OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Semester I Examinations 2022-2023
Solutions/Grading Scheme

Course Instance Code(s)	1MAI1, 1MAO3
Exam(s)	MSc in Computer Science (Artificial Intelligence), MSc in Computer Science (Artificial Intelligence) - Online
Module Code(s)	CT5132, CT5148
Module(s)	**Programming and Tools for Artificial Intelligence**
Discipline	School of Computer Science
Paper No.	1
External Examiner	Dr. John Woodward
Internal Examiner(s)	Prof. Michael Madden
	Dr. James McDermott *
Programme Coordinator(s)	Dr. Matthias Nickles, Dr. James McDermott
Instructions	**Answer all 4 questions. All are worth equal marks.**
Duration	2 Hours exam
Number of pages	10 (including this page)
Discipline	Computer Science
Requirements	
Release in Exam Venue	Yes ☒ No ☐
Release to Library	Yes ☒ No ☐
Calculator	Allowed (non-programmable)

Question 1: Basic Python

(a) Define a function named `palindrome` which accepts one string argument and tests whether it is palindromic (i.e. is the same whether read left-to-right or right-to-left) and returns a `bool`. Show how to call this function. **[5]**

Solution. One solution below, of course many others possible. [1 for defining a fn with an argument (not def initial('abcbc')) and return (not print). 2 for correct algorithm even if syntax is bad. 1 for calling it correctly outside.]

```
def palindrome(s):
  for x, y in zip(s, reversed(s)):
    if x != y:
      return False
  return True
palindrome('TENET') # RECOMMENDED MOVIE
```

Common error. A couple of people wrote:

```
def palindrome(s):
  s = input('Please enter a string')
```

But that ignores the `s` that is passed in! The spec states that the function should accept a string argument, not take input from the terminal.

Another common error was writing this inside the loop:

```
if s[0] == s[-1]: return True
```

That is incorrect, as it checks only the first and last character.

(b) Given the following data structure: **[5]**

```
x = [{'a': 1, 'b': 2}, {'a': 11, 'b': 12}, {'c': 100}]
```

Write an expression using `x` that will have the value `100`.

Solution `x[2]['c']`. **Grading**. 2 for something involving x and square brackets. 5 for completely correct.

(c) For the following function, write a docstring containing three doctests to demonstrate usage. State where the docstring should be placed. **[5]**

```
def validate(s):
    if len(s) != 8:
        return 'Bad length'
    if not s[0].isupper():
```

```
        return 'Initial uppercase required'
    return 'Ok'
```

Solution

```
    '''
    >>> validate('Abcdefgh')
    Ok
    >>> validate('abcdefgh)
    Initial uppercase required
    >>> validate('Abcd')
    Bad length
    '''
```

This should be directly after the first line, before the start of the body.

[2 marks for doctests. 2 for correct syntax. 1 for precise placement.]

(d) Suppose we have a `dict`, `d`. How can we create a new `dict` in which the keys are sorted alphabetically? You can answer with code or in (precise) English. **[5]**

Solution below.

```
def dict_sort(d):
    result = {}
    for k in sorted(d.keys()):
        # in modern Python, dicts remember the order in which their keys
        # were added, and use that order eg when being printed or iterated
        result[k] = d[k]
    return result
```

Comment. What does it mean to say the keys are sorted? In older Python this question wouldn't make sense: keys don't have any order. But in modern Python, a dictionary remembers the order of insertion of its keys. So the question is asking us for a dictionary where the keys have been inserted in alphabetical order. Slightly tricky! But we did take advantage of the idea in Assignment 1 (the treasure chest).

(e) In the following code, explain what happens when we call `len(c)`. **[5]**

```
class C:
    def __init__(self, data):
        self.data = data
    def __str__(self):
        return str(self.data)
    def __len__(self):
        return len(self.data)
```

```
c = C([4, 5, 6])
len(c)
```

Solution. c is an object of type C. len(c) is translated to `c.__len__()`, which results in `C.__len__(c)`, ie we have self equal to c now. Inside `__len__` we have len(c.data) which is len([4, 5, 6]) since [4, 5, 6] was saved as self.data in the constructor. len([4, 5, 6]) delegates to list's builtin `__len__` which returns 3.

Question 2: Advanced Python

(a) The following code will not print out the integers `[0, 1, 4, 9, 16]` as we might expect. Why not? How can we fix it? Why is `map` designed to work this way? **[5]**

```
map(lambda x: x**2, range(0, 5))
```

Solution. map returns an iterator, not a list [2]. We can enclose the expression in list(), as that will iterate the iterator to construct the list [1]. map doesn't create the list internally, to save space for large lists, and to in principle allow infinite maps [2].

