PY410 / 505
Computational Physics 1

Salvatore Rappoccio

C++: Class Templates

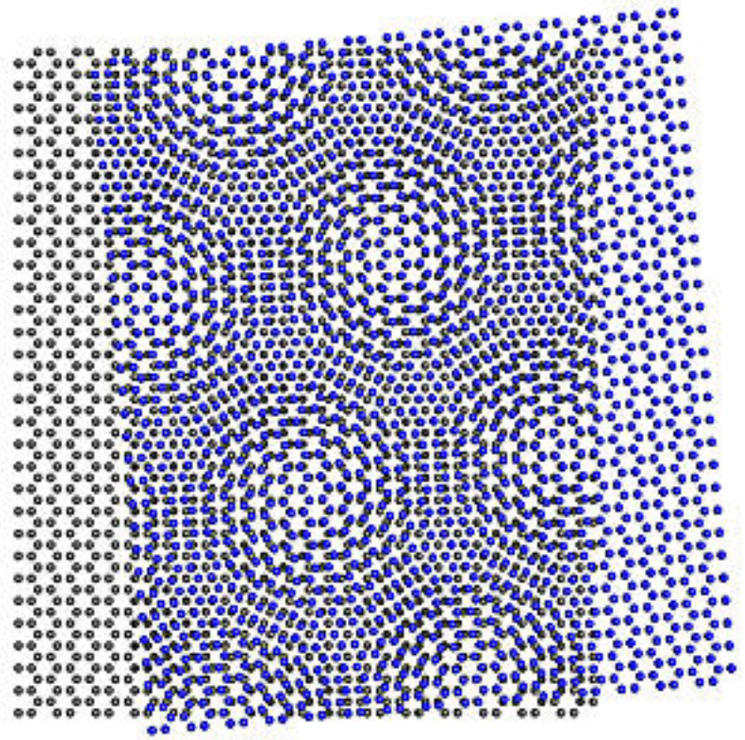
- Similar to function templates, declare with “template <class T>”

- Then the code you write has a PLACEHOLDER value called “T”. “T” is not a class. It is a dummy. It does not exist.

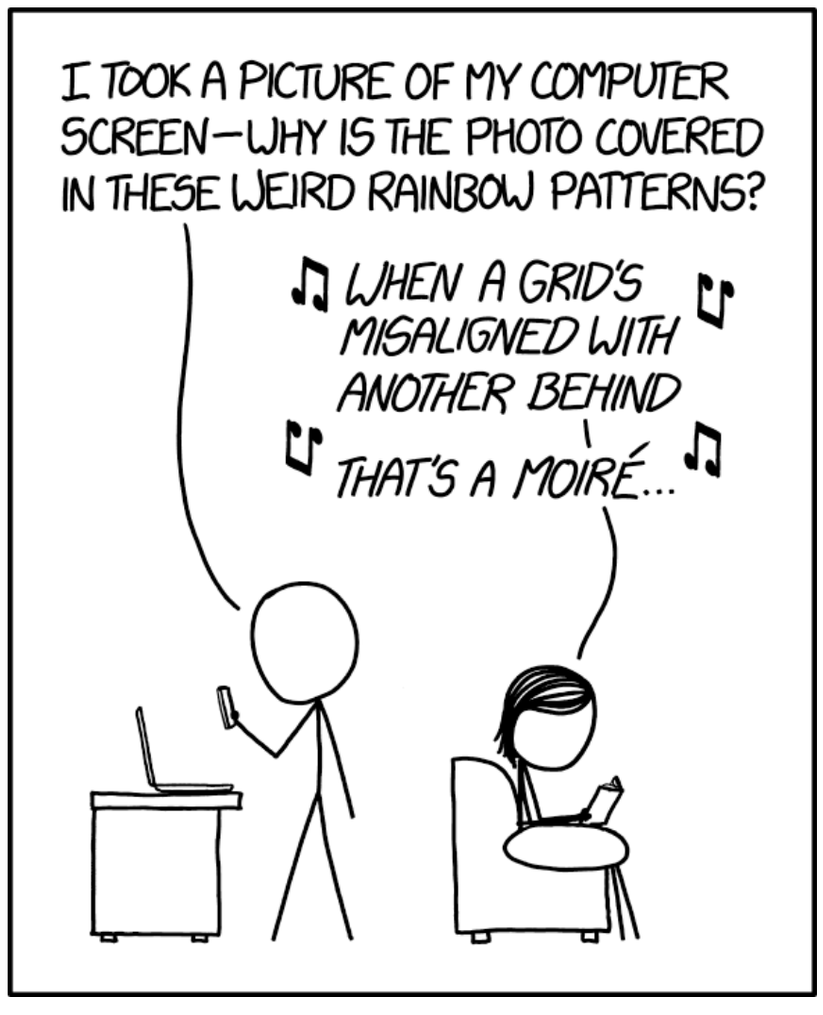
- This defines an INTERFACE to operate on the class object

