FIT1045/FIT1053 Algorithmic Problem Solving – Tutorial 3.
Solutions

Objectives

After this tutorial, you should be able to:

- Interpret and use while-loops and for-loops.
- Use appropriate control-flow tools to solve simple problems.
- Manipulate sequence types: list, string, and range.
- Implement and use tables in Python.

Prepared Question

1.
```
def hours_to_legal_limit(bal):
    hours = 0
    while bal >= 0.05:
        hours += 1
        bal -= 0.015
    return hours
```

2. The function requires that we check whether a condition has been reached, and do the same instruction (add 1 hour) every time the condition is not met. This means the task is more suited to a `while`-loop.

Warm-up

Do not solve these questions by running in Python.
Try running the code using Python Tutor after having a go for further understanding
Link: http://www.pythontutor.com/

Loops:

- Write a function, `adding(x)` that takes an integer as input, and returns the sum of all even numbers from 0 to `x` (inclusive). Use a for-loop in your function. (E.g. `adding(10)` returns 30, because 2+4+6+8+10=30.)
- Write a function, `double(x)`, that takes an integer as input, and returns the number of times `x` must be doubled before the resulting number is larger than 100. (E.g. `double(5)` returns 5, because `x` doubled once is 10, doubled twice is 20, ... , doubled five times is 160.)

Range: What sequence of numbers is equivalent to the given ranges?

- `range(2, 11, 3)`
- `range(0, -3, -1)`

Loops (cont.): Identify what is wrong with the given loops, and rewrite the loops to reflect the intended behaviour.

-
```
my_list = [3, 7, 4, 9, 12, 2]
for num in my_list:
    num = num//2
```

-
```
my_string = 'hare paws wall tuba draw'
for i in range(len(my_string)):
    if my_string[i] == 'a':
        my_string[i] = 'e'
```

Lists: What does `x` yield at the end of the code block?

- ```
  my_list = ['1045', '1008', '2004', '2099']
  x = my_list[1:3][-1]
  ```
- ```
  x = ['monkey', 'tiger', 'lion', 'mouse']
  animals = x
  animals[3] = 'meerkat'
  ```

Nested lists and Tables:

- How would you access the value `10`:
  ```
  nested_list = [1, 2, [3, 4, [5]], 6, 7, [8, 9, 10], 11, 12]
  ```
- Implement a nested loop in Python to create the following table:
  ```
  table = [[1, 2, 3], [1, 2, 3], [1, 2, 3]]
  ```

List of Lists

What you would find is that every element of the `outer_list` refers to the same `inner_list`. When you follow the links you would see we are overwriting the same list contents again and again, changing `inner_list` from [0,0,0,0] to [0,1,2,3] to [4,5,6,7] to [8,9,10,11] to [12,13,14,15]. For `make_my_list` to do as the programmer likely intended, you will need N many distinct inner lists.

Multiplication Table

```
def multiplication_table(x):
    table = []
    for num1 in range(1, x+1):
        line = []
        for num2 in range(1, x+1):
            line += [num1 * num2]
        table.append(line)
    return table
```

There are many ways to implement the extension option, but a simple way using the tools you currently know is shown below. This could be made more concise by using decomposition.

```
def maths_table(x, operator):
    table = []
    for num1 in range(1, x+1):
        line = []
        for num2 in range(1, x+1):
            if operator == '*':
                line += [num1 * num2]
            elif operator == '/':
                line += [num1 / num2]
            elif operator == '+':
                line += [num1 + num2]
            elif operator == '-':
                line += [num1 - num2]
        table.append(line)
    return table
```

Loop Challenges

1. ```
   def padded():
       my_list = []
       for num in range(1, 101):
           number = str(num)
           while len(number) < 3:
               number = '0' + number
           my_list += [number]
       return my_list
   ```

2. ```
   def binary(number):
       count = 0
       for num in number:
           count += int(num) #could also use a conditional expression
       return count
   ```

3.
```
def even(my_list):
    seen = []
    for i in range(len(my_list)):
        if my_list[i] not in seen:
            seen += [my_list[i]]
            count = 0
            for j in range(i, len(my_list)):
                count += (1 if my_list[i] == my_list[j] else 0)
            if count%2 != 0:
                return False
    return True
```

Checkpoint

1. What is the definition of a *computational problem*?
2. What range best reflects the sequence `10, 7, 4, 1, -2`? (There are multiple correct answers.)
3. Write a Python function, `double`, that takes a list of integers as arguments, and returns the same list with all values doubled.
4. Write a Python function, `make_table`, that takes an integer `x` as an argument, and returns an `n` by `n` table, where `table[i][j] == i+j`. For example, `make_table(3)` would return `[[0,1,2],[1,2,3],[2,3,4]]`.

Solutions

1. "A *computational problem* is a typically **infinite** collection of questions (called inputs or instances), each of which has at least one correct associated answer (called output)." Note: wording may be different, but key underlined concepts should be present.
2. `range(10, -3, -3)`, or `range(10, -4, -3)`, or `range(10, -5, -3)`.
3. ```
   def double(my_list):
       # cannot use 'for i in my_list'
       for i in range(len(my_list)):
           my_list[i] *= 2
       return my_list
   ```
4. ```
   def make_table(n):
       t = []
       for i in range(n):
           row = []
           for j in range(n):
               row += [i+j]
           t.append(row)
       return t
   ```