|  |  |  |
| --- | --- | --- |
|  | number of pages: **8** | lecturer: **Serhat Erdogan - Rob Coekaerts – Janne Gilis – Ruben Naudts – Bart Thumas** course unit (OPO/OLA): **Programming 2**  code OPO/OLA: **MBI04j – MBI04h**  The **standard duration** for this exam is **2h**, but each student will be given 30% extra time, so. **2h36 in total**. |
| date: …  group: … | | surname:  first name:  student number:  time of submission: ……………………………h……………………..min |
| aid: **None** | | marks …/**10** |

**Exam – instructions**

Welcome to the written exam for the course Programming 2. This exam is part of the assessment for this course and counts for 10 out of 20 points (50%). A student must pass this written exam in order to pass the complete course. The remaining 50% of the points will be earned during the practical exam.

Take your time to read and answer the questions carefully. Start by working out your answers on scratch paper and only write the final answer clearly and neatly on the exam sheet.

If you make a mistake during answering and run out of space on the exam sheet, inform the supervisor. You will receive an official UCLL sheet to write down your answer.

Be thorough and clear in your responses. Illegible text will not be graded and may result in a loss of points. As this is a written exam, we do not expect perfect syntax as in a programming environment, but make sure that the basic Python requirements are correctly followed.

**Good luck!**

**DISCLAIMER:**

This sample exam is presented in English. Please be aware that the final exam will be provided in the language corresponding to your enrolled program:

* Toegepaste Informatica: The final exam will be in Dutch.
* Applied Computer Science: The final exam will be in English.

**Question1 - Assume the following implementation of the class BMICalculator:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | class BMICalculator:  def \_\_init\_\_(self, weight\_in\_kg, height\_in\_m):  self.weight\_in\_kg = weight\_in\_kg  self.height\_in\_m = height\_in\_m  def bmi(self):  return self.weight\_in\_kg / self.height\_in\_m\*\*2 |

**We should be able to use this class as follows:**

|  |  |
| --- | --- |
| 9  10  11  12 | *# Usage:*  calc = BMICalculator (85, 1.78)  print(calc.bmi) *# Expected Output: 26.83* |

**However, when we execute our script, we get the following:**

|  |
| --- |
| <bound method BMICalculator.bmi of <\_\_main\_\_.BMICalculator object at 0x10847ee40>> |

**Explain what is wrong with the class BMICalculator and explain how we can fix this error?**

We are trying to print the bmi property, but bmi is defined as a method instead of a (calculated) property.

We can fix this by adding @property above def bmi(self):

**Question 2 – Given the following classes A, B and C:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | class A:  def show(self):  print("A.show()")  class B(A):  def show(self):  print("B.show()")  super().show()  class C(B):  def show(self):  super().show()  print("C.show()")  obj = C()  obj.show() |

**What will be the output of the code when we run our script? If the code results in an error, explain why the error occurs.**

B.show()

A.show()

C.show()

**Question 3 – Given the following piece of code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | def divisors(n):  result = []  for k in range(1, n + 1):  if n % k == 0:  result.append(k)  return result  print("Divisors of 28:", divisors(28))  *# Expected Output: Divisors of 28: [1, 2, 4, 7, 14, 28]* |

**Rewrite the function using a list comprehension (such that there are no more for-loops), making sure to keep the same functionality.**

def divisors(n):

return [k for k in range(1, n+1) if n % k == 0]

**Question 4 – Write a recursive method ‘countX’ that takes a text and returns the number of times the character ‘x’ (lower and upper case) appears in the text.**

|  |
| --- |
| print(countX("This is a text where we want to calculate how many x characters are present in the text."))  >>> 3 |

**Write the implementation of this method:**

def countX(text):

if not text:

return 0

first\_is\_x = 1 if text[0].lower() == 'x' else 0

return first\_is\_x + countX(text[1:])

**Question 5 – Consider the implementation of the class Book:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | class Book:  def \_\_init\_\_(self, title, author, year):  self.title = title  self.author = author  self.year = year  def \_\_str\_\_(self):  return f'{self.title} by {self.author} ({self.year})'  def ??? (self, other):  if not isinstance(other, Book):  raise RuntimeError  return self.year < other.year  dune = Book("Dune", "Frank Herbert", 1965)  jaws = Book("Jaws", "Peter Benchley", 1974)  *# Print oldest book*  if dune < jaws:  print(dune)  else:  print(jaws) |

**Replace the 3 question marks (???) in the code above with the necessary code such that the oldest book can be printed:**

\_\_lt\_\_

**Question 6 – Given is the following Pandas Series that contains the amount of animals of a certain type :**

|  |
| --- |
| import pandas as pd  animal\_counts = {  "Lion": 5,  "Elephant": 3,  "Giraffe": 7,  "Zebra": 12,  }  series\_animal\_counts = pd.Series(animal\_counts)  print(series\_animal\_counts) |

**Output:**

|  |
| --- |
| Lion 5  Elephant 3  Giraffe 7  Zebra 12  dtype: int64 |

**Write the code (using the principles learned in the data processing course) to change the count to 0 for animals where the current count is higher than 10**

series\_animal\_counts[series\_animal\_counts > 10] = 0

**Question 7 – Given the following Pandas DataFrame:**

|  |
| --- |
| import pandas as pd  data = [  ['Lionel Messi', 'Forward', 'Inter Miami', 30],  ['Cristiano Ronaldo', 'Forward', 'Al Nassr', 25],  ['Harry Maguire', 'Defender', 'Man Utd', 20],  ['Kylian Mbappe', 'Forward', 'PSG', 18],  ['Kevin De Bruyne', 'Midfielder', 'Manchester City', 22],  ['Virgil Van Dyck', 'Forward', 'Liverpool', 28],  ]  df = pd.DataFrame(data, columns=["Name", "Position", "Club", "Goals"])  print(df) |

**Its output:**

|  |
| --- |
| **Name Position Club Goals**  0 Lionel Messi Forward Inter Miami 30  1 Cristiano Ronaldo Forward Al Nassr 25  2 Harry Maguire Defender Man Utd 20  3 Kylian Mbappe Forward PSG 18  4 Kevin De Bruyne Midfielder Manchester City 22  5 Virgil Van Dyck Forward Liverpool 28 |

**What will be the output when we run the following code?**

|  |
| --- |
| df.loc[df['Name'] == 'Virgil Van Dyck', 'Position'] = 'Defender'  x = df[df["Position"] == "Defender"]  y = x["Goals"].min()  df = df[df["Goals"] > y]  group\_df = df.groupby("Position").sum("Goals")  print(group\_df.size) |

**Write the output of the code above. If the code results in an error, explain why the error occurs.**

3