Moiré

- Colloquium here in 2020 (Andrea Alù) https://arxiv.org/abs/2001.03304
- Moiré Pattern in bilayers (eg Graphene, TMDs)
- Can lead to superconductivity



http://prb.aps.org/abstract/PRB/v81/i12/e125427



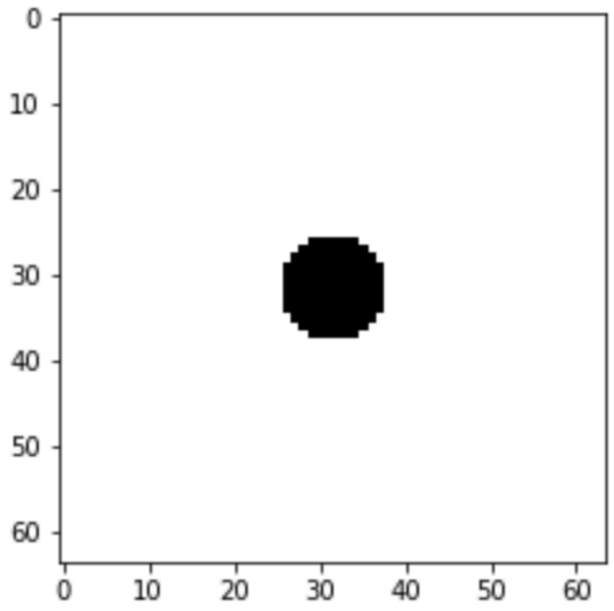
2D Fourier Transform

- In optics every lens does a 2D Fourier transform
- What is the Fourier transform of a pinhole?

```
N = 64
x = np.linspace(-10,10,N)
y = np.linspace(-10,10,N)
```

```
R = 2
xx, yy = np.meshgrid(x, y, sparse=True)
z = (np.sqrt(xx**2 + yy**2) >= R)*1
plt.imshow(z, "gray")
```

```
<matplotlib.image.AxesImage at 0xae2a4a90
>
```

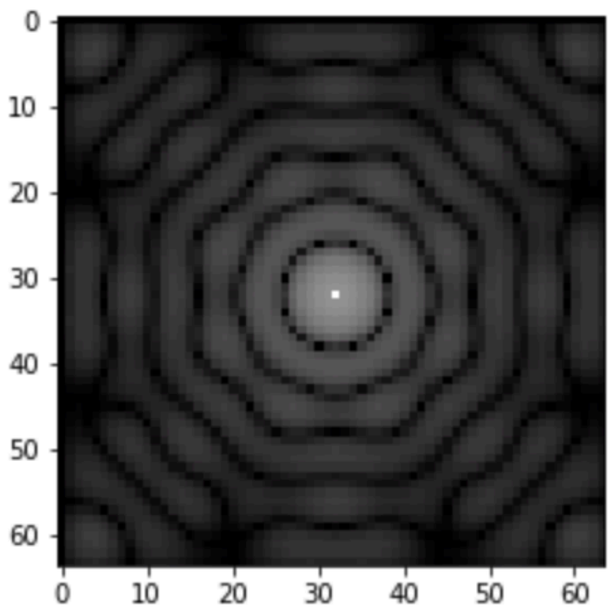


2D Fourier Transform

- Numpy 2D FFT
- Remember to use 2^N pixel
- Shift result to center for convenience
- Plot in log scale
- Remember: Delta Function in 1D is sine in Fourier space

```
Fz = np.fft.fft2(z)
Fzcenter = np.fft.fftshift(Fz)
plt.imshow(np.log(1+np.abs(Fzcenter)), "gray")
```

```
<matplotlib.image.AxesImage at 0xad9da640>
```

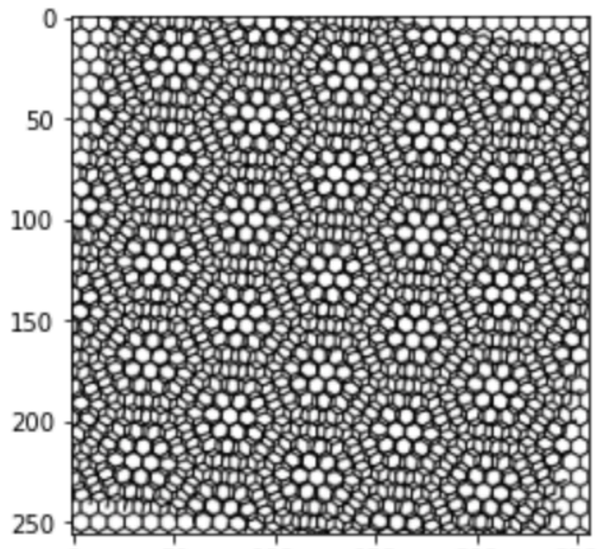


Bilayer Moiré

- Hexagonal lattice
- Bilayer with one layer rotated by angle theta

```
bilayer = np.array(ImageOps.grayscale(Image.open('lattice.png')))
plt.imshow(bilayer, "gray")
```

