 'Sheep', 'Turkeys',  'Buffaloes', 'Camels'], dtype=object) |

1. Consider only the year 2020. Use Python to find the total number of animals in each *country.* Also use Python to construct a summary table giving the number of animals of each *type*, given in decreasing order of number of animals. *Take careful note of the unit column. Include both your code and the output in your answer.*

|  |
| --- |
| #Total number of animals in each country:  ani\_2020 = animals.query("year==2020")  ZZZ = ani\_2020.groupby('country')['type'].nunique()  **print**(ZZZ)  #Number of animals of each type in decreasing order:  aaa = ani\_2020.groupby(['type'])['value'].count().reset\_index().sort\_values('value', ascending=False)  **print**(aaa)  output: |