Retake exam — Programming in Python 2020–06–17

Duration: 08:00–13:00. 20 minutes extra for online submission.

Think about the following

- If you have registered for the room exam, use your registration code on the first page (above question A1)
- If you upload your copy to Studentportalen, fill in your anonymity code in the second box (you can also write both your registration code and your anonymity code).
- Upload your completed copy to Studentportalen. If that does not work, you can also send your completed PDF to adrien.coulier@it.uu.se. If you don't succeed in writing your answers in the PDF, write your answers in a text file or Word document. *Clearly indicate the question number and answer them in order.* As a last resort (if nothing else has worked due to technical difficulties), a scan or a picture of your answers may also be accepted.
- The solutions from previous questions can be built upon in the next questions, as long as not stated otherwise.
- It is permitted (and sometimes recommended) to introduce new methods and functions. The statement "Write a function that" does not mean that your answer should not be structured with the help of extra functions.
- All questions are related to the Python 3 programming language, and all your code should be written in this language. Extra care should be taken regarding readability (i.e. your code should be well structured and correctly indented). Variable, function, method and class names should be descriptive, but can usually be kept relatively short.

  Note that your grade can be negatively affected by, among other things:

  - unnecessary variables,
  - bad readability,
  - repetition of identical snippets of code,
  - failure to make good use of given snippets or code written earlier,
  - Badly ineffective code (for instance with many unnecessary function calls).

Important

This exam is made of two parts. The answers have dedicated boxes directly under each question. The boxes are sized so that they can contain a reasonable answer. If you still don't have enough space, use the extra pages at the end of this document. In this case, please make it clear that this extra space has been used.

There is extra space for task A1, if you were willing to comment on your choices.

Please leave the grading boxes empty!

Each task in the B part should be answered on a separate sheet, after the tasks' description. It is *your* responsibility to submit this exam digitally, with answers to all the tasks you wished to be answered. Please make sure that the program you use is capable of filling in a PDF form (so that you don't end up with just an empty file). Adobe Acrobat Reader is one of the softwares capable of saving and reading such files. Please remember to save your copy often and make sure that you upload the correct file in the end!

I will be available on Zoom from 10:30 to 11:30 to answer questions, at the following link: `https://uu-se.zoom.us/j/67543396212`. Outside this window, you can ask me questions using the email above.

Please continue to the next page.

Grading

To pass this exam (grade 3), it is necessary that the A-part be mostly correct. This does not mean that every task needs to be exactly right, but you do need to demonstrate that you fulfill the course's goals, which state that students, after passing the course, should be able to:

- indentify how programming and programming constructions, data types, modules and functions can be used to solve problems in enginnering or bioinformatics,
- implement such solutions by writing Python programs with several interacting components,
- use and describe the fundamental concepts module, function, class, object and related subconcepts,
- use an integrated development environment for testing and debugging

For grade 4, it is necessary that at least half of the tasks of the B-part are completed and correct. You should solve *all* the tasks of the B-part to get grade 5. The code quality is also taken into consideration for these grades.

Aids

A subset of important aspects of Python are gathered in the separate reference sheet.

Since this is a home exam, it is tolerated to use any available digital or physical resource. This includes running your code on your computer and looking up the documentation, course web page, and similar resources. However, *it is forbidden to ask questions or to communicate with anybody.* This exam should be completed individually. You *must* specify your source if you use a snippet of code from a specific example. Copying code without clearly stating its source is considered plagiarism. In other words, you can look up examples but you should formulate your own answers entirely.

If you don't remember exactly what a function is called or how a part of the Python syntax looks like, you can point it out and describe which assumptions you make. We judge both how you solve the problem ad how you can handle the Python programming language and it's functionalities.

Good luck!

Registration Code: | |
Anonymity Code: | |

1 Part-A (mandatory)

A1. Check the correct answer (only one solution is correct for each question, unless otherwise stated).

a) Which function from the `math` module computes the euclidian distance of a vector, stored as a list? See `https://docs.python.org/3/library/math.html`

1) `ceil`
2) `comb`
3) `dist`
4) `erf`
5) `hypot`
6) `lgamma`

b) Which of these statement about tuples is true?

1) All its elements must have the same type
2) Its values can only be set when the tuple is created
3) A tuple always has more than one element
4) If we use `print` on a tuple, we always have the same result since a tuple is immutable.
5) All strings are also tuples

c) What is `random` in the code below?

```
import random

x = random.randint(0, 100)
```

1) A function
2) A variable
3) A parameter
4) A module
5) A class
6) It is not possible to say

d) What is `Worker` in the code below?

```
x = Worker
y = x(999)
```

1) A function
2) A variable
3) A parameter
4) A module
5) A class
6) It is not possible to say

e) By convention, how does one call the first parameter of a method?

