CMPT 120 — Practice Questions

Spring 2014, Jérémie O. Lumbroso

These functions do not exhaustively cover the entire program of this course, but they are a pretty darn good practice list! You should complete your studying by redoing all CodeWrite exercises, and exams given thus far.

1. Definitions and elementary functions

These questions span the entire class, but mostly only require superficial knowledge of the notions that are covered (by opposition to coding questions, for instance, which require how to assemble everything together in a new way).

Q1-A. What does the `print` statement do?

Q1-B. Explain what is an `if-else` statement?

Q1-C. What does the `range` function do? In what context have you used it?

Q1-D. What is the difference between = and ==? Give a usage example for each.

Q1-E. What is the difference between an `if-elif` statement and an `if-if` statement? Use a flowchart to explain yourself if you find it helpful. Provide an example.

Q1-F. How do you write an empty string in Python? an empty list?

Q1-G. How can you print all integers between 0 and 10 using a `for` loop?

Q1-H. How can you print all integers between 0 and 10 using a `while` loop?

Q1-I. Suppose `unk` is a variable. Is `unk[:]` cause an error, or is it a correct Python expression. Is there any restriction on what kind of variable `unk` can be, and if so for what must be the data type of `unk` (float, integer, string, etc.).

Q1-J. In what contexts have you seen the `in` keyword? Give a different example for each possible usage.

Q1-K. What operator would you use to determine if an integer variable contains an even or odd number?

Q1-L. Let `unk` be a variable, explain what `unk * 4` is for each of the following data types: if `unk` is an integer? a float? a string? a boolean (`True` or `False`)? a list?

Q1-M. The expression `unk/2` gives the result `34`. What does `unk` contain?

Q1-N. What is the difference between a `for` loop and a `while` loop?

Q1-O. What is a variable? Give different examples of variables in Python.

Q1-P. What is the `def` keyword used for?

Q1-Q. Can a `for` loop be infinite? If so, give an example of such an infinite loop.

Q1-R. What is a float? an integer? What is the difference between those types?

Q1-S. What is a flag variable - what is its defining property? In what contexts would you use one?

Q1-T. Suppose you have an infinite loop (for instance, `while True:` etc.), how can you exit from this loop - which keyword can you use?

Q1-U. Suppose is `unk` a list of size 10, after doing `unk[0]=0`, what is the value of `unk[0]`? Now suppose `unk` is a string, would your answer be the same, and if not, what would change?

Q1-V. How can you get the length of a string?

Q1-W. What does the following expression compute: `"My age is: " + 27`

Q1-X. What are two ways of iterating through the characters of a list using a `for` loop? What are the advantages/disadvantages of each method?

Q1-Y. In what context is the `return` keyword used? Are there situations where it is mandatory?

Q1-Z. Give an example of a function that returns a value, and of a function that does not return a value.

Q1-AA. What is an accumulator variable? Give a simple example where an accumulator is used.

Q1-BB. How can you convert an integer to a string? Give an example of when this is useful.

Q1-CC. How can you convert a string to an integer? Give an example of when this is useful.

2. General algorithmic questions (no Python)

Here are some questions, which don't involve the Python programming language, but rather the beginning of the course, and the general understanding of programming. (Examples of algorithms are given, for example, in Lecture 3.)

Q2-A. What is an algorithm? What is a type of algorithm that most people use in their everyday life (that is not related to computers)?

Q2-B. Explain precisely how to draw a triangle, considering the person you are explaining to, only knows how to draw straight lines, and rotate by a given angle.

Q2-C. Same question as above, except you now must draw n sided polygon.

Q2-D. Given a set S of numbers, provide an algorithm to compute the sum of S.

Q2-E. Given a set S of numbers, give an algorithm to compute the average of S.

Q2-F. Given a set S of strictly positive numbers, provide an algorithm to find the largest element of S.

Q2-G. Given a set S of numbers, provide an algorithm to compute the sum of S.

