CS 101: Introduction to Computer Science

Mahindra École Centrale

Duration: 1 hour 30 minutes | *First Mid-Term Exam* | Total 100 marks

February the 9th, 2015

Problem I (*Marks:* 40)

This first problem is a list of **Multiple Choice Question**. Each question (from Q.I.1 to Q.I.20) carries 2 marks, and has one or more correct answer(s). You need to write on your answer paper the correct answers, as something like : *Problem 1: 1) a, b 2) c 3) d ... 20) a, b, d* etc on your answer paper.

Qu.I.1) Which of the following results in a **syntax error** (printing `SyntaxError: invalid syntax`)?

a) `print '" Yes ... " , she said. '`

b) `print " He said, « Yes! »"`

c) `print 'Hum... " That"s okay ?'"`

d) `print ' 3\'`

Qu.I.2) What gets printed when Python execute that program:

```
1 x = True
2 y = False
3 z = False
4 if x or (y and z):
5     print "yes"
6 else:
7     print "no"
```

a) `yes`

b) `no`

c) `SyntaxError: invalid syntax`

d) `True`

Qu.I.3) What gets printed when Python execute that program?

```
1 x = '5'
2 y = 2
3 z = x + y
4 print z
```

a) `SyntaxError: invalid syntax`

b) `52` **c)** `7` **d)** `z`

Qu.I.4) What gets printed when Python execute that program?

```
1 a = 6
2 b = a/2
3 print b
```

a) `SyntaxError: invalid syntax`

b) `3` **c)** `6` **d)** `3.0`

Qu.I.5) What syntax can you use to *comment* out one piece of code in Python?

a) `* this line is a comment`

b) `(that is a comment)`

c) `// that line is a comment`

d) `# that line is a comment`

Qu.I.6) Which one of the following is *the beginning* of a valid Python `if` statement?

a) `if a >= 22:`
b) `if (a >= 22)`
c) `if (a => 22)`
d) `if a >= 22`

Qu.I.7) If you have a variable `example`, what can you do to check its type?

a) `getType(example)`
b) `Type(example)`
c) `type(example)`
d) `example.type`

Qu.I.8) Which block of the code will **never** get executed (ie. **regardless of the value of a** which is obtained from the user):

```
1 a = input("Value of a is ")
2 if a<5:
3     blockA
4 elif a>8:
5     blockB
6 elif a>20:
7     blockC
8 else:
9     blockD
```

a) `blockA`
b) `blockB`
c) `blockC`
d) `blockD`

Qu.I.9) How many times the following while loop will execute?

```
1 i = 0
2 while 1 < 2:
3     print i
4     i -= 1
```

a) Infinite number of time
(if no we do not interrupt the program)
b) 1 **c)** 2 **d)** 0

Qu.I.10) What is the output of the following program (defining the function `example` and print its result when applied to `"SmallString"`) ?

```
1 def example(s):
2     s = s + '2'
3     s = s * 2
4     return s
5
6 print example("SmallString")
```

a) `SmallStringSmallString2`
b) `SmallString2SmallString2`
c) `SmallStringSmallString`
d) `SyntaxError: invalid syntax`

Qu.I.11) Select all option(s) that print `CS101-is-very-cool`:

a) `print "CS101", "is", "very", "cool"`
b) `print "CS101", "is", "very", "cool" + "-" * 4`
c) `print "CS101-" + "is - very - cool"`
d) `print "CS101" + "-" + "is" + "-" + "very" + "-" + "cool"`

Qu.I.12) What is the output for the following program?

```
1 a = 2
2 b = 2.0
3 if a = b:
4     print "a and b are equal."
5 elif a > b:
6     print "a is greater than b."
7 else:
8     print "a is less than b."
```

a) `a and b are equal`
b) `a is greater than b`
c) `SyntaxError: invalid syntax`
d) `a is less than b`

Qu.I.13) What gets printed when Python execute that line: `print type(1j)`

a) `<type 'int'>` **b)** `<type 'str'>` **c)** `<type 'complex'>` **d)** `<type 'long'>`

