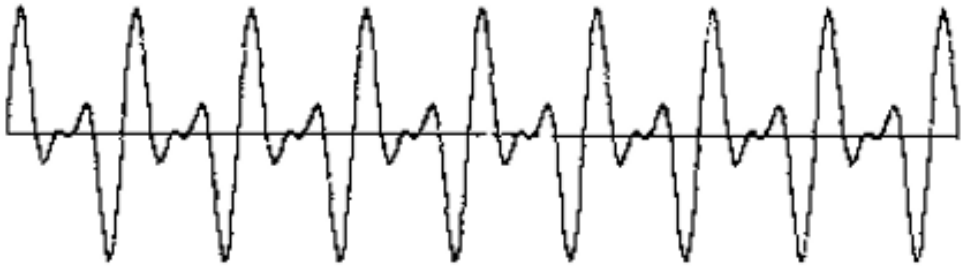
多媒體程式設計
音訊資料處理

Instructor: 馬豪尚

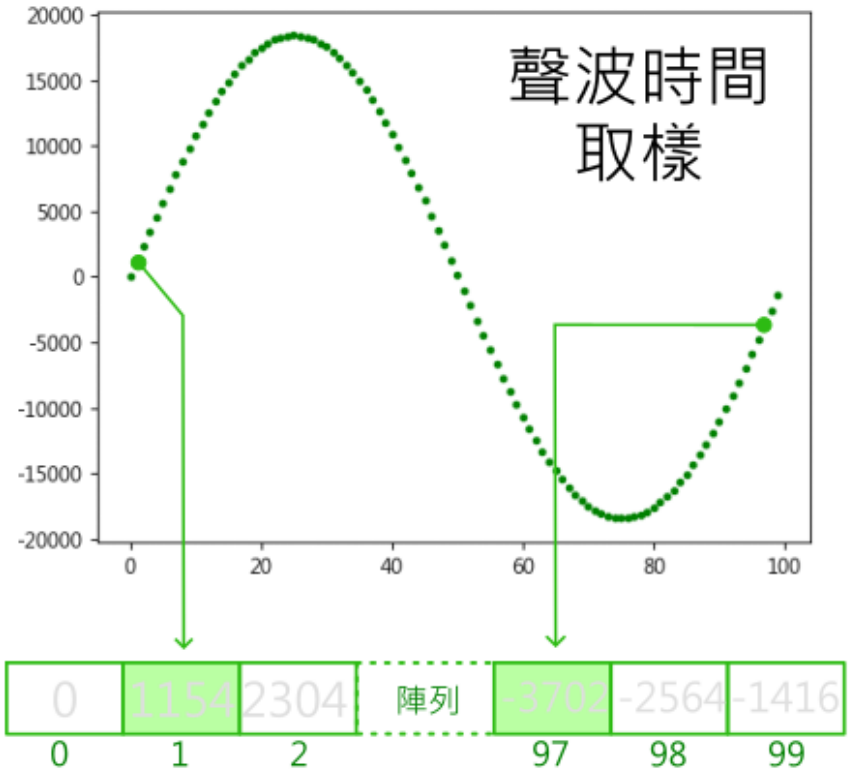
音訊的表示和儲存

- 真實世界中的聲波是連續的類比訊號
- 音訊在電腦中是如何表示和儲存?
  - 在電腦當中，是無法表示真實世界的連續性質的
  - 將聲波數位化，變成一個個離散的數位訊號