Q2-H. Let w be a word. Give an algorithm to determine if w is a palindrome (a palindrome is a word that can be read both from left to right and from right to left, for instance “madam” is such a word; as is “civic”).

Q2-I. Give an example of specific algorithm that you use in your every day life.

Q2-J. Give an example of an algorithm of every day life you have recently explained to someone

3. Early Practice Questions

These questions the very first lectures (no strings, no flag variables).

Q3-A. Suppose *n* is a variable containing a strictly positive integer. What does it contain at the end of the following code:

```
n = n*2
n = n+568
n = 2*n
n = n+120
n = n/4
n = n-314
```

Q3-B. Same question with the following code:

```
n = n/2
n = n+284
n = 2*n
n = n+120
n = n-688
```

Q3-C. Suppose the variable `my_height` contains the height in centimeters of a student in the class, to ride *Space Mountain* a person needs to measure at least 102cm. Write a program that will print "Go on champ!" if the student is tall enough, and "Sorry, next time!" if not.

Q3-D. Same as above, except we now also have a variable `my_country` (which can either be "USA" or "France"); the requirement for *Space Mountain* is different in France: a person must measure at least 132cm. Write a program that takes into account `my_height` and `my_country` before letting the student know if he can go on the ride.

Q3-E. Write a *function* that takes no parameters and prints "Hello World."

Q3-F. Write a *program* to sum all integers between 1 and 250.

Q3-G. Write a function `is_odd(n)` that returns True if the strictly positive integer `n` is odd, and False if not.

Q3-H. Write a function `say_hello(name)` which takes one string parameter `name`, and prints the message "Hello " and the string contained in `name`. (For instance `say_hello("Johnny boy")` prints `"Hello Johnny boy"`.)

Q3-H. Write a program to sum all integers between 1 and 250, which can be divided neither by 7 nor 13.

Q3-I. Write a program to sum all integers between 1 and 250, which can be divided by both 7 and 13.

Q3-J. Write a function `make_hello(name)` which takes one string parameter `name`, and returns the message "Hello " and the string contained in `name`. (For instance `make_hello("Johnny boy")` returns `"Hello Johnny boy"`.)

Q3-K. Write a function `max_of_two_values(a, b)` which takes two parameters `a` and `b`, both of which are numbers (integers or floats, or a mix), and returns the largest of the two numbers.

Q3-L. Write a function `max_of_three_values(a, b, c)` which takes three parameters `a`, `b` and `c`, all of which are numbers (integers or floats, or a mix), and returns the largest of the three numbers.

Q3-M. Write a function `myfun(a, b, c)` which takes three parameters `a`, `b` and `c`, all of which are numbers (integers or floats, or a mix), and returns True only if the average of the three numbers is larger than two times the maximum; it returns False otherwise.

Q3-N. What is the value of `valA` after the code below has executed?

```
valA = 1
for i in range(1, 4):
  for j in range(1, valA+1):
    valA = valA + i
```

Q3-O. What is the value of `valA` after the code below has executed (note that the difference with the code above is that we are now doing multiplications instead of additions)?

```
valA = 1
for i in range(1, 3):
  for j in range(1, valA+1):
    valA = valA * i
```

Now consider `valB` (the range of the outer `for` loop changed, it is now up to 4):

```
valB = 1
for i in range(1, 4):
  for j in range(1, valB+1):
    valB = valB * i
```

Without computing the actual value of `valB`, can you tell whether `valB` is 1000 times larger than `valA` or not?

Q3-P. (PeerWise question.) What would be the output and type of this entire Python statement: **`print`** `"float(5*1**2)"`? What Python function can tell you the type of an expression? (If you are in doubt with you answer, use it to find if you are correct.)

4. Flag, Accumulators, and Strings

These questions now introduce advanced string manipulation, and thus also include functions and questions with flag variables and accumulators, both related to strings and not.

Q4-A. Suppose `s` is a string, in what case does `s[0]` returns an error?

Q4-B. Write a function that takes a string as a parameter, and returns its last character if it exists, and an empty string if not.