1) `me`
2) `__init__`
3) `this`
4) `self`
5) `append`
6) `0`

f) If the line below is already in the program (and cannot be removed), what is most important to also write in the program?

```
file = open('ourfile.txt', 'r')
```

1) `file.close()`
2) `text = file.readlines()`
3) `print(file)`
4) `for c in file.read()`
5) `with file.readlines() as lines:`

g) What is the difference between a function and a method

1) One can define their own functions
2) One can define their own methods
3) All methods are magic
4) A function is called on an object
5) A method is called on an object
6) Only functions can modify an object
7) Only methods can modify an object

h) How many lists does the following code create?

```
a = [1, [2, 3]]
b = a
c = a[1]
d = a[1][0]
e = a[1][1]
print(a.pop)
```

1) 1
2) 2
3) 3
4) 4
5) 5

i) What is the type of the following expression?

```
[1, 2, 3.5, 4][2]
```

1) `int`
2) `float`
3) `str`
4) `list`

Comments to my answers for A1:

Grading:

A2. The function `cube(lst)` takes a list as parameter and returns a new list with every value cubed (i.e. to the power of three). Write down this function, as well as an example that uses this function to compute the cubes of 5, 8 and 9.

Grading:

A3. The function `capitalize` takes a word as parameter (as a string) and returns a new version of this word where the first letter is capitalized. The other letters remain unchanged. For instance, `'hej'` becomes `'Hej'`. Write down the body of this function. Remember that `str` have a method `upper` that returns a new string where *all* the letters are capitalized.

```
def capitalize(word):
```

Grading:

A4. The body of a function is written below. Write down the beginning of the function's definition (it's signature) with name and parameters. Add a suitable docstring.

```
    count = 0
    for c in line:
        if c == 'Q':
            count += 1
    return count
```

Grading:

A5. The function `longestWord` returns the longest word from a string. Remember the method `split` can be used to split a string into a list of words. Write the body of this function.

```
def longestWord(data):
    """Returns the longest word found in input string data. Words are
     defined as being delimited by whitespaces."""
```

Grading:

A6. Define a new class `Point` describing coordinates in 2-dimensional room. Its constructor takes two parameters, `x` and `y`, and internally store them as a tuple in an attribute (an instance variable) `pos`. There should be *no* other attribute.

Grading:

A7. Write a new method for `Point`, called `rot90`. `rot90` rotates its point by steps of 90 degrees counter-clockwise. `p.rot90()` should then rotate `p` by 90 degrees, `p.rot90(2)` should rotate it by 180 degrees, and both `p.rot90(3)` and `p.rot90(-1)` rotates `p` by 270 degrees.

Tips: Rotating coordinates by 90 degrees is equivalent to a multiplication by i in complex numbers. Alternatively, the new coordinates (x_r, y_r) can be written as $x_r = -y,\ y_r = x$

Grading:

A8. Python uses special methods with specific names for specific functionalities. Complete the method `__eq__` for the class `Point`. `__eq__` is used by Python when comparisons are done using `==`. This method should return `True` if two points have the same coordinates, and `False` otherwise.

```
def __eq__(self, other):
    """Compares self to other, returning True if two Point instances
    represent the same coordinates, False otherwise."""
```

Grading:

A9. Assume that all the code you have written in `Point` is in a file called `smallvector.py`. Write the necessary code to use `Point` from *another* Python file, write two vectors with coordinates $x_1 = 3$, $y_1 = 5$; $x_2 = -5$, $y_2 = 3$, rotate the second vector by 270 degrees counterclockwise, and compare both vectors with `==` and print out the result.

Grading:

A10. The program bellow contains several minor typos. Write a correct version of the code with comments describing what was wrong. The function `type` returns the type of the value of its parameter.

```
lex = {}
lex{'a'} = 'hej'
lex['b'] = [5 9]
for k in lex.keys:
    print('Information for key {k}')
    print( f'Val: {lex[k]}, Type: {type(lex[k])}'
```

Grading:

Part-B (for grades 4 and 5)

These task should be completed using the separate answer sheets at the end of this document. Make sure to answer every task on a separate sheet.

B1. Write the body of the function below. This function should build a dictionary with keys and values. It continuously reads two lines from the standard input (i.e. from the keyboard). The first line is used to create the key, making sure all the letters are uppercase. The user specifies the value on the next line. This value should be read as a integer. If the user inputs an empty line as key, the function should stop and return the dictionary.

```
def read_uppercase_keys_and_ints():
    """Reads two lines per entry, one with key, one with value, stores all
    entries in a dictionary, which is returned. Keys are stored as UPPERCASE only,
    values as ints. Input is stopped by entering an empty line as key."""
```

B2. Assume you have a list, `plist`, where the elements are points from class `Point`, that you wrote in part A. We want to sort these points by angle relative to the origin $(0, 0)$, where point $(1, 0)$ has angle 0, and positive angles go counterclockwise. The sorted list should be stored as `plist2`. The original list `plist` should *not* be modified. Write a code that does that.

Tips: you can use the function `atan2` from the `math` module, which can be used to compute an angle according to the definition above. Here is the function's documentation:

math.atan2(y, x) Return atan(y / x), in radians. The result is between $-\pi$ and π. The vector in the plane from the origin to point (x, y) makes this angle with the positive X axis. The point of `atan2()` is that the signs of both inputs are known to it, so it can compute the correct quadrant for the angle. For instance, `atan(1)` and `atan2(1, 1)` are both $\pi/4$, but `atan2(-1, -1)` is $-3\pi/4$.

