

## 1 while

Here's some code that prints the prime factors of the integer `n` from largest to smallest. Notice that the `while` expression evaluates boolean expression; if the expressions true, it executes the body and then repeats.

```
def factors(n):
    i = n
    while (i > 0):
        if (n % i == 0):
            print(i)
            i = i - 1
```

```
$ python3 factors.py 99
99
33
11
9
3
1
```

```
$ python3 factors.py 75
75
25
15
5
3
1
```

```
$ python3 factors.py 645367
645367
1
```

## 2 for and range

The keyword `for` is used to iterate over sequences of objects. We actually seen one type of sequence already—strings.

```
for i in "this sentence is false"  
    print(i,end=" ")  
print()
```

Notice here we make use of the optional keyword argument to `print end`, which says how the string should be terminated (by default it ends with a newline

`n`). The function `print()` just prints a newline character.

The `for` keyword is often used to iterate over ranges of integers. One can create ranges of integers using the `range` type. Here is some code that

```
def print_matrix(n):  
    for i in range(n):  
        print(str(i) + ":\t", end="")  
        for j in range(n):  
            print(j, end="\t")  
        print()
```

```
$ python3 matrix.py 3  
0: 0 1 2  
1: 0 1 2  
2: 0 1 2
```

```
$ python3 matrix.py 4  
0: 0 1 2 3  
1: 0 1 2 3  
2: 0 1 2 3  
3: 0 1 2 3
```

```
$ python3 matrix.py 5  
0: 0 1 2 3 4  
1: 0 1 2 3 4  
2: 0 1 2 3 4  
3: 0 1 2 3 4  
4: 0 1 2 3 4
```

## Group Work

Write python code to compute the following functions:

- `print_square(n)` : prints a square of dimension `n` where the border is `*` and the interior is `.`
- `is_prime(n)` : returns `True` and only if `n` is prime.
- `gcd(a, b)` : returns the greatest common divisor of `a` and `b`