Re-exam in Advanced Programming in Python (DAT515)

Chalmers University of Technology
25 August 2022, 8:30 to 12:30, on campus Johanneberg
Examiner: Aarne Ranta aarne@chalmers.se
Tel. 1082, mobile 0729 74 47 80

Write your answers directly below the questions. You can of course use separate sheets of paper to draft and experiment, but only the question papers will be graded. This reflects the fact that the answers can and should be short.

You will need 15 points out of 30 in questions 1-6 of this exam to get accepted with the grade that your lab work allows.

If you have done extra labs (colouring or clustering) you will also need to get half of the points for the corresponding extra questions. If you have not done extra labs, your answers to those questions will not be graded.

The final result of this exam will be reported as 3, 4, 5, or rejected. As specified in the course plan, you will get

- grade 5 if you have at least 50 points from the labs, at least half of the points of the bonus questions corresponding to your labs, and at least 15 points from questions 1-6.
- grade 4 if you have at least 40 points from the labs, at least half of the points of the bonus questions corresponding to your labs, and at least 15 points from questions 1-6.
- grade 3 if you have at least 30 points from the labs and at least 15 points from questions 1-6.
- grade U otherwise

For your reference: the syntax of (the relevant parts of) Python

```
<stm> ::= <decorator>* class <name> (<name>,*)?: <block>
      |   <decorator>* def <name> (<arg>,*): <block>
      |   import <name> <asname>?
      |   from <name> import <imports>
      |   <exp>,* = <exp>,*
      |   <exp> <assignop> <exp>
      |   for <name> in <exp>: <block>
      |   <exp>
      |   return <exp>,*
      |   yield <exp>,*
      |   if <exp>: <block> <elses>?
      |   while <exp>: <block>
      |   pass
      |   break
      |   continue
      |   try: <block> <except>* <elses> <finally>?
      |   assert <exp> ,<exp>?
      |   raise <name>
      |   with <exp> as <name>: <block>
<decorator> ::= @ <exp>
<asname>    ::= as <name>
imports     ::= * | <name>,*
<elses>     ::= <elif>* else: <block>
<elif>      ::= elif exp: <block>
<except>    ::= except <name>: <block>
<finally>   ::= finally: <block>
<block>     ::= <stm> <stm>*
<exp> ::= <exp> <op> <exp>
      |   <name>.?<name>(<arg>,*)
      |   <literal>
      |   <name>
      |   ( <exp>,* )
      |   [ <exp>,* ]
      |   { <exp>,* }
      |   <exp>[exp]
      |   <exp>[<slice>,*]
      |   lambda <name>*: <exp>
      |   { <keyvalue>,* }
      |   ( <exp> for <name> in <exp> <cond>? )
      |   [ <exp> for <name> in <exp> <cond>? ]
      |   { <exp> for <name> in <exp> <cond>? }
      |   { <exp>: <exp> for <name> in <exp> <cond>? }
      |   - <exp>
      |   not <exp>
<keyvalue> ::= <exp>: <exp>
<arg>      ::= <name>
           |   <name> = <exp>
           |   *<name>
           |   **<name>
<cond> ::= if <exp>
<op>   ::= + | - | * | ** | / | // | % | @
       |   == | > | >= | < | <= | != | in | not in | and | or
<assignop> ::= += | -= | *=
<slice> ::= <exp>? :<exp>? <step>?
<step>  ::= :<exp>?
```

Question 1 (12 p). What is the *value* (or possibly error) of the following expressions? Remember that **None** is also a value!

- `'python'.reverse()`
  Answer:

- `'python'[-1::-2]`
  Answer:

- `[str(x) for x in range(3)]`
  Answer:

- `{c: list(c) for c in 'typhoon'}`
  Answer:

- `{'x' for x in range(120)}`
  Answer:

- `1 > 2 and x > 1`
  Answer:

- `[print(x) for x in range(3)]`
  Answer:

- `2 * False`
  Answer:

- `2 * 'False'`
  Answer:

- `lambda x: range(x)`
  Answer:

- `[x for x in range(3) if x < x**2 < x**3]`
  Answer:

- `{{}}`
  Answer:

Question 2 (3 p). Write a lambda expression for a function that sorts the characters in a string in their usual alphabetical order, so that for instance 'bAdC' is converted to 'AbCd'. Both the argument and the return value are strings, and they must contain the same characters, although possibly in a different order, not forgetting duplicates.