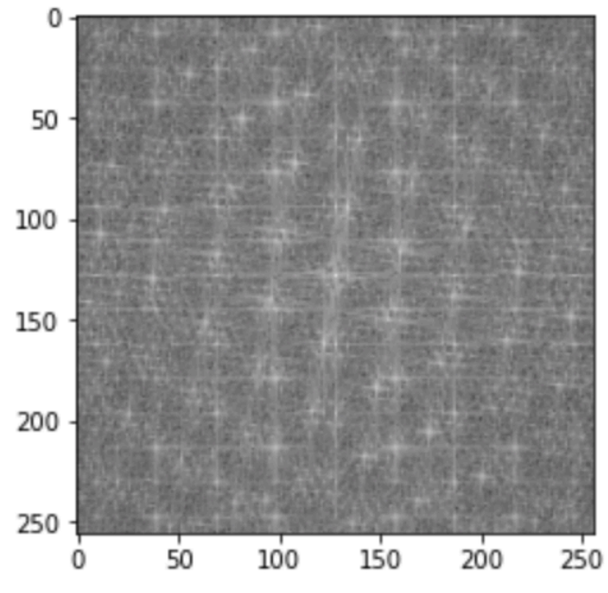
```
<matplotlib.image.AxesImage at 0xaee21490>
```



- FFT shows peaks of periodic structures
- Let’s mess around with it

```
Fbilayer = np.fft.fft2(bilayer)
Fbilayercenter = np.fft.fftshift(Fbilayer)
plt.imshow(np.log(1+np.abs(Fbilayercenter)), "gray")
```

```
<matplotlib.image.AxesImage at 0xaee4c418>
```

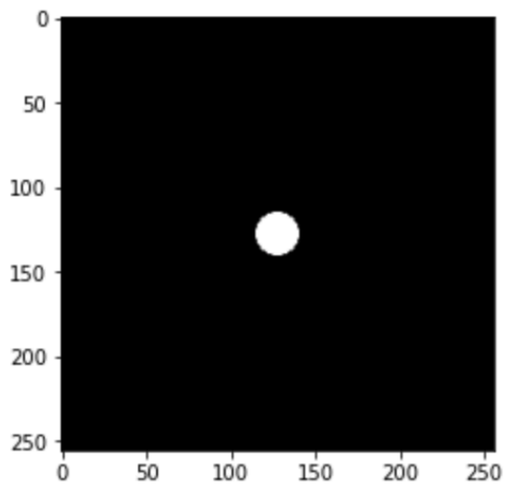


FFT filter

- Low pass filter

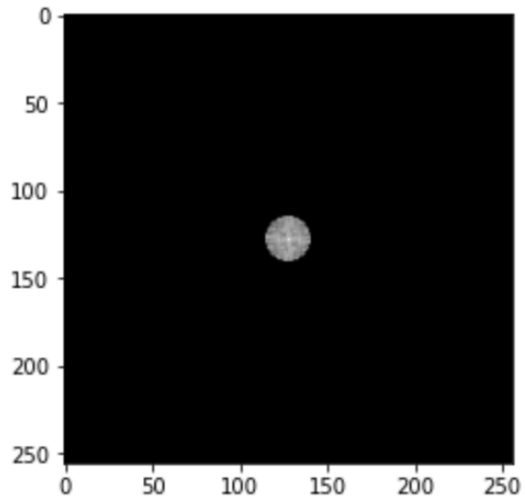
```
R = 1
xx, yy = np.meshgrid(x, y, sparse=True)
lowpass = (np.sqrt(xx**2 + yy**2) <= R)*1
plt.imshow(lowpass, "gray")
```

```
<matplotlib.image.AxesImage at 0xacb8d838>
```




```
plt.imshow(np.log(1+np.abs(lowpass*Fbilayercenter)), "gray")
```

```
<matplotlib.image.AxesImage at 0xac953d48>
```



- Take the inverse FFT

```
filtered = np.fft.ifft2(lowpass*Fbilayercenter)
```

```
plt.imshow(np.log(1+np.abs(filtered)), "gray")
```

```
<matplotlib.image.AxesImage at 0xacb6b580>
```