UNIVERSITY OF GOTHENBURG

DIT009 - Fundamentals of Programming

H24 Exam 1 - Lindholmen, 2024-10-30

Teachers:	Francisco Gomes and Ranim Khojah
Questions:	+46 31 772 6951 (Francisco) Francisco will visit the exam hall at ca 15:00 and ca 16:30
Results:	Will be posted within 15 working days.
Grades:	00–49 points: U (Fail) 50–69 points: 3 (Pass) 70–84 points: 4 (Pass with merit) 85–100 points: 5 (Pass with distinction)
Allowed aids:	No aids (books or calculators) are allowed.

Read the instructions below. Not following these instructions will result in the point deductions.

- Write clearly and in legible English (illegible translates to "no points"!). Motivate your answers, and clearly state any assumptions made. Your own contribution is required.
- Before handing in your exam, **number and sort the sheets in task order!** Write your anonymous code and page number on every page! The exam is anonymous, **do not** leave any information that would reveal your name on your exam sheets.
- Answers that require code must be written using the Python programming language. When answering what is being printed by the program, make sure to write as if the corresponding answer were being printed in the console.
- You can assume that all python files in the code presented in this exam are in the **same folder**. Similarly, **you can assume** that all libraries used in the code (math, regex, files, json, etc.) are properly imported in their respective file.

Cheatography

Regular Expressions Cheat Sheet

by Dave Child (DaveChild) via cheatography.com/1/cs/5/

Anchors

^	Start of string, or start of line in multi-line pattern
\A	Start of string
$	End of string, or end of line in multi-line pattern
\Z	End of string
\b	Word boundary
\B	Not word boundary
\<	Start of word
\>	End of word

Character Classes

\c	Control character
\s	White space
\S	Not white space
\d	Digit
\D	Not digit
\w	Word
\W	Not word
\x	Hexadecimal digit
\O	Octal digit

POSIX

[:upper:]	Upper case letters
[:lower:]	Lower case letters
[:alpha:]	All letters
[:alnum:]	Digits and letters
[:digit:]	Digits
[:xdigit:]	Hexadecimal digits
[:punct:]	Punctuation
[:blank:]	Space and tab
[:space:]	Blank characters
[:cntrl:]	Control characters
[:graph:]	Printed characters
[:print:]	Printed characters and spaces
[:word:]	Digits, letters and underscore

Assertions

?=	Lookahead assertion
?!	Negative lookahead
?<=	Lookbehind assertion
?!= or ?<!	Negative lookbehind
?>	Once-only Subexpression
?()	Condition [if then]
?()|	Condition [if then else]
?#	Comment

Quantifiers

*	0 or more	{3}	Exactly 3
+	1 or more	{3,}	3 or more
?	0 or 1	{3,5}	3, 4 or 5

Add a ? to a quantifier to make it ungreedy.

Escape Sequences

\	Escape following character
\Q	Begin literal sequence
\E	End literal sequence

"Escaping" is a way of treating characters which have a special meaning in regular expressions literally, rather than as special characters.

Common Metacharacters

^	[	.	$
{	*	(	\
+	)	|	?
<	>		

The escape character is usually \

Special Characters

\n	New line
\r	Carriage return
\t	Tab
\v	Vertical tab
\f	Form feed
\xxx	Octal character xxx
\xhh	Hex character hh

Groups and Ranges

.	Any character except new line (\n)
(a|b)	a or b
(...)	Group
(?:...)	Passive (non-capturing) group
[abc]	Range (a or b or c)
[^abc]	Not (a or b or c)
[a-q]	Lower case letter from a to q
[A-Q]	Upper case letter from A to Q
[0-7]	Digit from 0 to 7
\x	Group/subpattern number "x"

Ranges are inclusive.

Pattern Modifiers

g	Global match
i *	Case-insensitive
m *	Multiple lines
s *	Treat string as single line
x *	Allow comments and whitespace in pattern
e *	Evaluate replacement
U *	Ungreedy pattern

* PCRE modifier

String Replacement

$n	nth non-passive group
$2	"xyz" in /^(abc(xyz))$/
$1	"xyz" in /^(?:abc)(xyz)$/
$`	Before matched string
$'	After matched string
$+	Last matched string
$&	Entire matched string

Some regex implementations use \ instead of $.

