Programming for Data Science DS2002

Friday, 12th of January 2024

9:00 – 13:00

ITE, Halmstad University

Contact: Slawomir Nowaczyk

- The exam has **10 questions**, each corresponding to one of the topics covered in the course, and each is worth **15 points**. However, the questions are not necessarily of equal difficulty. Make sure you focus on the easiest ones first!
- Remember to mark each page with your anonymous code.
- Please answer each question in the designated area. If you run out of space, use extra paper sheets, but clearly write down that the answer is continued elsewhere, and on which page number.
- Mark MCQs with "X"; most of the time, only one answer is correct (barring mistakes, of course) – unless the question explicitly asks to mark all correct answers.
- Especially when writing code, use comments and explain your reasoning. It is easy to make small mistakes, but if we know what your intention was, we can better understand your actual knowledge and possibly give partial credit.
- The questions where you are asked to write code test two things: your algorithmic thinking, i.e., your ability to give step-by-step instructions to a computer, and your knowledge of key programming/Python concepts. Solutions do not have to be perfectly correct in terms of syntax, and they are not (of course!) expected to be well-tested.
- In order to **pass** this exam, you need to get at least a total of **75 points** (50%), as well as at **least 3 points** (20%) for **every** question! Pay attention to the time left!
- The final grade in the course is a weighted average of exam and lab results.
- Tools allowed: pen/pencil, eraser, English dictionary (as long as it is a paper one and without any notes).
- No calculators or other electronic devices are allowed!

I confirm that I have read and understood these rules. Student ID: | | | | – | | | |

Good Luck!

1. Key Programming Concepts

(a) Define the "variable scope" concept and explain the difference between local and global scopes in Python. Why are these different scopes beneficial? Explain how Python's scope resolution for variables works when a variable is defined both inside a function and globally. Provide a short code example.

(b) Explain the concept of loops in programming – why are they important and useful? What are the main challenges when using loops?

(c) When importing a module in Python, what is the purpose of the `as` keyword?

- ☐ To rename the module for easier access in the code
- ☐ To import only a specific part of the module
- ☐ To check if the module exists and handle exceptions
- ☐ To import the module in a specific scope

(d) What is the difference between "mutable" and "immutable" objects? Give one example of a data type which is mutable in Python, and one example that is immutable. Do they necessarily need to be like this, or could they be swapped? Why or why not?

(e) In Python, a function that accepts no arguments is defined using the keyword _____, followed by the _____, then _____ and _____, and finally function _____.

2. Variables

(a) In Python, you encounter variables of different types, such as integers, strings, and floating-point numbers. Describe the differences between these three types, and give two more examples of possible types. What happens when a variable of one type is assigned a value of another type? Explain (with code examples) when Python performs type conversion, and how is it done.

(b) Discuss the rules and conventions for naming variables in Python. What are the restrictions on variable names? Provide examples of valid and invalid variable names and explain why. Also, discuss the importance of following naming conventions in code readability and maintainability.

(c) Given "`a, b, c = 5, "Hello", 3.14`", what are the types of the three variables?

- ☐ float, int, str
- ☐ int, int, str
- ☐ str, float, int
- ☐ int, str, float

(d) Given the code below:

```
x = "first"

def outer_function():
    x = "second"

    def inner_function():
        nonlocal x
        x = "third"
        return x

    inner_result = inner_function()
    return inner_result

def change_global():
    global x
    x = "fourth"

print(x)
outer_result = outer_function()
print(outer_result)
change_global()
print(x)
```

What is the value of `x` after the `outer_function` is called but before `change_global` is called?

What does the `nonlocal` keyword in `inner_function` indicate about the variable `x`?

What is printed on the console as the final output of the script?

(e) What is the default value of a variable declared but not initialised in Python?

- ☐ None
- ☐ 0
- ☐ "" (empty string)
- ☐ Python does not allow declaration without initialisation

(f) What happens if you try to access an unassigned variable in Python?

- ☐ Returns None
- ☐ Returns 0
- ☐ An error occurs
- ☐ Creates an empty variable

3. Conditionals

(a) In Python programming, an `if` statement can execute a block of code only if the given condition evaluates to _____. Describe the syntax for `if` statement.

