COMP10001 Foundations of Computing
Semester 1, 2022

Tutorial Questions: Week 12

— VERSION: 1477, DATE: MAY 23, 2022 —

Discussion

1. What are some of the (numeric) bases we can store numbers with? How can we convert between bases in Python?
2. Why are binary and hexadecimal convenient for computers? Why is decimal more difficult?
3. What is "Unicode"? What problems does it solve?
4. What are some encodings which we can use to represent text?

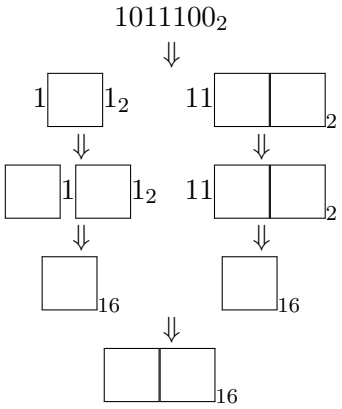
   Now try Exercise 1
5. Revise some of the points in the ACM (Association for Computing Machinery) Code of Ethics and Professional Conduct. Why is it important to consider Ethics when studying computing?
6. What does "dual use" refer to in the context of computing?

   Now try the previous exam questions

Exercises

1. Convert the following binary numbers into hexadecimal.

   (a) Convert the binary number 1011100 to hexadecimal by filling in the boxes in the following diagram with a single digit:



   (b) Convert the binary number 111101001 into hexadecimal using a method like the one shown above.

Problems

1. Write a function which takes a string (which is composed of the first 128 code points of Unicode) and returns it as a sequence of binary ASCII values, stored as a string of binary digits in which each character should be represented with 8 digits. Write another function which takes this string of 1s and 0s and converts it back into the original string of text. `ord()`, `chr()` and `int(_, 2)` should be useful.
   `to_binary('a_1')` should return `'011000010101111100110001'`.
   `to_text('011000010101111100110001')` should return `'a_1'`.

Previous Exam Questions

1. Use the following operations to write a single expression which evaluates to the prescribed output. There should be no redundant operations, i.e., removing any required operation should change the functionality:

   (a) Output = `'&entity;'`
   - String concatenation
   - `.lower()` method
   - Dictionary lookup

   (b) Output = `True`
   - `set()` function
   - List concatenation

   (c) Output = [3, 4, 5]
   - `range()` function
   - List slicing
   - `int()` function

2. The following function is meant to take an integer num and decompose it into k-digit sub-sequences (noting that the first integer could be made up of less than k digits in the case that the first digit is 0), map each sub-sequence back into a character based on its ASCII value (`65 = 'A'`, `66 = 'B'`...) and compose the characters into a string. The following is an example function call which illustrates its intended behaviour:

```
>>> print(num2txt(97097114103104))
aargh
```

   Identify exactly three (3) errors in the code (using the provided line numbers), determine for each whether it is a "syntax", "run-time" or "logic" error, and provide a replacement line which corrects the error.

```
1  def num2txt(num, k=3):
2      numstr = str(num)
3      txt = ""
4      mismatch = numstr % k
5      if mismatch:
6          numstr = "0" * (k - mismatch)
7      start == 0
8      for end in range(k, len(numstr)+1, k):
9          txt += chr(int(numstr[start;end]))
10         start = end
11     return txt
```

3. The function `reverse_records(csv_filename, new_filename)` takes a string `csv_filename` containing the filename of a csv file, and copies the contents of that file to a new csv file, whose name is given by `new_filename`, in the following manner. The header record of the input file is copied first. Then the order of the remaining records is reversed, so that the last record in the input file is saved first, then the second last record, and so on.

   For example, if the input csv file contains:

```
col1,col2,col3
1,2,3
4,5,6
7,8,9
```

   then the output csv file will contain:

```
col1,col2,col3
7,8,9
4,5,6
1,2,3
```

   Provide code to insert into each of the numbered boxes in the code on the next page to complete the function as described. Note that your code will be evaluated at the indentation level indicated for each box.

```
import [ 1 ]
def reverse_records(csv_filename, new_filename):
    csv_file = [ 2 ]
    reader = csv.reader(csv_file)
    header = [ 3 ]
    data2d = list(reader)
    newdata2d = [ 4 ]
    csv_file.close()
    new_file = open(new_filename, "w")
    writer = csv.writer(new_file)
    writer.writerow(header)
    writer.[ 5 ]
    new_file.[ 6 ]
```