Qu.I.14) On a *regular* computer (laptop, desktop, smartphone), what components take care of: computing (1), storing *temporary* data (2) and storing *permanent* data (3).

a) CPU (1), hard drive disk (2), RAM (3)
b) RAM (1), CPU (2), hard drive disk (3)
c) hard drive disk (1), RAM (2), CPU (3)
d) CPU (1), RAM (2), hard drive disk (3)

Qu.I.15) The following innovations are sorted in *historical order* (from the oldest to the more recent), which order is the correct one?

a) calculator machines, programming languages, the computer mouse, the Internet, Facebook
b) Facebook, the Internet, calculator machines, the computer mouse, programming languages
c) calculator machines, the Internet, programming languages, Facebook, the computer mouse
d) calculator machines, the computer mouse, the Internet, programming languages, Facebook

Qu.I.16) According to many historians, who is often considered as *the first programmer*:

a) Richard Stallman **b)** Bill Gates **c)** Lady Ada Lovelace **d)** Steve Jobs

Qu.I.17) In Python, executing that line `print "Result is:", (2/2)*(2**2)` will produce the output:

a) `"Result is:", 2` **b)** `4` **c)** `Result is: 1` **d)** `Result is: 4`

Qu.I.18) What is Python exactly?

a) a species of *snakes* and a family of computer,
b) a programming language, that you have to *buy* from a company,
c) a *text editor* to write programs, written in the Spyder language,
d) a free and open-source programming language.

Qu.I.19) What is the output for that piece of code?

```
1 print "First is:", 180678 % 5, "and second is:", 2015 < 1000
```

a) `First is: 36135.6 and second is: False`
b) `First is: 36135 and second is: 1015`
c) `First is: -2 and second is: True`
d) `First is: 3 and second is: False`

Qu.I.20) In Python, a **for loop** is written as something like (choose the good one(s)):

a)

```
1 for(int i = 0, i += 1, i < 100) {
2     # block to be executed for the value i (one or more line)
3 }
```

b)

```
1 # mylist is a list of values, like [-2, 3, 4, 2015]
2 for i in mylist:
3     # block to be executed for the value i (one or more line)
4 # End of the for loop
```

c)

```
1 # mylist is a list of values, like [-2, 3, 4, 2015]
2 for i in mylist do:
3     # block to be executed for the value i (one or more line)
4 done;
```

Problem II (*Marks:* 23)

Write two small programs to solve a linear and quadratic equation.

Qu.II.1) (*Marks:* **8**)

Write a (small) Python program that (1) ask the user values for two `float` numbers `a` and `b`, and then try to solve the *linear* equation $ax + b = 0$ for x, by (2) computing the solution x if it exists, and (3) printing it.
You will deal with the special case correctly, by printing an error message like `"No solution"` when it is needed.

Remark: the semantics of your small program (the *logic* of what you write) is more important that the *syntax*.

Qu.II.2) (*Marks:* **15**)

Similarly, write a (small) Python program that (1) ask the user values for three `float` numbers a, b and c, and then try to solve the *quadratic* equation $ax^2 + bx + c = 0$ for x, by (2) computing the solution(s) x if it exists or x_1, x_2 if they exist, and (3) printing the solution(s).
If there is no solution x (or x_1, x_2) to be printed, you can choose to print the text `"No real solution"`.

Remark: again, the *logic* of your program is more important that the *syntax*.

Problem III (*Marks:* 12)

As seen in last lectures and lab, the Python program below is defining a function called `approx_sqrt` that use the Babylonian method for computing an approximation of the square root of a number `x`.

This method is an iterative method, starting with a `guess` of the square root (`guess = x`), which has to be biger than the value of `x`, and then update that `guess` as long as necessary (cf. *line 7*).

The update of the guess is done by the formula `guess = (guess + x/guess) / 2.0` (cf. *line 9*).

Your job is to modify a little bit that program by checking that the value `number` given by the user (cf. *line 17*) is bigger than `0`.