Q4-C. Write a function that takes a string as a parameter, and: if the string is empty, it returns an empty string; if the string has an odd number of characters, it returns the middle character (with the string `"david"` it returns `"v"`); if the string has an even number of characters, it returns the middle two characters (with `"hello"`, the function returns `"ll"`).

Q4-D. What built-in Python function can tell you if a string is all lowercase characters? Give an example of usage of this function. Now write a function that takes a string, and returns True if it is all in lowercase, and False if not - without using the built-in Python function used for this purpose (you may however use other built-in functions).

Q4-E. Write a function that counts the number of uppercase characters in a string.

Q4-F. Consider a completely new Python shell, which of the following expressions are valid strings, and which are not (error or not a string).

- `"this is a string"`
- `'this is maybe a string'`
- `'c'`
- `['t', 'h', 'i', 's']`
- `"ab" + "cd"`
- `"ab" + 'cd'`
- `'ab' + 3`
- `str(3) + "abc"`
- `int(3) + "4"`
- `3 + int("4")`
- `"'"`
- `"'" + '"'`
- `"4" * 3`

Q4-G. How can you compare the first character to the last character?

Q4-H. Write a function that takes a string, and switches the case of each character: makes uppercase characters lowercase and vice-versa (for instance `"Hello"` becomes `"hELLO"`).

Q4-I. Let `s` be a string, write code so that the first and last characters are removed from `s`.

Q4-J. Write a function that takes a string, and returns the number of vowels it contains.

Q4-K. (PeerWise question.) To promote the greatness of coding using Python, SFU has decided to finally create a superior, ultra-stable, super-fast, password encrypted Wi-Fi connection exclusively for its students. The Wi-Fi connection is called "SFUNET-CODEIT", and using your knowledge of coding in Python, its password can only be obtained by solving the code below.

```
def wifi_pass(value):
     password = ""
     for j in range(len(value)):
          if j%2 != 0:
               password = password + value[j]
     print password

wifi_pass("lyko3utshh6aal2lvnbo1thpxa8sas")
```

What is the password for SFUNET-CODEIT?

Q4-L. (PeerWise question.) Mandy has some cupcakes leftover after her party. She decides to give these to her friends whose first name either starts with a vowel or whose first name has at least 5 letters. Mandy wants to write a function that, given a friend's name, will return whether to give them a cupcake or not. She is not sure which of her following attempts is correct:

```
def pick(Name):
    for ch in Name:
        if ch== 'A' or ch== 'E' or ch=='I' or ch=='O' or ch=='U':
            print 'cupcake'
        elif len(Name)>= 5:
            print 'cupcake'
        else:
            print 'no cupcake'


def pick(Name):
    for ch in Name:
        if ch= 'A' or ch= 'E' or ch='I' or ch='O' or ch='U':
            return 'cupcake'
        elif len(Name)>= 5:
            return 'cupcake'
        else:
            return no cupcake
```

```
def pick(Name):
    for ch in Name:
        if ch== 'A' or ch== 'E' or ch=='I' or ch=='O' or ch=='U':
            return 'cupcake'
        elif len(Name)>=5:
            return 'cupcake'
        else:
            return 'no cupcake'


def pick(Name):
    for ch in Name:
        if ch== 'a' or ch== 'e' or ch=='i' or ch=='o' or ch=='u':
            return 'cupcake'
        elif len(Name)>= 5:
            return 'cupcake'
        else:
        return 'no cupcake'


def pick(Name):
    for ch in Name:
        if ch== "A" or ch== "E" or ch=="I" or ch=="O" or ch=="U":
            return 'cupcake'
        elif len(Name)> 5:
            return 'cupcake'
        else:
            return 'no cupcake'
```

Q4-M. What are the two types of quotes that can be used to describe strings.

Q4-N. Write a function that takes a string and removes all punctuation (period, comma, colon, semi-colon, exclamation point, question mark and... both single and double quotes).