音訊數位化 - 取樣

› 對聲音訊號做「取樣」的動作，取樣把時間變成離散，取樣的時間點愈密集，音訊的品質就會愈高



取樣率 sample rate

1秒內取多少點

取樣週期 sample period

隔多久取1點

CD音質取樣率44100

每1秒取44100點

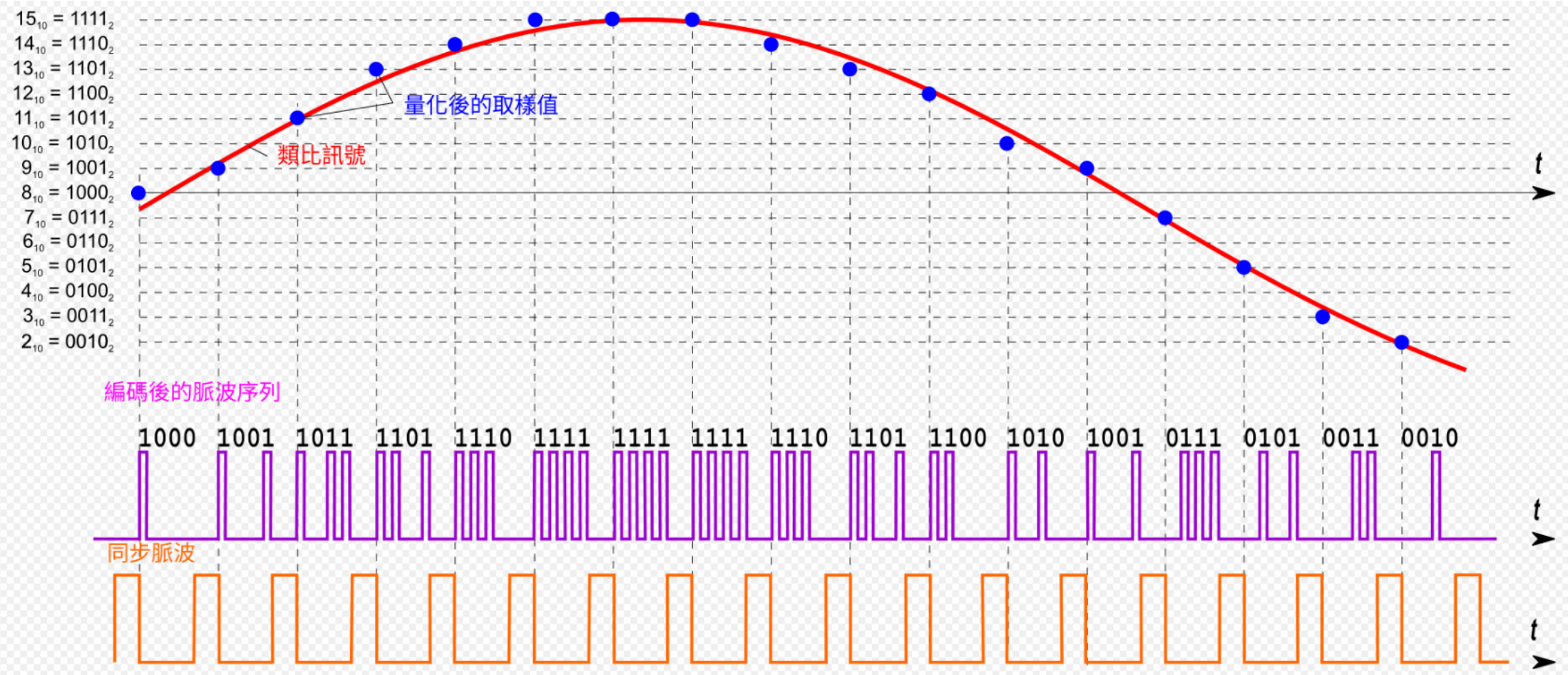
每1/44100秒取1點
(約0.00002秒)

音訊數位化 -量化

- 量化(quantization)是把振幅類比值變成離散值
- 音訊位元深度
  - 除了每秒鐘取幾個點之外，每個點要用多少容量來表示
  - 16bits、24bits、48bits甚至64bits

音訊數位化

› 整數量化-圖上的音訊位元深度以4bits來表示



音訊數位化

- 音訊解析度
  - 包含音訊取樣率和音訊位元深度(audio bit depth)
  - 整數量化16位WAV文件可以存儲從0 dBFS到-96 dBFS的音頻

位元深度	訊噪比	整數取值總數（單個取樣點）	有符十進位取值範圍
4	24.08 dB	16	−8至+7
8	48.16 dB	256	−128至+127
16	96.33 dB	65,536	−32,768至+32,767
24	144.49 dB	16,777,216	−8,388,608至+8,388,607
32	192.66 dB	4,294,967,296	−2,147,483,648至 +2,147,483,647
48	288.99 dB	281,474,976,710,656	−140,737,488,355,328至 +140,737,488,355,327
64	385.32 dB	18,446,744,073,709,551,616	−9,223,372,036,854,775,808至 +9,223,372,036,854,775,807

音訊數位化

- 浮點數量化
  - 優勢
    - 可以使得相鄰的值之間差距更小，達到更精確地表示音訊
  - 劣勢
    - 對於相同位元深度，在取值很大的情形下，相鄰浮點數比相鄰整數間隔更大
    - 需要更多的位元數來表示

音訊檔案格式

- 未壓縮格式，代表的是儲存未壓縮的音頻樣本
  - WAV、PCM、AIFF、DSD
- 壓縮無失真格式，這類型的檔案可解壓縮回原始大小，繼而完整保留音質
  - FLAC、APE、ALAC、WavPack
- 有損壓縮格式，這類型格式會於傳輸過程中損失資料，進而無法解壓縮完全還原回原始音訊
  - MP3、MP4、 AAC、Ogg Vorbis