(b) Which of the following are valid conditions (syntactically correct and meaningful) in Python? Mark all correct answers. Assume these variables are defined: `x` is a number, `mystr` is a string, `isempty` is a boolean.

- ☐ `x > 5`
- ☐ `len(mystr) == 5`
- ☐ `"Python"> mystr`
- ☐ `x = 5`
- ☐ `x >= 18 or <= 65`
- ☐ `(x >= 18) and (x <= 65)`
- ☐ `not isempty`
- ☐ `len(mystr) > -1`
- ☐ `"Hello"- "H"`
- ☐ `(x > 10) and (x < 5)`

(c) Complete the following Python code to print "Teenager" if the age is between 13 and 19, "Adult" if the age is 20 or more, and "Child" otherwise:

```
age = 18
if age >= 20:

    print("_____")

______ age >= 13:
    print("Teenager")

______:
    print("Child")
```

(d) In Python, what does the expression `(False or True) and (not False)` evaluate to?

- ☐ True
- ☐ False
- ☐ It is not an expression
- ☐ It raises an exception

(e) What value will be printed by the following Python code?

```
x = 10
if x > 5:
    if x > 15:
        print("A")
    else:
        print("B")
else:
    print("C")
```

(f) Explain what happens in terms of short-circuit evaluation when the following Python code is executed: `a = False and (b / 0)`. Specifically, discuss whether an error occurs and why. How could one modify the code to flip your answer concerning the error?

(g) Discuss the importance of indentation in Python. Specifically in `if` statements, how does the indentation change the results of code? Give an example. Moreover, incorrect indentation in conditional statements can lead to `IndentationError`. Give an example of such code.

4. Loops

(a) Fill in the blanks to complete the Python code that prints each number:

```
numbers = [1, 2, 3, 4, 5]

for num in ________:

    print(______)
```

(b) Describe how a `while` loop works in Python and give an example where a `while` loop would be more appropriate than a `for` loop.

(c) What will the following Python code print?

```
for i in range(2):
    for j in range(4):
        print(i, j)
```

(d) The `break` statement in Python is used to:

- ☐ Continue executing the next iteration of the loop without completing the current one.
- ☐ Repeat the current iteration of the loop from the beginning.
- ☐ Exit the current loop and resume execution at the next statement outside of the loop.
- ☐ Increase the loop counter by a specified increment.

(e) Complete the following Python code to skip printing the number 3:

```
for i in range(5):
    if i == 3:

        --------
    print(i)
```

(f) Explain the role of a loop control variable in a Python loop, and provide an example of how it's used in a `while` loop and in a `for` loop.

(g) What is the output of the following code?

```
for i in range(1, 5):
    if i == 3:
        continue
    for j in range(1, 4):
        if j == 2:
            break
        for k in range(1, 4):
            if k == 3:
                continue
            print(f"i={i}, j={j}, k={k}")
```

(h) Which Python method is used to make an object iterable?

- ☐ `__iter__()`
- ☐ `__next__()`
- ☐ `__loop__()`
- ☐ `__yield__()`

5. Functions

(a) Fill in the blanks to correctly define a Python function:

```
_____ greet_____:
    print("Hello, World!")
```

(b) A function `add_numbers` is defined as `def add_numbers(a, b): return a + b`
Write down two different correct calls to this function, and the corresponding results.
Write down three different *incorrect* calls to this function, and describe the corresponding error messages (you don't need to know the exact wording of the errors, of course, but make sure the meaning is clear, and that these are three *different* errors).

(c) Explain the meaning of `return` keyword in Python. Why is it important? What happens if a function does not contain a `return` statement?

(d) What will be printed when the following code is executed?

```
x = 10
def test():
    x = 20
test()
print(x)
```

- ☐ Error
- ☐ 10
- ☐ 20
- ☐ None

(e) Describe how arguments work in Python functions. Explain the default parameter values, and provide two examples of a function with a default parameter(s). Show examples of correct and incorrect calls. Explain the difference between keyword and positional arguments in Python functions, with examples.

