Algebra University College
Department of Software Engineering

Programming in Python
Midterm Exam
November 22, 2021

Student Name: _____

University ID: _____

Outcome	O1	O2	O3	O5
Minimum Points	7	7	3	10
Desired Points	3	3	2	2
Total	**10**	**10**	**5**	**12**
Available Time	**25 min**	**25 min**	**12.5 min**	**30 min**

Total Points: 37 **Time: 92.5 min**

- Closed book exam - no use of the Internet or any other resource.
- Printed on both sides of the page.
- Answer on empty paper and code on the computer.
- Crib sheets not permitted.
- Exam should be returned, and code uploaded according to the instructions.
- Any special instructions: see next page.

Declaration of Academic Integrity

By signing below, I pledge that the answers and code of this exam are my own work without the assistance of others or the usage of unauthorized material or information.

_____
Signature

Question	Outcome	Points	Score
1	1	2	
2		2	
3		2	
4		4	
5	2	1	
6		3	
7		3	
8		3	

Question	Outcome	Points	Score
9	3	2	
10		3	
11	5	1	
12		5	
13		3	
14		3	
15	-	-	-
16	-	-	-

TOTAL SCORE: _____ EXAMINED BY: _____

Instructions:

- Make sure to write your name and university ID (Croatian: *jmbag*) in the first page, and to sign a *Declaration of Academic Integrity*.
- This question booklet consists of 5 pages. Make sure you have all of them.
- Keep quite during the exam. For assistance, raise your hand and an invigilator will come to see you.
- Answer the theory questions on the special empty paper.
- Use *Visual Studio Code* or *Notepad++* for coding. Use Python interpreter version *3.8* or higher.
- Each coding task must be solved in a separate folder named ***taskX***, where *X* is a task number. A script with an entry point must be named ***main.py***. All task folders must be in the folder named ***MidtermExam_22Nov_21***.
    - **MidtermExam_22Nov21**
        - **task1**
            - **main.py**
            - …
        - **task2**
            - **main.py**
            - …
        - …
        - **taskN**
            - **main.py**
            - …
- After finishing with coding, add the ***MidtermExam_22Nov_21*** folder to the **ZIP** archive named ***FirstName_LastName.zip***, and upload the zipped file to https://results.vua.cloud/:
    - Open the page https://results.vua.cloud/
    - Login using AAIEdu credentials
    - Enter username and password
        - username: same as for Infoeduka + add "@racunarstvo.hr"
        - password: same as for Infoeduka
    - Open the exam course folder. Be sure to upload to the right folder; otherwise, your exam will not be graded!
- To be graded, the code must have no syntax error (0 points otherwise).
- Finally:
    - Organize your work.
    - Mysterious or unsupported answers will not receive full credit.
    - Provide exact answers and simplify all answers as much as possible.
    - Use comments in the code and write a self-documentary code!
- Each question defines minimum and desired outcome points in format (minimum/desired points).
- Do not write in the table on the first page.

Best of luck!

Karlo Knežević

Learning outcome 1

1. (2/0 points)

a) (1/0 point) Explain the difference between the keyword *is* and the operator *==*.

b) (1/0 point) Why the following code prints *True* despite the operators' differences?

```
a = 0
b = 0
print((a is b) == (a == b))
```

2. (0/2 points) Visualize all the objects and references stored in memory after executing the following code (before running *garbage collector*). Clearly show all frames (*global* and *function*, visualize *return value*), render all objects on the heap and draw pointers as arrows.

```
def func(a, b):
    a = 5
    b[1] = a
a = -10
b = [1, True]
func(a, b)
```

3. (1/1 points) Using a *list comprehension*, programmatically create a list of tuples containing pairs of letters in alphabetical order, where first letter is lowercase, and second letter is uppercase. An example is shown below. Your solution must be in folder **task3/main.py**.

[('a', 'B'), ('c', 'D'), ('e', 'F'), ('g', 'H'), ('i', 'J'), ('k', 'L'), ('m', 'N'), ('o', 'P'), ('q', 'R'), ('s', 'T'), ('u', 'V'), ('w', 'X'), ('y', 'Z')]

4. (4/0 points) Write a script to do the following:

1. (0.5 points) Keep asking the user to enter the positive integer number, *n*, greater than *0* as long as the entered number does not satisfy the previous constraint (assume the user is entering an integer).
2. (0.5 points) Generate a list, *l*, of *n* random integer elements from interval *[-10, 10]* (both values included).
3. (1 point) Print the number of unique values in the list *l*, aligned to the ***right***. Before that print ***centered*** title "Unique elements".
4. (1 point) Print the number of occurrences of each element of the list. On the ***left side*** print the element, on the ***right side*** print the frequency. Before that print ***centered*** title "Statistics".
5. (1 point) Print ***centered*** the most frequent elements in the list, element, and frequency in parentheses. Before that print ***centered*** title "Most frequent elements".

