FIT1045/FIT1053 Algorithmic Problem Solving – Tutorial 3.

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

Alcohol is a drug that is processed by the body in a relatively predictable way. Because alcohol can impair an individual's driving, in Australia you are not legally allowed to drive unless your blood alcohol level (BAL) is below 0.05%. It takes approximately one hour for your BAL to decrease by 0.015.

1. Implement in Python a function, `hours_to_legal_limit(bal)`, that takes an individual's current BAL and returns an integer representing the number of hours that individual must wait before they can legally drive, rounded to the larger number (we want to be on the safe side!). You must use either a for-loop *or* a while-loop inside your function. The function can be handwritten.
2. Explain why you chose to use one kind of loop instead of the other kind. The explanation can be short, but it must be meaningful.

You must attempt both parts to receive the mark.

Workbook

§2.2 Loops, §3 Sequences, and §4.1 Tables.

While your tutors mark the prepared question, attempt the above Workbook sections. Your tutors will go through some of these questions with the class.

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

Consider the following Python code:

```
def make_my_list(N):
    inner_list = [None]*N #create inner_list with room for N elements
    outer_list = [None]*N #create outer_list with room for N elements
    for i in range(0, N):
        inner_list[i] = 0
    for i in range(0, N):
        outer_list[i] = inner_list
    for i in range(0, N):
        for j in range(0, N):
            outer_list[i][j] = N*i + j
```

Assume `N = 4`. In your group conduct a code trace on the whiteboard or on paper by doing the following:

1. Create two rectangles for `inner_list` and `outer_list` respectively with room for four squares within them;
2. Step through the algorithm creating links between `inner_list`/`outer_list` and their contents;
3. As contents change, write out the new value and change the links to match.

Discuss your findings with your group.

Multiplication Table

Implement a function in Python, `multiplication_table(x)`, that takes an integer, `x`, as input, and that returns an `x` by `x` multiplication table. For example, `multiplication_table(3)` should return `[[1,2,3],[2,4,6],[3,6,9]]`. While implementing this function, you should consider the following:

- the appropriate range of your loops;
- how to nest your loops;
- how to construct a table in a manner that is 'safe' for Python.

Extension: Add another parameter to your function to create `maths_table(x, operator)`. The new parameter takes a string of a mathematical operator (e.g. `'*'` for multiplication, `'+'` for addition, `'/'` for division, etc.) and the function returns an `x` by `x` table where `table[i][j]` is equal to `i+1 <mathematical operator> j+1`. For example, `maths_table(3, '*')` should return `[[1,2,3],[2,4,6],[3,6,9]]`.

Loop Challenges

1. Write a function that returns a list of the numbers 1 to 100 in string form, where every number is zero-padded so that it has exactly three digits. (I.e. the function should return the list `['001', '002', ..., '010', '011', ..., '099', '100']`.)
2. Write a function that takes a string representation of a binary number and returns the number of `1`s in the number. (I.e. inputting `'1001011'` would return `4`.)
3. Write a function that takes a list and returns `True` if all items in the list occur an even number of times; otherwise it returns `False`. (I.e. inputting `['a', 'b', 'a', 1, 1, 'b']` would return `True`; inputting `['a', 'b', 'c', 'b']` would return `False`.) Do not change the input list.

Extension: Write a function that takes a positive integer and returns the smallest positive integer where the sum of the two integers is a prime. (I.e. inputting 25 would return 4 because 25+4=29, and 29 is the first prime number after 25.) Once you have the function working, look up the Wikipedia article on the Sieve of Eratosthenes and see if you can improve on your implementation.