Sound

Pure Tone(Sine waves로 구성, "Sinusoidal")의 합 -> Complex Tone

Sinusoidal(sine wave)을 만드는 것 -> *Phasor*

e.g. sin, cos

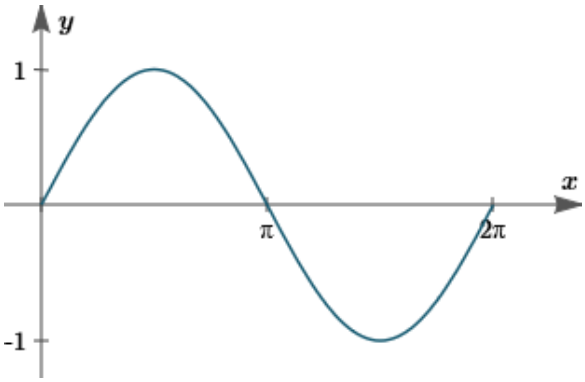
그러면 sin, cos () 괄호 안 입력값 어떻게?

0 ~~~~숫자 증가 ~~~ 2π (= 6.28...) -> radiant

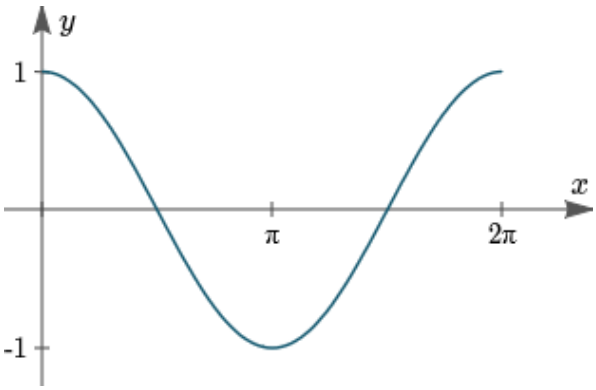
0도 180도 360도 -> degree

➔ Sin(), cos()의 **입력값**으로는 degree X, radiant O

y = sinx



y = cos x



현재의 함수 모양이 2π 간격으로 반복됨 (e.g. 0~100π 이면 50번 반복)

sin 0 = 1, sin π/2 = 1, sin π = 0, sin 3π/2 = -1, sin 2π = 0

cos 0 = 1, cos π/2 = 0, cos π = -1, cos 3π/2 = 0, cos 2π = 1

오일러 공식

실수 x에 대해 다음이 성립한다.

$$e^{ix} = \cos x + i \sin x$$

여기서 e = 2.71(자연로그의 밑)

i = imaginary(허수) [<-> real(실수)]

$$i = \sqrt{-1}$$

$$i^2 = \left(\sqrt{-1}\right)^2 = -1.$$

유일한 변수 -> θ (theta)

모든 수를 포괄할 수 있는 개념 -> 복소수 (a + bi/ a = 0 일 때 허수, b = 0일 때 실수)

f(θ) = 오일러 공식 일 때

θ = 0 일 때

f(θ) = e의 0승이므로 = 1

θ = π/2 일 때

f(θ) = cos(θ) + sin(θ)i = cos(π/2) + sin(π/2)i = 0+1*i = i

θ = π 일 때

f(θ) = -1

θ = 3/2π일 때

f(θ) = 1

그 이후 계속해서 반복...

오일러의 공식을 복소수 평면(complex plain)에다가 그려놓으면

a+bi, (a,b)

Phasor

(phasor를 만들기 전 필요한 라이브러리)

from matplotlib import pyplot as plt (=import matplotlib.pyplot as plt)

➔ matplotlib-> Ploting(그래프화) 할 때 쓰는 라이브러리 (2차원 기준)

from mpl_toolkits.mplot3d import Axes3D

➔ 3D ploting / 회전하는 모양의 3D graph 필요

amp = 1 / (amplitude, 진폭 -> 1 ~ -1 왔다갔다)

sr = 10000 / (sampling rate, Hz

-> 1초에 몇 개의 숫자가 포함되어 있는가? 얼마나 고해상도, 고음질인가?)

dur = 0.5 / (duration, 몇 초 동안 소리가 나는가?)

freq = 100.0 / (frequency, Hz -> 1초에 sin함수가 몇 개 있는가?)

sin함수 그리는 방법

1) 0~2π 까지 모든 값 넣어서 ->
2) sin 함수에다가 집어넣으면
3) sin 곡선이 나타남

1)

theta = np.arange(0, 2*np.pi [, 0.1])

➔ 여기서 0부터 2 π(6.28....) 정도의 값을 arange-> index화 시켜라!