Screen width must be 80 characters! Your solution must be in folder **task4/main.py**.

```
Enter n:-6
n must be positive
Enter n:100
--------------------------------Unique elements---------------------------------
                                                                              20
---------------------------------Statistics-----------------------------------
-10                                                                            7
 -9                                                                            8
 -8                                                                            3
 -7                                                                            3
 -6                                                                            5
 -5                                                                            5
 -4                                                                            6
 -3                                                                            1
 -2                                                                            3
  0                                                                            7
  1                                                                            5
  2                                                                            8
  3                                                                            6
  4                                                                            6
  5                                                                            7
  6                                                                            3
  7                                                                            2
  8                                                                            1
  9                                                                            6
 10                                                                            8
-----------------------------Most frequent elements-----------------------------
                                    -9(8)
                                     2(8)
                                    10(8)
```

Learning outcome 2

5. (1/0 point) What means -> in function annotations?

6. (3/0 points) Write a function *calculate*, having two arguments: list of operands and list of operators. Operands are arbitrary real numbers and operators are + (addition), - (subtraction), * (multiplication) and / (division). Function must return a result of **serial application** (no operator precedence) operators to operands, or None in case of division by zero. Your solution must be in folder **task6/main.py**.

```
#prints the result of the calculation
# 1 + 2 - 3 * 4 + 5 = -5
print(calculate([1, 2, 5, 4, 3], ['+', '-', '*', '+']))
#prints the result of the calculation
# 1 + 2 - 3 * 4 + 5 = 0
print(calculate([1, 2, 3, 4, 5], ['+', '-', '*', '/']))
```

7. (3/0 points) Write a generator function *my_letters*, having two string arguments, *start* and *end*, and one integer argument, *step*. The *step* must have a default value of one. *Start* and *end* are UTF-8 characters, and *step* is a difference between each character in the sequence. Generator function generates *tuple* having a first element enumeration and second element a character.

Demonstrate using of function *my_letters* in *for* loop, having a function on a place of iterable. Print generated elements **in the same line**. Your solution must be in folder **task7/main.py**.

```
#(1, 'a') (2, 'f') (3, 'k') (4, 'p') (5, 'u') (6, 'z')
for ... in my_letters("a", "z", 5):
    ...
```

8. (0/3 points) Write a recursive *fibonacci* function returning n-*th* Fibonacci number. Fibonacci number, $F_n = F_{n-1} + F_{n-2}$, is defined as a sum of previous two numbers. $F_1 = 0$ and $F_1 = 1$. Use function annotations for function argument and a function return value. Write a Docstring for implemented function. Your solution must be in folder **task8/main.py**.

Learning outcome 3

9. (1/1 points) What is a package and what is a first module executing when importing package (if existing)?

10. (2/1 points) Create a module *lib.py*. In the *lib.py* create a **hidden** reference *h* with value of *10* and the *adder* function as follows:

```
def adder(a, b):
    return a + b
```

Print return value of adder function in *lib.py* passing any numeric value and hidden variable h. Import *lib.py* in *main.py* using ***from-**** notation. Print return value of adder function from *lib.py* **having on the screen only** sum of given values *5* and *10*. Your solution must be in folder **task10/main.py** and **task10/lib.py**.

Learning outcome 5

11. (1/0 point) What is benefit of using **with** block when working with files?

12. (5/0 points) Write a script to copy text contents of a file to another file. Assume both files exist and the contents in another file must not be corrupted (lost or modified). Your solution must be in folder **task12/main.py.**

13. (3/0 points) Write a script to read a random line from a text file. If file is empty, print an appropriate message. Your solution must be in folder **task13/main.py.**

14. (1/2 points) Generate a list of *20* random integers in range *[-100, 100]*. Serialize that list to *json* file, *numbers.txt*. Print to the screen the contents of the *numbers.txt* file (raw content, not deserialized content). Your solution must be in folder **task14/main.py.**