B3. All the remaining tasks concern two classes `FallingBall` and `RollingBall`. These perform a very simple simulation with timesteping, as shown in the code below. Your code should of course work if the constants from the example are modified or if one creates several objects of the same class.

```
balls = [FallingBall(10, 0, 0.5), RollingBall(10, 10, 0.2)]

timestep = 0.01
for i in range(0, 1000):
    time = i * timestep
    if i % 10 == 0 or i < 5:
        print(f'Time: {time:5.2f}')
        print(f'{balls}')

    for b in balls:
        b.step(timestep)
```

Instructions continue on the next page!

The beginning of the output from the program above should look as follows. The values from your solution should be exactly the same.

```
Time: 0.00
[FallingBall(10.0000, 0.0000), RollingBall(10.0000, 10.0000)]
Time: 0.01
[FallingBall(9.9990, -0.0981), RollingBall(10.0998, 9.9804)]
Time: 0.02
[FallingBall(9.9971, -0.1962), RollingBall(10.1994, 9.9608)]
Time: 0.03
[FallingBall(9.9941, -0.2941), RollingBall(10.2988, 9.9411)]
Time: 0.04
[FallingBall(9.9902, -0.3917), RollingBall(10.3980, 9.9215)]
Time: 0.10
[FallingBall(9.9464, -0.9675), RollingBall(10.9892, 9.8038)]
```

In our simple simulation, falling balls only have y-coordinates, while rolling balls only have x-coordinates. They are affected by the Earth's gravitation. Their constructor takes a position (either y- or x-coordinate), a speed, and an air drag or friction coefficient, respectively.

Write the beginning of the definition of these two classes, with a constructor that stores the necessary information in various attributes.

B4. Write the methods `__repr__` and `__str__` for both classes. They should return a string that describes the state of its object in the form `FallingBall(y, v)` and `RollingBall(x, v)`, where `x`, `y` and `v` are values from the actual object. `__repr__` and `__str__` should return the same thing, but make sure not to repeat your code unnecessarily.

a) Write both methods for `FallingBall`.

b) Write both methods for `RollingBall`.

B5, B6. For the next two tasks, we will implement the `step` method, that , given a timestep Δt (expressed in seconds), updates the object's speed and position for both classes. Generally speaking, both methods should compute the acceleration a and update v following $v_1 = v_0 + a\Delta t$. The position should be updated according to $p_1 = p_0 + v_1\Delta t$.

One must be extra careful here to make sure our simulation does not become unrealistic with this very simple timestepping scheme. When the falling ball touches the ground (at $y = 0$), it should keep its momentum and bounce upward, conserving its speed (also called an elastic choc). In other words, the downward speed changes sign to an upward speed.

The rolling ball is slowed down by friction. If you are not careful, the speed will change sign several times when approaching 0. Make sure that the ball stays still when the friction is strong enough to negate the speed within one timestep.

Remember that although you cannot compute too many timesteps by hand, you can use the example above to check that your code behaves properly, e.g. positive/negative signs, order of magnitude of the various values, or if they change faster as time goes on or not.

Instructions continue on the next page!

B5. Implement `FallingBall.step`.

On Earth, there is always a downward acceleration (negative y) of about 9.81 m/s^2. This means that downward speed increases by 9.81 m/s each second.

Our model also includes air drag. The acceleration from air drag in our model depends on the square of the current speed and on the air drag coefficient, that we specified in the constructor. The air drag force is of course directed in the opposite direction to the ball's velocity. When the ball goes upward, it is slowed down downward and vice versa.

Make sure your `step`-method makes the ball bounce each time it reaches $y = 0$. Motivate how you do that.

Tip: not taking the bounce into account, the acceleration can be described as $a = -9.81 - fv^2\text{sgn}(v)$, where f is the air drag coefficient, and the function sgn(v) is -1 if $v < 0$ and 1 otherwise. This function is not built-in in Python.

B6. Implement `RollingBall.step` The ball rolls with an initial speed and direction. It is continuously slowed down by friction.

The acceleration from friction is proportional to the normal acceleration and a friction coefficient. On a plane (like in our model), the normal acceleration is equal to Earth's acceleration, 9.81 m/s^2. The friction's acceleration goes against the current velocity.

Make sure your `step`-method stops the ball when the friction is strong enough to nullify the speed. Motivate how you do that.

Tip: the acceleration that applies to the ball can be described as $a = -9.81f\text{sgn}(v)$, where f is the friction coefficient.

Answer to B1:

Grading:

Answer to B2:

Grading:

Answer to B3:

Grading:

Answer to B4:

Grading:

Answer to B5:

Grading:

Answer to B6:

Grading:

Extra space, page 1:

Grading:

Extra space, page 2:

Grading:

Extra space, page 3:

Grading:

Extra space, page 4:

Grading:

Extra space, page 5:

Grading:

Extra space, page 6:

Grading: