10.Cycles_continuaition

January 20, 2019

1 More cycles

1.1 for and while are interchangeble

```
In [1]: for i in range(10):
            print('*' * i)
```

```
*
**
***
****
*****
******
*******
********
*********
```

```
In [2]: i = 0
        while i < 10:
            i += 1
            print('*' * i)
```

```
*
**
***
****
*****
******
*******
********
*********
**********
```

1.1.1 PRACTICE

Rewrite this cycle in for-form

```
In [3]: i = 0
        nums = []

        while i < 5:
            nums.append(i)
            i += 1
```

```
In [5]: nums = []

        for i in range(5):
            nums.append(i)
```

Rewrite this cycle in while-form

```
In [8]: xs = [-10, -8, -6, -4, -2, 0, 2, 4, 6, 8]
        summa = 0

        for i in range(len(xs)):
            summa += xs[i]
```

Alternative (better) notation

```
In [10]: for i in xs:
             summa += i
```

```
In [14]: i = len(xs) - 1
         summa = 0

         while i >= 0:
             summa += xs[i]
             i -= 1
```

```
In [15]: summa
```

```
Out[15]: -10
```

1.2 Nested cycles

We can make nested cycles

```
In [34]: for i in range(3):
             for j in range(3):
                 # i + j
                 print(f'i = {i}\tj = {j}\tsum = {i + j}')
```

```
i = 0        j = 0        sum = 0
i = 0        j = 1        sum = 1
i = 0        j = 2        sum = 2
i = 1        j = 0        sum = 1
i = 1        j = 1        sum = 2
i = 1        j = 2        sum = 3
i = 2        j = 0        sum = 2
i = 2        j = 1        sum = 3
i = 2        j = 2        sum = 4
```

1.2.1 PRACTICE

Make nested cycle (depth equal to 2 is ok) and do something with values inside it (e.g. multiply them)

```
In [38]: for i in range(3):
             for j in range(3):
                 # i * j
                 print(f'i = {i}\tj = {j}\tprod = {i * j}')
```

```
i = 0        j = 0        prod = 0
i = 0        j = 1        prod = 0
i = 0        j = 2        prod = 0
i = 1        j = 0        prod = 0
i = 1        j = 1        prod = 1
i = 1        j = 2        prod = 2
i = 2        j = 0        prod = 0
i = 2        j = 1        prod = 2
i = 2        j = 2        prod = 4
```

Let's work with matrices. For now we will emulate them via nested lists

```
In [26]: matrix = [[0, 1, 2], [3, 4, 5], [6, 7, 8]]

         # Don't pay attention to this line for now
         print('[', '\n '.join(map(str, matrix)), ']', sep='')
```

```
[[0, 1, 2]
 [3, 4, 5]
 [6, 7, 8]]
```

What is a row in this matrix and a column?

1.3 Nested list slicing

How to get a row of a matrix?

```
In [39]: matrix[1]
```

```
Out[39]: [3, 4, 5]
```

How to get an element of a matrix?

```
In [40]: matrix[1][0]
```

```
Out[40]: 3
```

How to get a column of a matrix? Well slightly harder, cause we don't have ready underlying structure

```
In [44]: # Get 1st column of matrix
         col = []
         for i in range(len(matrix)):
             for j in range(len(matrix[i])):
                 if j == 1:
                     col.append(matrix[i][j])
         col
```

```
Out[44]: [1, 4, 7]
```

```
In [55]: # Better notation
         # Don't worry now
         col = [matrix[i][j] for i in range(len(matrix)) for j in range(len(matrix[i])) if j ==
         col
```

```
Out[55]: [1, 4, 7]
```

1.3.1 PRACTICE

Print each element of matrix

```
In [56]: for row in matrix:
             for element in row:
                 print(element)
```

```
0
1
2
3
4
5
6
7
8
```

Fill 3x3 matrix with numbers from 0 to 9

```
In [57]: matrix = [[0 for i in range(3)] for j in range(3)]

         # Don't pay attention to this line for now
         print('[', '\n '.join(map(str, matrix)), ']', sep='')
```

```
[[0, 0, 0]
 [0, 0, 0]
 [0, 0, 0]]
```

```
In [58]: height = len(matrix)
         width = len(matrix[0])
         value = 0

         for i in range(height):
             for j in range(width):
                 matrix[i][j] = value
                 value += 1

         matrix
```

```
Out[58]: [[0, 1, 2], [3, 4, 5], [6, 7, 8]]
```