Exam in Advanced Programming in Python (DAT515)

Chalmers University of Technology
10 January 2022, 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 1-8 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 1-8.
- 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 1-8.
- grade 3 if you have at least 30 points from the labs and at least 15 points from questions 1-8.
- grade U otherwise

If your lab grading has not been completed yet, your final grade will be determined when the lab grading is complete. Hence you don't need to worry about possibly missing lab points: they will be adjusted later. In particular, since extra lab grading may be delayed, you should do the bonus questions here even if you don't know the outcome of the grading yet.

For your reference: the syntax of (most 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?

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

- `sorted('python')`
  Answer:

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

- `{c for c in 'typhoon'}`
  Answer:

- `2 * 3 * 'python'`
  Answer:

- `4 * list(range(1, 4))`
  Answer:

- `'typhoon'[-2:]`
  Answer:

- `'from python to typhoon'.split()[3][1:-1]`
  Answer:

- `set('python') == set('typhoon')`
  Answer:

Question 2 (3 p). Write a function

```
tuples2dict(tuples)
```

that converts a list of tuples such as `[(1, 'one'), (2, 'two')]` to a dictionary that has the first elements of the tuples as keys and the second elements as values. The function body has to be a one-liner that just returns a dictionary comprehension.

Answer:

Question 3 (3 p). Write a `lambda` expression for a function that takes two strings (or lists) and returns the longest one of them (or just one of them if they are of equal length).

Answer:

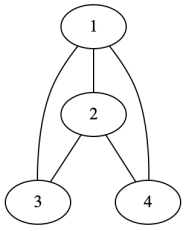
Question 4 (4 p). Consider a dictionary of the following form:

```
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 continents are there?* by returning an integer.
Answer:

Also write an expression for the query *List all continents with their total area in descending order.*
Answer:

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



Answer:

Question 6 (3 p). The following code defines a hierarchy of different kinds of publications. Complete the code by writing getter methods that return the values of the "hidden" class variables in each class, so that they need not be accessed from application code. **Do not repeat getter methods that are inherited from a superclass**. Write the getter method definitions in the empty space below each class definition.

```
class Publication:
    def __init__(self, title, author):
        self._title = title
        self._author = author
    def publication_type(self):
        return 'publication'
```

```
class Book(Publication):
    def __init__(self, title, author, publisher=None):
        super().__init__(title, author)
        self._publisher = publisher
    def publication_type(self):
        return 'book'
```

```
class Article(Publication):
    def __init__(self, title, author, journal=None):
        super().__init__(title, author)
        self._journal = journal
    def publication_type(self):
        return 'article'
```

Question 7 (2p). Which of the following class instance creations are acceptable in Python, given the class definitions in Question 6 and your getter methods with their natural names? Write the outcome in front of each statement, either "OK" or "wrong".

```
fight = Book('Why Men Fight')
fight = Book('Why Men Fight', '')
fight = Book('Why Men Fight', 'Russell')
fight = Book('Why Men Fight', 'Russell', publisher = 'The Century Co')
fight = Book('Why Men Fight', 'Russell', 'The Century Co')
```

Question 8 (3p). Assuming the last valid definition of `fight` in Question 7 and the following definitions,

```
object = Book('Word and Object', 'Quine', 'MIT Press')
denoting = Article('On Denoting', 'Russell', 'Mind')
euro50 = Publication('50 euro banknote', 'European Central Bank')
```

show the results of the following statements (which can be an error message):

```
print(fight.publication_type())
```

```
print(denoting.get_publisher())
```

```
print(fight.get_author() == denoting.get_author())
```

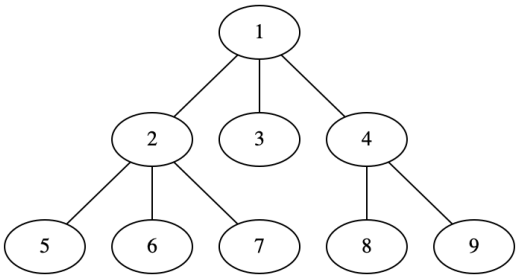
```
my_collection = [fight, object, denoting, euro50]
```

```
print([p.get_title() for p in my_collection if p.get_author() == 'Russell'])
```

```
print([p.get_title() for p in my_collection if p.publication_type() == 'other'])
```

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 two colours:

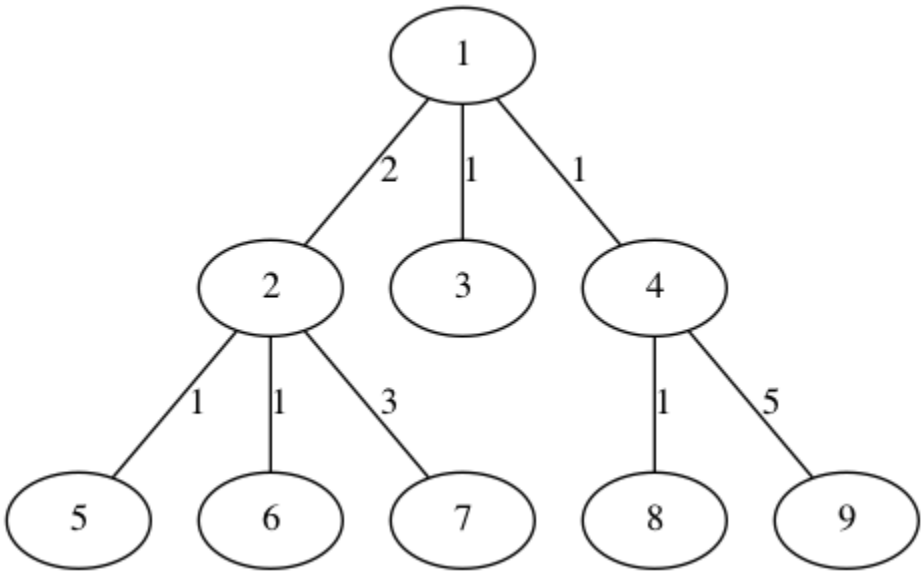


Show an order in which the vertices can be removed in the simplify phase of the simplify-select algorithm. (There are many possible orders, but it is enough to show just one of them.)

The graph above is actually a tree. Can all trees be coloured with just two colours so that adjacent vertices never get the same colour? Think about the problem and give an argument if you think it is possible (or even a proof if you can), or give a counterexample.

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

Consider the following graph with weighted edges:



Show the k-spanning tree clusters with k = 3, 4, 5. It is enough to show which edges are removed in each case.