THE UNIVERSITY OF SYDNEY

INFO1110 / COMP9001 Week 8 Tutorial

Programming idioms

Programming Idioms and Patterns

During this tutorial we will be revising on concepts we have encountered over the semester. It will cover concepts such as loops, files, strings, command line arguments and typical patterns that are used.

Loops, Ifs and common problems

Given the following code segments, discuss with your class and tutor the issues with them.

```
x = 10
if x > 10:
        y = 20
else:
        print('x is less than 10')
print(y)
```

- What issues can you spot?
- How could we fix them?

```
l = [1, 2, 3, 4, 5, 6]
for i, j in enumerate(l):
        if i == 2:
                del l[i]
```

- What issues can you spot?
- How could we fix them?

```
def populate_list(l):
        if l == None:
                return False
        i = 0
        while i < 10:
                l.append(i)
                i += 1
        return True

x = 10
l = []
if (x > 10) or populate_list(l):
        print(l[0])
        print(l[1])
        print(l[2])
```

- What issues can you spot?
- How could we fix them?

Loop Patterns

As covered in previous tutorials, while and for have two very different semantics in regards to python. `while` loops depend strictly on a condition. This condition is some boolean expression or variable.

Given the following `while` and `for` loop statements, outline the similarities between the loops and check to see if they are different

```
for i in reversed(range(40)):
        print(i)

i = 40
while i >= 0:
        print(i)
        i -= 1
```

Question 1: Generating numbers

You are to write a program that will output a sequence of even numbers up until `n`. `n` is provided as a command line argument.

Example, n is 4

```
python3 even_numbers.py 4
2 4 6 8
```

Example, n is 10

```
python3 even_numbers.py 10
2 4 6 8 10 12 14 16 18 20
```

You should use the modulus operator and a `while` loop.

After you have implemented this simple loop, incorporate a change that will allow you to skip a certain number of elements in the sequence.

```
python3 even_numbers.py 10 3
8 10 12 14 16 18 20 22 24 26
```

Question 2: Convert for loop to while

You will write code that will take the existing for loop and convert it to version that uses a while loop. For every instance you see `for` you will need to replace it with a corresponding while and create the required variables for iterating.

```
n = 10
l = []
for i in range(10):
        l.append(i)


for i, e in enumerate(l):
        if e % 2 == 0:
                l[i] = l[i]*2

print(l)
```

Question 3: Searching for an element

You will write a function that will search a list for an item. If the item does not exist, your function will return `-1`. If it does exist, your function will return the index it exists at.

This function will be used in a program of your own choosing to search for an element in the list.

For example:

```
python3 find_item.py
Yes! I found it at index 5
```

Question 4: Search for substrings

Write a program that will search a file with a given string and output all the lines that string exists on.

For example:

```
python3 substring.py text.txt hello
3. She said hello to the person walking by
5. they waved back and replied by saying "hello"
```

Question 5: Convert list to a string

Without invoking the implicit string function on list and using a loop. Write a function called `list_to_string(items)` that will iterate through all the elements in a list and create a string of all the items.

```
l = [1, 2, 3, 4, 5, 6]
s = list_to_string(l)
print(s)
```

Your program should output:

```
1, 2, 3, 4, 5, 6
```

2D Lists and beyond

Lists are a very fundamental data structure in python and has a large use of practical applications. We are not only limited to just one dimensional lists, we can create lists of n-dimensions assuming it is required.

However, initialisation of a list can be performed in a multiple of ways in python. Given the following segments of code, what is the difference between implementations and syntax.

```
l = [0]*4
print(l)
```

```
l = []
i = 0;
while i < 4:
        l.append(0)
print(l)
```

```
l = [0, 0, 0, 0]
print(l)
```

Discuss with your class and tutor about the differences and when you may use one or the other.

2d lists are very common because we have a number of practical applications for them. A common case is to be able to represent a matrix and navigate it based on row and columns.

We can initialise a two dimensional list by creating a list and then iterating through it and append an element to each.

```
l = []
for i in range(4)
        l.append([0, 0, 0, 0]
```

We observed in the previous section how to intialise a list using the multiply operator. What issues do you observe with this initialisation?

```
l = [[0]*4]*4
```

Does everything seem okay? Observe closely what happens when we modify an element in one of the lists.

Question 6: Treating a 2D list as a 1D list

We want to introduce a method of representing a two dimensional list as a one dimensional list using **row-major** order. As long as we have the same number of elements as a two dimensional list we can navigate it in a similar manner.

Convert the following two dimensional list to a one dimensional list and change the iteration method

```
l = [None] * 4
i = 0
width = 4
height = 4

while i < height:
        l[i] = [0]*4

l[0][1] = 1
print(l[0])

i = 0
while i < height:
        j = 0
        while j < width:
                print(l[i][j], end='')
        print()
```

Example of accessing an element in row 1 and column 2 when represented as a one dimensional list

```
print(l[(1 * width) + 2])
```

- Can we expand on this pattern for higher dimensions? Representing a three-dimensional list as a one-dimensional list?

Question 7: Character grid

You are tasked with writing a program that will construct a `N` by `M` grid with a default character. These variables will be passed as command line arguments to the program.

Example:

```
python char_grid.py 5 5 C
```

You will only want to construct a simple grid with a single character. It is not enough to just print the character in a grid, your program must be able modify individual cells in the grid with a new character.

As a hint: Store the characters in a list or 2d list, create a function that will allow x, y coordinates to be specified.

Example output (unmodified, character used is `O`)

```
OOOOO
OOOOO
OOOOO
OOOOO
OOOOO
```

Example output (modified, character used is `O`)

```
OHOOO
OOOOG
OOOOO
OOOOO
OOOO
```

Question 8: Bouncy String

Using a while loop, you will write a program that will take a command line argument, starting position and index and output the characters in that sequence. During the iteration the index is at the last element it should start iterating in reverse until it reaches the first element.

Example:

```
python bouncy.py quokkas 3 6
kkasak
```

Example:

```
python bouncy.py jumbo 4 9
obmujumbo
```