- If `number >= 0`, your new program should do exactly as that initial program (compute an approximation of $\sqrt{\texttt{number}}$) and print it.
- If `number < 0`, your new program has to print a certain warning message (like `"Error: that number x has to be non-negative"`), and do nothing else.

Hint: You do not need to write additional lines of code for this. Instead, you can write a flowchart for the complete program that includes the below lines plus this additional check.

```
1  epsilon = 0.00001    # small number
2
3  def approx_sqrt(x):
4      guess = x  # We start with that guess = x
5
6      while abs( guess**2 - x ) >= epsilon :
7          # As long as the guess**2 is not close enough of x:
8          guess = (guess + x/guess) / 2.0
9          print "The current guess is", guess
10
11     # End of the while loop
12     print "Before returning, the last guess is", guess
13     return guess    # Exit the function, returning this guess
14
15 x = input("Enter a value x > 0, to compute approx_sqrt(x). Value x = ")
16 print "You chose the value", x, "for that number x."
17 x_sqrt = approx_sqrt(x)
18 print "The computed approximation is", x_sqrt
```

Problem IV (*Marks:* 25)

A function $x(t)$ (distance x as function of time t) is given by $x(t) = 10t + 5t^2$.

By evaluating the *derivative* of x (with respect to t) at $t = 5$, both *algebraically* and *numerically*, check that:

- For the *algebraic* evaluation, use the usual formulae.
- For *numerical* evaluation, use *backward*, *forward* and *central* differences, learnt in the MA101 course, with a time step $\Delta t = 1$.

Then compare each of the three numerically evaluated values against the algebraic value and decide which among them is most accurate. The above problem is solved in the program provided below. However, this program has a few errors, in particular, **five syntax** and **two semantic errors**. You have to spot and list out these errors. Additionally, you need to write out what the final printed output from the program would be if all the syntax and semantic errors were corrected.

- **Syntax errors** are grammatical mistakes in the language of the program – those for which the Python interpreter will throw errors. (*Note:* no errors related with indentation are present in that program.)
- Semantic errors represent mismatches between what the program is *supposed* to do, and what it is *actually* doing (Python will usually not detect such mathematical or logic errors).

Marks: One for spotting each Syntax error (there is 5), five for each Semantic error (there is 2), and 5 for correctly representing the final print statement from the program.

You need to write down in your answer sheet the locations in the programs where these errors are (line number, and how to correct these errors).

```
1  t = 5
2  xd_5 = 70  # xd_5 represents x'(5) the algebraic evaluation
3
4  # Initial values for the list D = [D[0],D[1],D[2]], for the first derivatives
5  D = [0, 0, 0]
6
7  # D[0], backward difference: x_back(t) = (x(t) - x(t-Delta_t)) / Delta_t
8  D[0] = ((10*t + 5*t**2) - (10*(t-1) + 5*(t-1)**2))/1
9
10 # D[1], forward difference: x_forw(t) = (x(t+Delta_t) - x(t)) / Delta_t
11 D[1] = ((10*(t+1) + 5*(t+1)**2) - (10*t + 5*t**2))/1
12
13 # D[2], central difference: x_cent(t) = (x(t+Delta_t) - x(t-Delta_t)) / (2 ↩
       Delta_t)
14 D[2] = ((10*(t+1) + 5*(t+1)**2) - (10*(t-1) + 5*(t-1)**2))/2
15
16 min_err = None   # Will contain the smallest error of these 3 approximations
17 kmin = 0
18
19 for k in range(0, 3):   # k will be 0, then 1, then 2
20     if (min_err is None) or (abs(D[k] - xd5) < min_err):
21         min_err = abs(D[k] - xd5)
22         kmin = k
23
24 if (kmin == 0):
25     method_used = "Backward"
26 elif (kmin == 1)
27     method_used = "Forward"
28 else
29     method_used = "Central"
30
31 print "The smallest error is", min_err, "obtained for kmin = ", kmin, "that is↩
       ", method_used, " difference"
```