Assignment week 2

October 6, 2016

1 Assignment week 2

Here we propose a possible implementation of the arithmetic functions - `digits(n, b)`: returns the digits of `n` in base `b` - `is_prime(n)`: returns `True` if `n` is prime and `False` otherwise - `prime_range(n)`: returns the list of primes between 0 and n-1 (inclusive) - `gcd(x,y)`: returns the gcd of `x` and `y` - `factor(n)`: returns the factorization of `n`

The corrections of the problems from the Euler projects will not be given since this worksheet is made public. If you have any question about one of them just send me an e-mail.

```
In [67]: import random
         import numpy as np
```

```
In [ ]:
```

```
In [68]: def digits(n, b=10):
             "Return the digits of n in base b"
             if not isinstance(n, int) or not isinstance(b, int):
                 raise TypeError("input must be integers")
             if n < 0 or b < 1:
                 raise ValueError("n must be non-negative and b positive")
             digits = []
             while n:
                 digits.append(n % b)
                 n //= b
             return digits
```

```
In [69]: digits(133)
```

```
Out[69]: [3, 3, 1]
```

```
In [70]: digits(133, 2)
```

```
Out[70]: [1, 0, 1, 0, 0, 0, 0, 1]
```

```
In [71]: # some intensive random testing of the function digits
         for i in range(100):
             b = random.randint(2, 100)
             n = random.randint(1,10000)
             dig = digits(n, b)
             test = sum(dig[i] * b**i for i in range(len(dig))) == n
             print(test, end=' ')
```

```
True True True True True True True True True True True True True True True True Tru
```

```
In [ ]:
```

```
In [72]: def gcd(x, y):
             "Return the gcd of x and y"
             while y:
                 z = y
                 y = x % y
                 x = z
             return x
```

```
In [74]: gcd(2,3)
```

```
Out[74]: 1
```

```
In [75]: gcd(3, 12)
```

```
Out[75]: 3
```

```
In [76]: # some intensive testing of the function gcd
         for i in range(100):
             a = random.randint(0, 100000)
             b = random.randint(0, 100000)
             m = random.randint(0, 100000)
             test1 = gcd(a,b) == gcd(a,a+b) == gcd(a+b,b)
             test2 = gcd(m*a, m*b) == m*gcd(a,b)
             print(test1, test2, end=' ')
```

```
True True True True True True True True True True True True True True True True Tru
```

```
In [ ]:
```

```
In [77]: def is_prime(n):
             "Checks whether n is prime"
             if not isinstance(n, int):
                 raise TypeError("only accepts integer input")
             if n <= 1:
                 # 0 and 1 are not prime numbers
                 return False
             elif n % 2 == 0:
                 # only 2 is prime among even numbers
                 return n == 2
             else:
                 # trial division in the case of odd numbers
                 i = 3
                 while i*i <= n:
                     if n % i == 0:
                         return False
                     i += 2
                 return True
```

```
In [78]: print([n for n in range(100) if is_prime(n)])

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79


In [ ]:

In [80]: def prime_range(n):
             "Returns the list of prime number smaller than n"
             f = np.ones(n, dtype=bool)
             f[:2] = False
             f[4::2] = False
             i = 3
             while i*i <= n:
                 f[i*i::i] = False
                 i += 2
             return [int(i) for i in f.nonzero()[0]]

In [81]: print(prime_range(100))

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79


In [82]: # check that is_prime and prime_range agree
         [n for n in range(10000) if is_prime(n)] == prime_range(10000)

Out[82]: True

In [ ]:
```