By **Dave Child** (DaveChild)
cheatography.com/davechild/
aloneonahill.com

Published 19th October, 2011.
Last updated 12th March, 2020.

Table 1: List of useful methods in Python, for Lists, Stringsm and Dictionaries.

Used in	Method specification	Description
—	int(value)	Converts the specified value into integer.
—	str(value)	Converts the specified value into a string.
—	float(value)	Converts the specified value into a float.
List	len(list)	Returns the number of elements in a list.
List	append(element)	Appends the specified element to the end of the list.
List	pop()	Removes and returns the element at the end of the list.
List	remove(element)	Remove the specified element from the lsit.
Dict	keys()	Returns a list with all the keys in a dictionary.
Dict	values()	Returns a list with all the values in a dictionary.
Dict	items()	Returns a list containing a tuple for each key value pair.
String	split(delimiter)	Split a string into a list of strings separated by the delimiter.
String	strip()	Remove empty spaces at the beginning and end of a string.
String	replace(pattern, replacement)	Returns a string where the pattern is replaced by the specified value.
String	lower()/upper()	Returns a new string with only lower or upper case characters.

Question 1 [Total: 15 pts]: This question verifies your knowledge of algorithms, variables, execution flow, and debugging programs. Write **exactly** what is printed when running the Python program below. **Important:** The **ordering and the formatting is important**.

```
counter = 10

def function_a(names, personA, personB):
    names.append(personA)
    result = "Result: "
    for i in names:
        result += i + " "

    names[-1] = personB
    print(result)

def function_b(x, y):
    z = 3

    if x % 2 == 0:
        z = x
        x = y
        y = z
    if x < y:
        x += y
    else:
        y += x
```

```
23
24      function_c(z)
25
26      print("A: x =", x)
27      print("A: y =", y)
28      print("A: z =", z)
29
30  def function_c(a):
31      counter = a + 2
32      total   = 0
33      for i in range(1,counter):
34          total = total + i + counter
35
36      print("B: total =", total)
37      print("B: counter =", counter)
38
39  def main():
40      names = ["Alan", "Margareth", "Ada"]
41      function_a(names, "Barbara", "Grace")
42
43      x = len(names)
44      y = len(names[1])
45      function_b(x, y)
46
47      print("Main: names =", names)
48      print("Main: counter =", counter)
49
50      function_a(names, "Peter", "Lynn")
51      print("Main: names =", names)
52
53      x = 5
54      y = 2
55      function_b(x, y)
56
57  if __name__ == "__main__":
58      main()
```

Question 2 [20 pts]: Using your knowledge of regular expressions, answer the questions below:

```
1  # A list of 10 social media posts
2  posts = [
3   "Just hit the gym! #fitnesslife - by user: fit_joe2023",
4   "Exploring the world one place at a time #wanderlust - by _travel_bella",
5   "Latest book read: 'Python Essentials' #reading - by user: bookworm123",
6   "Coding marathon tonight with @dev_guru and @code_mike2020 #hackathon",
7   "Can t believe it s 2024 already! #timeflies - by oldtimer67",
8   "New recipe dropped! Check out my food blog. #yummy - by foodie_jane",
9   "Music is life! #popdiva - shared by fan: soundwave_1987",
10  "Great game last night with @game_master. What a match! #sports",
11  "Loving the new tech gadgets this year. #innovation - by user tech_fanatic",
12  "Who else joined this platform in 2022? #newbies - by freshface22"
13 ]
14
```

```python
# REGEX used in the question
# a) r"\d{4}"
# b) r"[_@]\w+?\b"
# c) r"\bf.*?\b"
# d) r"[A-Z]\w+\b"
# e) r"\b\w+$"
```

Q2.1 [10 pts]: Write the result of doing `re.findall(expression, post)` in each regex marked in the code on each post in the list.

Important: 1) Indicate which regex you are answering to by using a, b, c, d, or e. For regex that return more than 3 matches, you may write only three matches in any order you prefer (note that writing more than three and including incorrect ones can risk deducting points).

