Programmazione 1 — Tutorato 3

1. Scrivere una funzione che prende in input una stringa e un dizionario. Se il dizionario è vuoto restituisce una stringa criptata e un dizionario che decripta, altrimenti utilizza il dizionario dato in input per decriptare la stringa.

2. Scrivere una funzione che dato n restituisce la matrice di Hilbert di ordine n:

$$H = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \\ \vdots & & & \ddots \end{pmatrix} \quad H(i,j) = \frac{1}{i+j+1} \; i,j = 0,\ldots,n-1$$

3. Scrivere una funzione `MapMat(Matrix, f)` che implementa la "map" su matrici intese come liste di liste. Ad esempio:

$$\left(\begin{bmatrix} 0 & \pi & 2\pi \\ \pi & \frac{3}{2}\pi & \frac{\pi}{4} \\ \pi & 0 & \frac{-1}{4}\pi \end{bmatrix}, sin \right) \longmapsto \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & \sqrt{2}/2 \\ 0 & 0 & -\sqrt{2}/2 \end{bmatrix}$$

4. Scrivere una funzione che data una matrice e un predicato p, restituisce la matrice stessa portando a 0 gli elementi x tali che $p(x) = False$.

5. Ricordando le due funzioni:

```
1  def MakeImage(F, n, scale=0.01):
2      data = [scale*i for i in range(-n,n)]
3      return np.matrix([[F(complex(a, b)) for a in data] for b in data])
```

```
1  def DrawImage(F, n, scale):
2      m = MakeImage(F, n, scale)
3      plt.figure(figsize=(8,8))
4      img = plt.imshow(m, extent=(-scale*n, scale*n, -scale*n, scale*n), cmap='
           hot')
5      plt.show()
```

implementare una funzione che prende in input una funzione $f : \mathbb{C} \to \mathbb{C}$ e restituisce il grafico dell'insieme $J(f) = \{z \in \mathbb{C} : \exists \delta \text{ tale che } |(f^n(z))| < \delta \; \forall n\}$ dove: $f^n = \underbrace{f \circ \ldots \circ f}_{n \text{ volte}}$