Common error. Several people wrote that we have supplied a function and a function, whereas `map` expects a function and a list. But `range(0, 5)` is not a function, it's a function call! The result of a function call is not (usually) a function. Calling `map(lambda x: x**2, [0, 1, 2, 3, 4])` would make no difference.

Several people just forgot to explain why `map` is designed this way.

(b) What is the time complexity of the following code, with respect to the length `n` of the list `L`? How can we use a different data structure to reduce it? **[5]**

```
def dupe(L): # detect duplicates
    n = len(L)
    for i in range(n):
        for j in range(i+1, n):
            if L[i] == L[j]:
                return True
    return False
```

Solution. This is $O(n^2)$ [3]. We can use a `set` [1], eg for x in L, if x already in set S then return True, else add x to S [1].

(c) Explain in simple terms how the `dict` data structure achieves $O(1)$ lookup performance. Consider only a typical lookup, ignoring the issue of collisions. **[5]**

Solution. Define n as the number of entries in the dict [1]. To lookup a key k, it calculates hash(k) mod table size [1] to give the "slot" in the table [1]. Underlying this is an array with O(1) indexing [2].

Common error. Several people responded that the dict uses a hash lookup in a table, but didn't explain the mod (to ensure we find a slot inside the table) or didn't state that the hash and the mod are both O(1). A few people said that Python uses a list internally, but it's not quite true – it uses more efficient data handling – we just used that for illustration. But no marks were subtracted for that.

(d) Without using any import, implement a *memoised* version of this function. **[5]**

```
def fib(n):
  if n <= 1:
    return 1
  else:
    return n * fib(n - 1)
```

Solution below. 10 for any complete solution. Partial marks for creating a cache, attempt to read the cache, attempt to write the cache.

```
def make_fib():
    cache = {}
    def fib(n):
        if n in cache: return cache[n]
        if n <= 1:
            cache[n] = 1
            return cache[n]
        else:
            cache[n] = n * fib(n-1)
            return cache[n]
    return fib
fib = make_fib()
```

Common error. Many people added the cache and the logic, but by putting the `cache = {}` inside `fib`, it is always re-initialised every time the function is called. If we want cache to be *updated* every time instead, it has to created outside the function. This can just be outside the function, or a neater solution is using a closure as in the code above. An even neater solution is to write a generic function memoise(f) which can work on any function f (see lecture materials) but this was not required for full marks.

Question 3: Data Science

(a) Consider Figure 1, below. Given only the two end-point values x_0 and x_n and the value n, how can we create a Numpy array of the values $(x_0, x_1, \ldots, x_{n-1})$? **[5]**

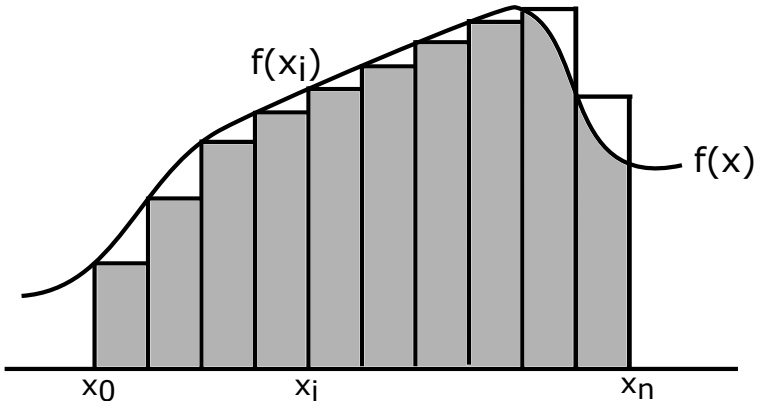

Figure 1

Solution. We can use linspace, specifically `np.linspace(x0, xn, n, endpoint=False)`. Award 5 marks for any very close solution, even omitting endpoint or trying (xn - x0) / n. Award 2 just for writing 'linspace'.

(b) Referring again to Figure 1, above: suppose we are given a Python function `f`, and a Numpy array `x` of the values $(x_0, x_1, \ldots, x_{n-1})$. We can approximate the area under the curve, between x_0 and x_n, by summing the area of rectangles as shown. The height of the ith rectangle is given by $f(x_i)$ as shown. Write a snippet of Python code or pseudocode, or explain in precise English how we can calculate this sum. **[10]**

Solution. The solution is short but I expect this requires some thinking.

```
# there are n-1 slices from x_0 to x_n-1
w = (x[-1] - x[0]) / len(x)
result = (w * f(x)).sum()
```

2 marks for concept of rectangle width, and 1 extra if correct. 2 marks for concept of f(x), and 2 extra if correct. 1 mark for w * f(x). 1 mark for sum. 1 mark for fully correct. The above solution uses Numpy fully to avoid writing any loop. Several students gave solutions using list comprehensions or maps, etc., and these received full marks. However, understanding how to use Numpy vectorisation is crucial. "Of course someone has to write loops. It doesn't have to be you."