Q2.2 [10 pts]: Write two regex to: 1) Extract all the sentences that end with exclamation point (!), and 2) Extract all hashtags in the post (the hashtags are words that begin with the symbol #). For this problem you can assume that all hashtags are unique in a post. You **must** also explain each of your regexes.

Question 3 [Total: 45 pts]: This question verifies your knowledge of code quality, refactoring, data structures, algorithms, and debugging programs. You were hired to maintain code that stores bank transactions, and received the code below.

```python
def function_a(transactions):
    for id in transactions.keys():
        transaction = transactions[id]

        matches = re.findall(r"\w+\d{2}\b", id)
        if len(matches) > 0 :
            print(id, transaction)

def function_b(transaction):
    bonus = 0
    limit = 1000
    if transaction[1] < limit :
        difference = limit - transaction[1]
        if difference >= 100:
            bonus = 100
        else:
            bonus = difference

    transaction[1] = transaction[1] + bonus

def main():
    print("Start of the Program: ")
    transactions = { "A0B26" : ("Alan" , 5000, "Ana"  ),
                     "V11B3" : ("Alice", 1000, "Maria"),
                     "SBS02" : ("Ada"  ,  200, "Alex" ),
                     "21234" : ("Grace",  700, "John" ),
                     "1BSAA" : ("Ada"  ,  950, "Alex" )}
    function_a(transactions)
```

```
29
30      for id in transactions.keys():
31          transaction = transactions[id]
32          function_b(transaction)
33
34      print("All Transactions: ")
35      for id in transactions:
36          transaction = transactions[id]
37          print(f"Value: {transaction[1]} SEK. From: {transaction[0]}. To: {
    transaction[2]}")
38
39  if __name__ == "__main__":
40      main()
```

Q3.1 [15 pts]: Lists and dictionaries are two different abstract data types in Python. Explain: i) the difference between lists and dicitionaries, and, **for each**, ii) one advantage and disadvantage of using them.

Q3.2 [8 pts]: The code above has one bug. You must: i) and explain what caused the bug, ii) describe how we can fix the bug.

Q3.3 [7 pts]: Assuming that the bug was fixed and the code works as intended, print the output shown by executing the code.

Q3.4 [15 pts]: Regarding `function_a` and `function_b`, using natural language, explain i) what is the purpose of each function and ii) how do they work. You must also iii) propose new names for each function. Your answer will be evaluated based on correctness, clarity, and correct usage of programming terminology.

Question 4 [Total: 20 pts]:: Using your knowledge of programming, algorithms, and recursion, answer the questions below:

Q4.1 [10 pts]: Write a function that takes a sorted list of integers as input and returns a string. For each element in the list (excluding the last element), find all other numbers in the list that are divisible by it. The resulting string should then include each element listing these divisible numbers.

Important: You can assume that the list is always sorted in an ascending order. In case there are less than two numbers in the list, your function should return the message: "Invalid input. We require at least 2 numbers."

```
Input: [1, 2, 3, 4, 6, 12]
Output:
1: 2 3 4 6 12
2: 4 6 12
3: 6 12
4: 12
6: 12
```

```
Input: [2, 4, 5, 8, 10, 20]
Output:
2: 4 8 10 20
4: 8 20
5: 10 20
8:
10: 20
```

```
Input: []
Output:
"Invalid input. We require at least
    2 numbers."
```

```
Input: [2, 5, 7, 11, 13]
Output:
2:
5:
7:
11:
```

Q4.2 [10 pts]: Write a function that takes a list of strings and a desired string. The function should go through the list and return the index of the desired string in the list. In case the string is not included in the list, the program should return -1. **Note: You must use recursion in your solution.**
Important: You can assume that the list contains only strings. The elements in the list can be in any order (i.e., the list is not sorted). In case there are multiple occurrences of the string in the list, the function should return the first occurrence (i.e., closer to the beginning of the list).

```
Input: ["apple", "banana", "cherry", "apple", "date"], "apple"
Output: 0

Input: ["apple", "banana", "cherry", "apple", "date"], "melon"
Output: -1

Input: [], "tomato"
Output: -1

Input: ["apple", "banana"], "banana"
Output: 1
```