Re-exam in Advanced Programming in Python (DAT515)

Chalmers University of Technology
12 April 2022, 8:30 to 12:30, on campus Johanneberg
Examiner: Aarne Ranta aarne@chalmers.se
Tel. 1082, mobile 0763178590

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.

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 (9 p). What is the value (or possibly error) of the following expressions? Think about the result when you write the expression in the Python shell, and remember that **None** is also a value.

- `'python'.append('3')`
  Answer:

- `['p', 'y', 't', 'h', 'o', 'n'].append(3)`
  Answer:

- `['p', 'y', 't', 'h', 'o', 'n'] + [3]`
  Answer:

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

- `['x' for x in 'python']`
  Answer:

- `lambda x, y: x//y`
  Answer:

- `[(lambda x: 10/x)(x) for x in range(5)]`
  Answer:

- `'pdf2html'.split(2)`
  Answer:

- `2 > 3 or 'False'`
  Answer:

Question 2 (3 p). Write a function

```
alpha_sort(strings)
```

that sorts a list of strings in the usual alphabetical order, so that for instance 'anna' appears before 'Berit'. Both the argument and the return value are lists of strings, and they must contain the same strings although possibly in a different order, not forgetting duplicates. Full points are given to a solution where the function body is a single return statement.

Answer:

Question 3 (3 p). Write a `lambda` expression for a function that takes two integers **x** and **y** and a string **s**, and drops **x** characters from the beginning and **y** characters from the end of **s**. For example, **`f(2, 3, 'python programming')`** should return **`'thon programm'`** if your lambda expression is bound to the variable **f**.
Answer:

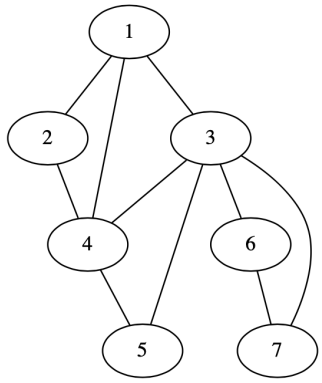
Question 4 (6 p). Consider a dictionary of the following form (containing all countries of the world):

```
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 different currencies are used in the countries of Europe?* by returning an integer.
Answer:

Also write an expression for a dictionary that, to each currency, assigns the list of those countries that use that currency.
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, all kinds of vehicles have a brand name, but cars have acceleration, and only for electric vehicles have range (how many kilometres the battery lasts). Electric cars have all these three attributes.

Write Python class definitions for the following four classes and their attributes:

- `Vehicle: brand`
- `Car: brand, acceleration`
- `ElectricVehicle: brand, range`
- `ElectricCar: brand, acceleration, range`

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

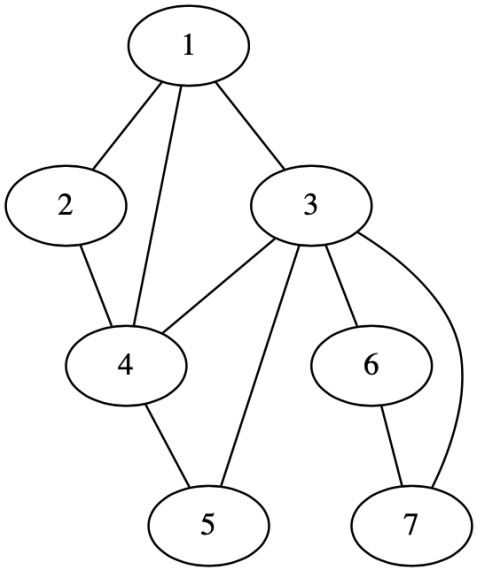
- `Vehicle('Airbus A350')`
- `Car('Saab 900', 10.5)`
- `ElectricVehicle('Brompton Electric Bike', 60)`
- `ElectricCar('Tesla Model 3', 5.3, 354)`

The definitions should use as much inheritance as possible, including **super()** when possible. You don't need to define getters or setters: just make sure that all the 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 (either one or two parts of the lab - the question is the same in both cases).

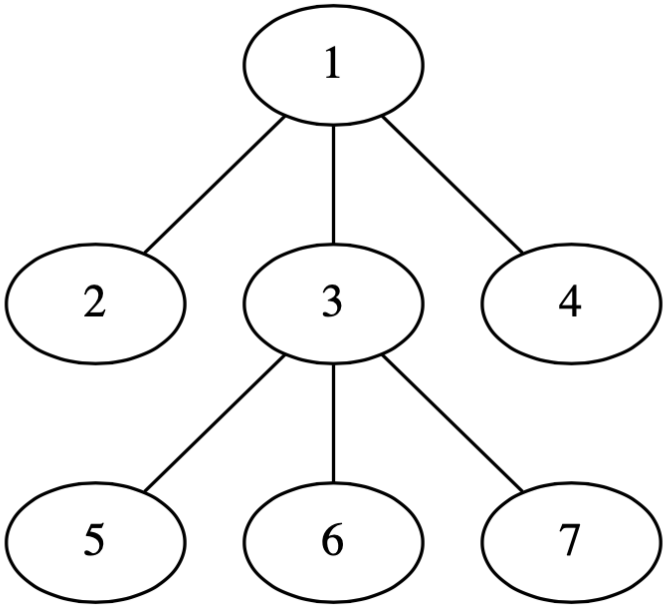
The problem is to colour the following graph with just three colours. Show

- an order in which the vertices can be removed in the simplify phase of the simplify-select algorithm,
- 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.

Then show the k-spanning tree clusters with k = 2, 3, 4. It is enough to show which edges are removed in each case. The aim is to remove the heaviest edges first.