Question 3 (6 p). Consider a dictionary of the following form (assumed to contain all countries of the world):

```
countries = {
 'Afghanistan': {'capital': 'Kabul', 'area': 652230, 'population': 36643815,
'continent': 'Asia', 'currency': 'afghani'},
 'Albania': {'capital': 'Tirana', 'area': 28748, 'population': 3020209,
'continent': 'Europe', 'currency': 'lek'},
 'Algeria': {'capital': 'Algiers', 'area': 2381741, 'population': 41318142,
'continent': 'Africa', 'currency': 'dinar'},
# etc
}
```

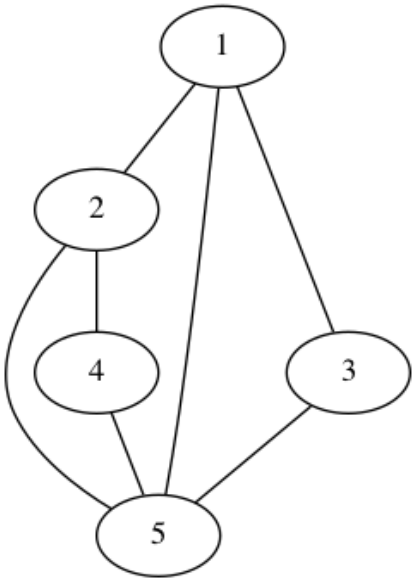
Write a Python expression that answers the query *How many countries in Europe have a population of over ten million?* by returning an integer.

Answer:

Also write an expression for a dictionary that, to each continent, assigns the total population of all countries on that continent.

Answer:

Question 5 (3 p). Write the adjacency list of the following graph as a Python dictionary.



Question 6 (6 p). Class hierarchies can model objects that share some attributes but differ in others. In our example, we consider literary works, which always have a title and an author. Some works are books, which have a number of pages as well. Some works are recordings and have a duration. Finally, an audio book has all these attributes.

Write Python class definitions for the following classes and their attributes, named as follows:

- `Work: title, author`
- `Book: title, author, number_of_pages`
- `Recording: title, author, duration`
- `Audiobook: title, author, number_of_pages, duration`

Your class definitions should enable the following expressions for class instances:

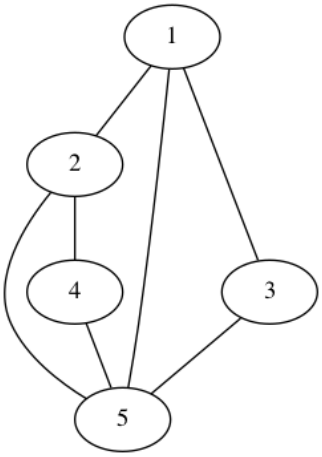
- `Work('Kniv', 'Jo Nesbø')`
- `Book('Kniv', 'Jo Nesbø', 758)`
- `Recording('Kniv', 'Jo Nesbø', 21.2)`
- `Audiobook('Kniv', 'Jo Nesbø', 758, 21.2)`

Use as much inheritance as possible. You don't need to define getters or setters: just make sure that all attributes appear as class variables that are set when the class constructors are called.

Bonus question on graph colouring (6 p). Answer this question if and only if you have submitted the extra lab on graph colouring (be it one or two parts - the question is the same in both cases).

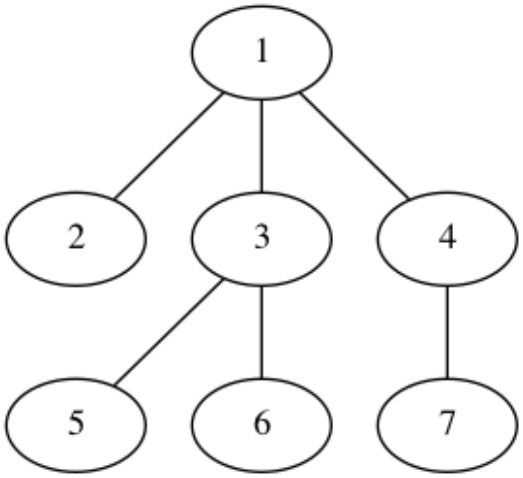
Use the simplify-select algorithm to colour the following graph with just three colours. Show

- an order in which the vertices can be removed in the simplify phase,
- how colours are then selected for each vertex at a time.



Bonus question on clustering (6 p). Answer this question if and only if you have submitted the extra lab on clustering.

Consider the graph



The weight of each edge is defined as the absolute value of the difference of the numbers of the connected vertices. For example, the weight of the edge (3, 5) is 2. Start by writing these weights next to each edge in the picture above. Then show (beside the graph) the k-spanning tree clusters with k = 2, 3, 4. It is enough to state which edges are removed in each case. The aim is to remove the heaviest edges first.