(c) What is a *mixin*? Explain with reference to an example in Scikit-Learn. **[5]**

Solution. A mixin is a class designed to be inherited from [1], in multiple inheritance [1]. In sklearn an example is the ClassifierMixin [1]. A typical classifier inherits from BaseEstimator

and ClassifierMixin [1]. The latter provides a score function which calculates classification accuracy [1].

Common error. This was the question with the lowest average grade, because a lot of people forgot to answer it, or answered with something completely unrelated. For full marks, the point about multiple inheritance is required.

(d) For each of these R dplyr verbs, state its purpose in a few words: **[5]**

- `select`
- `filter`
- `mutate`
- `gather` (or `pivot_longer`, if you prefer)
- `inner_join`

Solution

- choose a subset of columns
- choose a subset of rows based on some criteria
- add or change columns
- rewrite in a tidy format with one column per variable, one observation per row (hard to write in a few words, generous here)
- take in two tables with a common variable and return a single combined table using only rows where both tables have the same value of the common variable (ok to write something shorter also)

Question 4: Tools and Applications

(a) Suppose we are given a *set* of variable assignments, such as:

- `x1 = x3 + 3`
- `x0 = x2 ** 2`
- `x2 = x3 + 1`
- `x3 = 2`

Draw a graph representing the dependencies between the variables. State a *topological ordering* of the variables which would allow us to calculate their values. **[5]**

Solution. Expect a graph showing 4 nodes [1] with directed edges [1], all correct [1]. Ordering: eg `x3, x1, x2, x0`. [2]

Notice that a topological ordering is a list of the nodes, eg as above. But eg `x3, (x1 and x2 in parallel), x0` is not a topological ordering. (Still received most of the marks.)

(b) For the following set, again draw a graph representing dependencies, and explain why no topological ordering is possible. **[5]**

- `x4 = 2 * x3`
- `x2 = x1 + 1`
- `x1 = x0 ** 2`
- `x0 = 5 * x2`
- `x3 = x2 + 1`

Solution. A correct digraph [2], explaining that there is a cycle [1], specifically x2 - x1 - x0 - x2 [2]. If a good explanation is given but the term "circular" or "cycle" or similar is not mentioned, up to 4 marks.

(c) State an algorithm for topological ordering, given a directed graph with nodes N and edges E. **[5]**

Solution. Create an empty list L. Find a node n with in-degree 0 (if none exists, the graph contains a cycle and we fail). Append n to the list L. Delete any edges from E in which n is the source node and delete the node n from N. Repeat from the word "Find" until N is empty. Return L.

(d) Given the graph shown in Figure 2 below, write out the *adjacency matrix* which represents the graph. Use the adjacency matrix to calculate the out-degree of each node. **[5]**

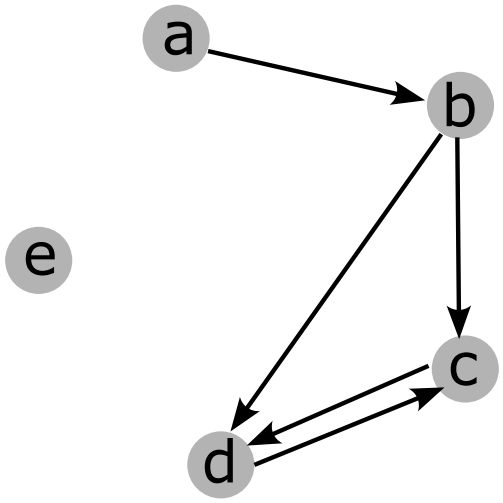

Figure 2

Solution. The matrix is shown below, with out-degrees calculated as row-sums on the right.

```
0 1 0 0 0    => 1
0 0 1 1 0    => 2
0 0 0 1 0    => 1
0 0 1 0 0    => 1
0 0 0 0 0    => 0
```

Give a mark for a binary square matrix [1], fully correct [3]. Correct out-degree calculated as a row-sum (or column-sum if they use that convention) [1].

(e) In Assignment 2, we constructed a *dissimilarity matrix*. What properties does a dissimilarity matrix have? **[5]**

Solution. Square [1], symmetric [2], zero on the diagonal [2].

Comment. Notice that we are not asked to reproduce any information from Assignment 2 – the question just tries to prompt you or remind you where you have seen dissimilarity matrixes before.