➔ [, 0.1] 부분은 0.1 간격으로 index화 시켜라

e.g. array([0., 0.1, 0.2,]

Out: array([0., 1., 2., 3., 4., 5., 6.,])

2)

s = np.sin(theta)

s

out: array([0. , 0.84147098, 0.90929743, 0.14112001, -0.7568025 ,

-0.95892427, -0.2794155])

-> 이걸 plot를 하게 되면 sin 곡선이 나타나게 됨

3)

fig = plt.figure() / plot 라이브러리 안에 있는 figure 함수 -> figure 형성

ax = fig.add_subplot(111) / figure 안에 있는 add_subplot (subplot을 add해라) 함수

-> ax 변수 형성

Figure => 화면 전체

subplot => 화면을 여러 개 분리해서, 각각의 plot을 만들기

e.g. (221) -> 총 2 by 2로 나누되, (줄 2개, 열 2개) 그 중 1번째 pick

ax.plot(theta, s, '.') / 실제 ploting 을 하려는, x, y값을 적는다

-> (theta = **x값**, s = sin함수의 output = **y값**, **'.'** (**점을 찍어라**))

ax.set_xlabel('theta in radians') -> **x축의 이름**

ax.set_ylabel('value') -> **Y축의 이름**

But! 여기에는 **시간 개념이 x**

음성 -> 시간의 개념이 없으면 소리가 나올 수 없다/ 반드시 시간이 필요

Sin 함수[**진폭(amplitude)** + **주파수(frequency)**] + **초개념(시간)** => 소리

저 sin 함수 하나가 generate 된 것이 1초인지, 1년인지 모르는 상황

만약 이게 1초라면? 이라는 전제를 넣어야 된다

그래서 time을 만들면

t = np.arange(1, sr*(1초인 경우)dur+1) / sr

총 1초라면 몇 개의 time tick을 만들어야 하나? = sr와 일치

Sampling rate와 똑같은 개수의 time tick / 여기서는 1000만

arange로 index를로 time tick를 설명해놓고, 그 다음에 시간: (/sr)로 나눠주면 된다

0~1초까지 / duration의 0.5초

theta = t * 2*np.pi * freq

theta는 시간(0~1초) x 2π (-> 한바퀴 도는 걸 만들어라) x freq (그걸 몇 바퀴 돌릴까?)

s = np.sin(theta)

현재는 theta값, s값, time까지 다 있는 상태 (과거의 sin 함수는 theta값, s값 2개였는데)

x축에 time(theta는 안 쓰는 이유, 너무 빨리 돌아가므로)

y축

fig = plt.figure()

ax = fig.add_subplot(111)

ax.plot(t[0:1000], s[0:1000],'.')

/ -> 0부터 1000개까지 (time ticks) (t,s) -> 2차원 벡터 둘의 숫자 일치

ax.set_xlabel('time (s)')

ax.set_ylabel('real')

오일러 공식 이용

c [=complex] = np.exp(theta*1j)

exponention 함수 / 1j = i 와 같은거

c = np.exp(theta*1j)

c

array([0.99802673+6.27905195e-02j, 0.9921147 +1.25333234e-01j,

0.98228725+1.87381315e-01j, ..., 0.9921147 -1.25333234e-01j,

0.99802673-6.27905195e-02j, 1. +1.96438672e-15j])

-> a +bi 형태로 구성되어 있음 (복소수의 벡터가 쭉 존재)

e-01 (=10의 -1)

e-02 (=10의 -2) -> 소수점을 어디로 옮기는가?

쓰는 숫자와 정보의 양이 똑같아짐 / **아무리 큰 수든, 작은 수든 똑같은 정보량으로 적어놓는다**

fig = plt.figure()

ax = fig.add_subplot(111, projection = '3d') / 3d의 형태로 만들어라

ax.plot(t[0:1000], c.real[0:1000], c.imag[0:1000], '.') / 여기에 나오는 점이 x,y,z의 좌표를 지님 ->

/1000개의 점이 생김

ax.set_xlabel('time (s)')

ax.set_ylabel('real')

ax.set_zlabel('imag')

a+bi -> a,b를 따로 받을 수도 있음 <- complex number에서 이 2 부분을 분할할 수 있음

t[0:1000] -> time 값

c.real[0:1000] / a값 -> 실수값

c.imag[0:1000] / b값 -> 허수값

real 만 본다 (**위**에 올라가서 보겠다) -> **cos 함수 모양**

imag 만 본다 (**옆**에서 보겠다) -> **sin 함수 모양**