6. Python Language

(a) What is the output of the following Python code?

```
def modify_list(lst):
    lst.append(4)
    lst = lst + [5]
    lst[0] = 'X'
    return lst

def another_function(lst):
    lst.append(6)
    lst = [7, 8, 9]
    return lst

original_list = [1, 2, 3]
modified_list = modify_list(original_list)
another_list = another_function(original_list)

print("Original List:", original_list)
print("Modified List:", modified_list)
print("Another List:", another_list)
```

(b) Briefly explain the idea of exceptions and exception handling in Python, and describe `try`, `except`, and `finally` blocks.

(c) When assigning `var1 = var2` in Python, where `var2` is a list, what happens?

- ☐ `var1` gets a copy of `var2`
- ☐ `var1` references the same list as `var2`
- ☐ An error occurs
- ☐ `var1` becomes None

(d) Fill in the blanks in the Python code below to create a list comprehension that generates a list of squares for all even numbers between 1 and 10 (inclusive).

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

squares_of_even = [__________ for num in ________ if ________]
print(squares_of_even)
```

(e) Suppose you have a Python file named `math_utils.py` that contains several utility functions for mathematical operations. One of the functions in this file is `add`. In another Python file in the same directory, you want to use the add function from the `math_utils` module. Fill in the blanks in the code below to correctly import and use the function:

```
------------------------------
result = add(10, 20)
print(result)
```

(f) What does it mean for Python to be a dynamically typed language? How does it differ from statically typed languages? Discuss the advantages and disadvantages of dynamic typing in Python.

(g) Discuss the usefulness of functions in Python programming. In your response, address the following aspects: Code Reuse, Modularity, Readability, Maintainability, Abstraction, Testing/Debugging.

7. Data Structures

(a) Given the code below:

```
a = np.array([[1], [2], [3]])
b = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
c = a * b
```

What are the shapes of the arrays `a`, `b`, and `c`? Compute the value of the array `c`.

In which array and along which axis was the broadcasting done? Write down how the array looks after broadcasting. Motivate your answer.

(b) Assume you have two variables, each defined as a list of equal-length lists representing a matrix.

Write a function `get_shape` that takes a matrix as above and returns its shape.

Write a function `matmul` that takes two matrices (as defined above) as arguments and returns the matrix multiplication result (if possible – make sure to check that the shapes of the matrices allow for multiplication... if not, return a message "These matrices cannot be multiplied").

(c) Which of the following data structures in Python are mutable? Mark all the correct answers.

- ☐ int
- ☐ tuple
- ☐ list
- ☐ set
- ☐ string
- ☐ dictionary
- ☐ class
- ☐ Pandas DataFrame
- ☐ Pandas Series
- ☐ NumPy array

(d) Fill in the missing parts in the code, choosing where to use `list` and where to use `dict`.

```
# Define a collection to store the names of students in a class

students = ____________

# Add student names

students.____________("Alice")

students.____________("Bob")

students.____________("Charlie")

# Define a collection to store student grades

grades = ____________

# Add grades

grades____________ = 5

grades____________ = 4

grades____________ = 3

# Function to print student names
def print_student_names(student_collection):

    for student ____________ student_collection:
        print(student)

# Function to print student grades
def print_student_grades(grade_collection):

    for ________________________ grade_collection____________:
        print(f"{student}: {grade}")
```

8. Object-Oriented Programming

(a) A rectangle is defined by two points (x_1, y_1) and (x_2, y_2), representing the top-left and bottom-right corners, respectively. Define a suitable class for such rectangles. Add a method to the class that computes and returns the area of the rectangle.

(b) When will the `__init__()` method be called?

- ☐ When the class is defined.
- ☐ When an instance of the class is created.
- ☐ It is used to initialise the class but is never called.
- ☐ It is used to initialise the class and is called before the `super()` method is called.

(c) Define a class `Employee` that represents a person working in a company, and provides suitable operations as in the following example:

```
a = Employee(firstname="Ada", lastname="Lovelace", pos="Programmer")
b = Employee(firstname="John", lastname="Grisham", pos="Accountant")
a.set_salary(100)            # Set Ada's salary
b.give_raise(percent=10)     # Increase her salary by 10%
```

(d) What is the primary purpose of a class in Python?

- ☐ To create specific data types.
- ☐ To store global variables.
- ☐ To serve as a blueprint for creating objects.
- ☐ To execute a program.

(e) Define a second class, `Company`, that represents a company that employs employees as defined above, and provides suitable operations as in the following example:

```
c = Company(name="Best␣of␣AI", address="The␣Moon")
c.hire(a, 200)              # Hire Ada with salary of 200
c.hire(b, 150)              # Hire John with salary of 150
# Calculate monthly salary cost (sum the salaries of all employees)
c.total_cost()
c.nr_positions("Programmer")   # how many programmers are employed?
c.
```

(f) In Python, what distinguishes a class variable from an instance variable?

- ☐ Class ones are defined within a method, while instance ones are defined outside of a class.
- ☐ Class ones are shared across all instances of the class, while instance variables are unique to each instance.
- ☐ Class variables are immutable, while instance variables are mutable.
- ☐ Class ones can only store numeric data, while instance ones can store any type of data.

9. Recursion

(a) What is recursion in programming?

- ☐ A method where a function calls itself
- ☐ A loop that executes indefinitely
- ☐ A technique to iterate over collections
- ☐ A way to declare multiple functions at once

(b) Create a recursive function that takes a positive integer as input and prints the numbers from that integer down to `1`. For example, if the input is `4`, the function should print: `4, 3, 2, 1`.

(c) Write a recursive function to print the names of all sub-folders and files within a given folder. The folder structure is represented as a tree, where each folder node contains a list of its sub-folders and files. Your algorithm should traverse this tree recursively and print the names of all encountered sub-folders and files. You can use the `isdir(folder_path)` function to check if a child is a folder or a file, and `listdir(folder_path)` to get a list containing the names of the files and sub-folders in the folder given by path

(d) When implementing a recursive Depth-First Search (DFS) algorithm on a binary tree, what does the recursive call represent?

- ☐ The root of the tree.
- ☐ The current node being processed.
- ☐ Handling the children of the current node.
- ☐ The leaf node (i.e., a node without children).

(e) Which of the following problems can be solved using recursion? Mark all the correct answers.

- ☐ Adding two numbers.
- ☐ Swapping two variables.
- ☐ Calculating the absolute value of a number
- ☐ Calculating factorial of a number.
- ☐ Converting Celsius to Fahrenheit.
- ☐ Concatenating two strings.
- ☐ Sorting.

(f) Explain the importance of "base case" in recursive functions. What does it mean? How is it different from a "typical" case? What happens if the base case is missing in a function?

10. Libraries

(a) In `Pandas` library, explain the difference between `.loc` and `.iloc` slicing methods. Motivate your answer with appropriate examples.

(b) Choose one Python *built-in* library/module, and describe its overall purpose in one sentence. Give at least three examples of code using this library, and describe what each of them does.

(c) How do you access the third element in the second row of a 2D NumPy array `arr`?

- ☐ arr[2, 1]
- ☐ arr[2][3]
- ☐ arr[1, 2]
- ☐ arr[2, 3]

(d) How can you generate a random floating-point number between 5 and 10?

- ☐ `random.randint(5, 10)`
- ☐ `random.random() * 5 + 5`
- ☐ `random.choice(5) * 10`
- ☐ `random.seed(10) + 5`

(e) What attribute of a NumPy array is used to determine its shape (dimensions)?

- ☐ `shape`
- ☐ `dim`
- ☐ `size`
- ☐ `type`

(f) Fill in the gaps in the following code:

```
import altair as alt
from vega_datasets import data
cars = data.cars()
cc = alt.value('lightgray')

interval = alt._____________


base = alt.Chart(cars)._________________.encode(
    y='Horsepower',

    color=alt.condition(_____________, 'Origin', cc),
    tooltip='Name'
).add_selection(
    interval
)


hist = alt.Chart(_____________).mark_bar().encode(

    x='_____________',
    y='Origin',
    color='Origin'
).properties(
    width=800, height=80
).transform_filter(interval)

sc = base.encode(_____________='Miles_per_Gallon') \

   | base.encode(_____________='Acceleration')
sc & hist
```

(g) Explain briefly the difference(s) between `scikit-learn` and `PyCaret`.

THE END