Semester I Examinations 2020/2021

Exam Code(s)	1MAI1, 1CSD1, 1SPE1, 2SPE1
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	Prof. Pier Luca Lanzi
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.** You may answer either: in a Word document or similar, and then `CT5132_CT5148_PTAI_Answer_Sheet.docx` is suggested; or on paper, uploading a scan of the pages. This is an **open-book** exam: you may **read** textbooks, notes, and **existing resources** on the internet. You may **not communicate** with anyone, in person, via phone, internet, or otherwise. You may **not post questions** on internet sites or elsewhere during the exam.
Duration	2 Hours exam plus 30 minutes for upload
Number of pages	7 (including this page)
Discipline	Computer Science

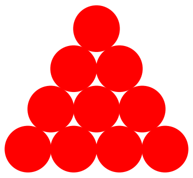
Requirements

Release in Exam Venue	[✓ Yes] No
Release to Library	[✓ Yes] No

Question 1: Basic Python

(a) Distinguish between a function that prints its result and a function that returns its result. What are the reasons for choosing one over the other? **[5]**

(b) Your friend wants to calculate the number of snooker balls in a triangular arrangement, where the last row is of length n, as shown. They have found some good code on the internet, but accidentally discarded indentation while copy-pasting. Fix this. **[5]**



```
def snooker(n):
if n == 0:
return 0
else:
return n + snooker(n - 1)
n = 5
m = snooker(n)
print(m)
```

(c) Your friend has written some code which calculates the number of followers for several Twitter accounts. It stores the result in a list, alternating account name and follower count, e.g.:

```
['StephenFry', 1000, 'BillGates', 2000, 'NYT', 1500]
```

Criticise this choice of data structure and suggest an alternative. **[5]**

(d) With reference to the variable x below, explain *duck typing* in Python. **[5]**

```
def count_zero(x):
    c = 0
    for a in x:
        if a == 0:
            c += 1
    return c
```

(e) What is the difference between these three pieces of code? Which is correct and why? **[5]**

```
import random
results = []
for repetition in range(100):
    random.seed(0)
    results.append(run_experiment())
```

```
import random
results = []
random.seed(0)
for repetition in range(100):
    results.append(run_experiment())
```

```
import random
results = []
for repetition in range(100):
    random.seed(repetition)
    results.append(run_experiment())
```

Question 2: Advanced Python

(a) What does this regular expression do? Explain with the help of some example strings. **[5]**

```
__\w*__\((.*)\):
```

(b) In the following code, at the line marked `Line 4`, which names are in scope: `random`, `randrange`, `main`, `f`, `x`, `y`, `z`? **[5]**

```
from random import randrange
def f(y):
    y = y ** 2
    # Line 4
    return y
z = f(2)
def main():
    x = 3
    print(f(x))
```

(c) Would the function `f` in part (b) be suitable for *memoisation*? Why or why not? **[5]**

(d) Suppose we are writing a networking program which stores its configuration in a `dict`, e.g.:

```
{'n_connections': 100, 'port': 8080}
```

Show how we can store such a configuration on disk and later read it back in to a `dict`, using `eval`. **[5]**

(e) What is the time complexity of the following function, with respect to the length of `text`? What can we say about complexity with respect to the length of `targets`? **[5]**

```
def count_letters(text, targets):
    # text is a long string, eg a book
    # targets is a list of letters a-z and A-Z which we wish to count
    count = 0
    for c in text: # one character at a time
        for t in targets:
            count += 1
    return count
```

Question 3: Data Science

(a) Given the array `a` below, write an array `b` such that `a * b` gives the result shown. Explain in your own words the rules for Numpy broadcasting using this example. **[5]**

```
a = np.array([[[ 1,  2,  3],
               [ 4,  5,  6]],

              [[ 7,  8,  9],
               [10, 11, 12]]])

# b = ?
a * b

array([[[  10,   20,   30],
        [ 400,  500,  600]],

       [[  70,   80,   90],
        [1000, 1100, 1200]]])
```

(b) In your own words, explain the significance of *whether or not* a Scikit-Learn `estimator` object contains a `_coef` variable (or another variable named with a leading underscore). **[5]**

(c) Rewrite the following R code using the `magrittr` pipe, where `D` is a tibble and `a` and `b` are columns. **[5]**

```
select(mutate(D, a=0.4*a, b=0.4*b), year, a, b)
```

(d) The following data is not *tidy.* Explain why not, and show what it would look like in tidy format. **[5]**

Parameters	n=3,m=3	n=3,m=4	n=4,m=3	n=4,m=4
Elapsed Time	122	150	143	190
Accuracy	90	95	87	91

(e) In the following data we wish to find the sum of sales per location. Using this example, explain the *group-by* mechanism (which exists in both R and Pandas). You don't need to write code or carry out calculations. **[5]**

Seller	Location	Day	Sales
A	Galway	Mon	1000
A	Galway	Tue	1100
A	Limerick	Mon	1200
A	Limerick	Tue	800
B	Galway	Mon	1100
B	Galway	Tue	1400
B	Limerick	Mon	1400
B	Limerick	Tue	1300
C	Galway	Mon	900
C	Galway	Tue	1000
C	Limerick	Mon	1000
C	Limerick	Tue	1500

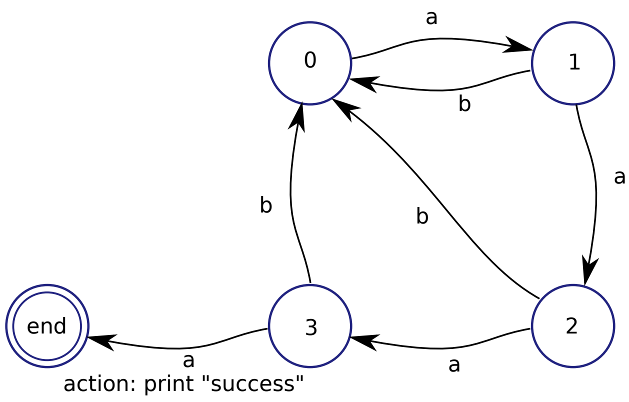
Question 4: Tools and Applications

(a) State the 4-grams in the following bitstring: `001000100011`. **[5]**

(b) Consider the grammar below, where `<expr>` is the start symbol. Show that it can generate some valid Python expressions, and can also generate some invalid ones. **[10]**

```
<expr>  ::= {<kvs>} | {<vs>}
<kvs>   ::= <key>: <val> | <key>: <val>, <kvs>
<vs>    ::= <val> | <val>, <vs> | <kvs>
<key>   ::= 'a' | 'b' | 'c' | 'd'
<val>   ::= 1 | 2 | 3 | 4 | 5 | 6 | 7
```

(c) In the finite state machine (FSM) shown below, the start state is state 0 and the end state is as indicated. Edges are labelled with inputs, and one edge also has an action as shown. Explain what this FSM does, with the help of some example inputs. Encode this FSM as Python code, either in the *State, input, state, action (SISA)* format, or another unambiguous format if you prefer. **[10]**



0
1
2
3
end
a
b
a
b
b
a
a
action: print "success"