Q4-O. Write a function that takes a string, and counts the number of words it contains (a word is defined as any alphabetical sequence of characters separated by non-alphabetical characters, for instance `"hello there"` contains two words; `"he...does...not...like...me"` contains five words; `"matter-of-fact"` contains three words).

Q4-P. Write a function `pirate_talk(s)` which takes a string `s`, and inserts the string `" Arrr!"` after every period or exclamation point, and replace `"you"` by `"ya"` (respecting the case - so `"You"` must be replaced by `"Ya"`).

Q4-Q. Write a function that takes a string and removes all letters - and only letters - that are repeated (for instance `"helloo"` should become `"helo"`).

Q4-R. Write a function that reverses a string.

Q4-S. Write a function `matching_parentheses(s)` that detects if the parentheses in the string s are matching. `"((aaa))"`, `"(aaa)(aaa)"`, `"(aaa()aaa)"` are examples of matching parentheses; `"(aaa)("`,

"`(aaa()aaa))`" are not. *Tip: when reading the string from left to right, there can never be more closing parentheses than opening parentheses.*

Q4-T. Write a function that takes a string containing matching parentheses, and checks that there is at least one letter "`a`" in between each set of parentheses (it returns True if that is the case, and False otherwise). For instance "`((bbbba))`" is a string which would return True; "`((a)) to (aaabbb)`" would also return True; "`(aa) and (bb)`" would return False because the second pair of parentheses does not contain an "`a`" between them. *Note: you do not have to check that the parentheses are matching, as this is an assumption.*

Q4-U. Write a function that removes one character out of two of a string.

Q4-V. Write a function that inserts a "/" after every five characters ("`Hello there`" becomes "`Hello/ ther/e`").

Q4-W. Consider this little bit of function:

```
def find_xxxx_post(s, x):
  # ...
  for i in range(len(s)):
    if x == s[i]:
      #...
```

How can you modify it so that it returns the *first* occurrence of the *character* `x` in the string `s`? the *last* occurrence?

Q4-X. Write a function that takes a string containing letters and spaces, and prints all words that are non-empty and contain only lowercase letters.

Q4-Y. Write a function `weirdassfun(s, x, p)` that prints the string `s` only if all characters in a position that can be divided by `p` are equal to `x`.

Q4-Z. Write a function that `find_substr(s, patt)` that takes two ***strings*** `s` and `patt` (take care that `patt` is not a single character as before), which returns True if the string `s` contains the substring `patt`, and False otherwise. You may not use the Python built-in function that does exactly this.

Q4-AA. Write a function that `replace_substr(s, patt)` that takes two ***strings*** `s` and `patt` (take care that `patt` is not a single character as before), which returns True if the string `s` contains the substring `patt`, and False otherwise. You may not use the Python built-in function that does exactly this.

Q4-BB. Write a function `reverse_substr(s, patt)` that reverses the first occurrence of `patt` in `s`, and returns the modified string. For instance `reverse_substr("python are cool", "are")` return "`python era cool`".

5. Turtles

A few questions on how to use turtles. Recall that you must first import the turtle module, and then commands are `turtle.fd`*,* `turtle.rt`*, etc.*

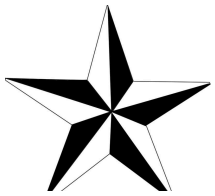
Q5-A. Write a program that draws a triangle of side 10.

Q5-B. Write a function that takes one integer parameter `n`, and another integer parameter `side`, and draws a *n*-sided polygon of side `side`.

Q5-C. Write a function that draws a square. Then using this function, write a function that takes an integer parameter `k`, and draws a *k* by *k* grid.

Q5-D. Write a function for each of these stars (or write a function that takes an integer parameter *k* and draws a star with *k* spikes):

Q5-E. Now write a function that draws the following star, using the function `turtle.fill` to fill the black regions:



Q5-F. Write a function that draws a spiral.

Q5-G. Write a function draws ten squares, which share the same center, but are of increasing size (you choose how the sizes are increasing).