cslsi-04-mueller

November 7, 2018

1 Sheet 03

- Student: Simon Müller
- Mail: s.mueller1995@gmail.com / s6siume2@uni-bonn.de

1.1 Exercise 1

```
In [1]: def calcPascalTriangle(height):
            if height <= 0:
                raise Exception('Height has to be non-negative and non-zero')

            data = []
            data.append([0, 1, 0])

            for h in range(1, height):
                data.append([])
                for w in range(0, len(data[h-1]) - 1):
                    data[h].append(data[h-1][w] + data[h-1][w+1])
                data[h].insert(0, 0)
                data[h].append(0)

            for i in range(len(data)):
                data[i] = list(filter((0).__ne__, data[i]))
            return data

        def formatPascalTriangle(tri):
            maxLen = len(tri) * 5 - 1
            retStr = ''
            for i in range(len(tri)):
                line = ' '.join(format(elm, '4d') for elm in tri[i])
                line = '{:^{x}}'.format(line, x=maxLen)
                retStr += line + '\n'
            return retStr

        def writePascalTriangle(height, path):
            pyramid = calcPascalTriangle(height)
            pyramidStr = formatPascalTriangle(pyramid)
```

```
            print('Writing this pascal triangle to file: ')
            print(pyramidStr)

            with open(path, 'w') as file:
                file.write(pyramidStr)

            print('Writing sucessfull!')
```

```
In [2]: writePascalTriangle(14, 'PascalTriangle-14.txt')
```

```
Writing this pascal triangle to file:
                                    1
                                  1   1
                                1   2   1
                              1   3   3   1
                            1   4   6   4   1
                          1   5  10  10   5   1
                        1   6  15  20  15   6   1
                      1   7  21  35  35  21   7   1
                    1   8  28  56  70  56  28   8   1
                  1   9  36  84 126 126  84  36   9   1
                1  10  45 120 210 252 210 120  45  10   1
              1  11  55 165 330 462 462 330 165  55  11   1
            1  12  66 220 495 792 924 792 495 220  66  12   1
          1  13  78 286 715 1287 1716 1716 1287 715 286  78  13   1

Writing sucessfull!
```

1.2 Exercise 2

```
In [5]: def sieveOfEratosthenes(limit):
            if limit <= 2:
                raise Exception('At least one prime number has to be calculated!')

            numbers = [True for i in range(limit+1)]
            num = 2
            while num * num <= limit(:
                if numbers[num]:
                    for multNum in range(2 * num, limit + 1, num):
                        numbers[multNum] = False
                num += 1

            return numbers

        def trialDivision(num, primes):
```

```
            itPrimes = iter(primes)
            divs = []
            curPrime = next(itPrimes)

            while num > 1:
                if num % curPrime == 0:
                    divs.append(curPrime)
                    num /= curPrime
                else:
                    curPrime = next(itPrimes)
            return divs
```

```
In [8]: %time nums = sieveOfEratosthenes(12345577)
        trials = [4, 1001, 1231, 123259, 12345577] #1234567811, 112233445589, 11223344556607, 19
        primes = [i for i in range(len(nums)) if nums[i] and i > 1]


        for trial in trials:
            divs = trialDivision(trial, primes)
            print("prime factors for {}: {}".format(trial, divs))
```

```
CPU times: user 2.36 s, sys: 40 ms, total: 2.4 s
Wall time: 2.4 s
prime factors for 4: [2, 2]
prime factors for 1001: [7, 11, 13]
prime factors for 1231: [1231]
prime factors for 123259: [123259]
prime factors for 12345577: [12345577]
```

1.3 Exercise 3

see corresponding solution file cslsi-04-mueller-03.py

```
In [9]:
```

```
In [ ]:
```