音訊檔案儲存 - WAV

› WAV檔遵守資源交換檔案格式之規則
  – 在檔案的前44(或46)位元組放置檔頭(header)，使播放器或編輯器能夠簡單掌握檔案的基本資訊

0	區塊編號	4	大	“RIFF”
4	總區塊大小	4	小	=N+36
8	檔案格式	4	大	“WAVE”
12	子區塊 1 標籤	4	大	“fmt “
16	子區塊 1 大小	4	小	16
20	音訊格式	2	小	1(PCM)
22	聲道數量	2	小	1(單聲道) 2(立體聲)
24	取樣頻率	4	小	取樣點/秒 (Hz)
28	位元(組)率	4	小	=取樣頻率*位元深度/8
32	區塊對齊	2	小	4
34	位元深度	2	小	取樣位元深度
36	子區塊 2 標籤	4	大	“data”
40	子區塊 2 大小	4	小	N (=位元(組)*秒數*聲道數量)
44	資料	=N	小	<音訊資料由此開始>

音訊特徵

- 在一個特定音框內，我們可以觀察到的三個主要聲音特徵可說明如下：
  - 音量（Volume）：代表聲音的大小，可由聲音訊號的震幅來類比，又稱為能量（Energy）或強度（Intensity）等。
  - 音高（Pitch）：代表聲音的高低，可由基本頻率（Fundamental Frequency）來類比，這是基本週期（Fundamental Period）的倒數。
  - 音色（Timbre）：代表聲音的內容（例如英文的母音），可由每一個波形在一個基本週期的變化來類比。

Python讀取音訊檔案

- 基本開檔讀檔Read
  - 無視音訊格式，讀進python為bytes的形式
- WAVE模組
  - Python 內建讀取寫入wav格式音訊的模組
- pydub套件
  - 提供簡單的音訊操作套件，可支援各種音訊格式
- Scipy
  - 數據分析常用套件，可以對數位訊號做處理和提供分析

WAVE模組讀取音訊檔案

- 載入模組
  - Import wave
- wave.open(file, mode=None)
  - file → wav音訊檔案
  - mode → ‘rb’ 讀取模式, ‘wb’ 寫入模式
- 宣告讀取物件
  - wavobj = wave.open(filename, "rb")

WAVE模組讀取音訊檔案

- 取得wav檔案格式資訊參數
  - params = wavobj.getparams()
  - 返回一個tuple(nchannels, sampwidth, framerate, nframes, comptype, compname)
- 依照wav格式標頭檔定義
  - nchannels → 聲道
  - sampwidth → 取樣長度(位元深度/8)
  - framerate → 取樣率
  - nframes → 音訊總取樣數(長度)
  - comptype → 返回壓縮類型（只有 'NONE' 類型）
  - compname → 跟壓縮類型一樣，用'not compressed' 代替 'NONE'

WAVE模組讀取音訊檔案

- 取得音訊資料
  - wavobj.readframes(n)
    - 返回以bytes型態字串表示的n個取樣點資料
- 轉成np.array
  - np.fromstring(bytesobj, dtype="int16")
  - np.frombuffer(bytesobj, dtype ="int16")

WAVE模組寫入音訊檔案

- 宣告寫入物件
  - wavobj = wave.open(filename, “wb")
- 設定聲道數、sample 大小(byte) 和取樣頻率
  - wavobj.setnchannels(nchannels)
  - wavobj.setsampwidth(sampwidth)
  - wavobj.setframerate(framerate)
- 寫入聲音資訊
  - wavobj.writeframes(wave_data.tostring())

Pydub套件讀取音訊

- 安裝套件
  - pip install pydub
- 載入模組
  - from pydub import AudioSegment
- 支援不同的音訊格式
  - audobj = AudioSegment.from_wav(file) → 讀取 .wav
  - audobj = AudioSegment.from_mp3(file) → 讀取 .mp3
  - audobj = AudioSegment.from_flv(file) → 讀取 .flv
  - audobj = AudioSegment.from_ogg(file) → 讀取 .ogg

Pydub套件讀取音訊

- AudioSegment.from_file(file, format='格式')
  - 可以從影片讀音軌檔案
  - 也可以直接讀取音訊檔案

方法	說明
AudioSegment.from_file("file.mp4", "mp4")	讀取 .mp4
AudioSegment.from_file("file.wma", "wma")	讀取 .wma
AudioSegment.from_file("file.aiff", "aiff")	讀取 .aiff

Pydub套件取得音訊資訊

- AudioSegment的物件裡有定義了各種"屬性"可以存取音訊物件的資訊
  - channels → 聲道數量
  - duration_seconds → 聲音長度，單位是秒
  - frame_rate → 取樣頻率
  - raw_data → 聲音資料
  - dBFS → 聲音響度

Pydub套件輸出音訊檔案

- audobj.export(filename, 參數='設定值')
  - export是AudioSegment物件下的方法
  - filename是要輸出的檔名

參數	設定值說明
format	輸出格式，預設 mp3，可設定 wav 或 raw
codec	編碼器，預設自動判斷
bitrate	壓縮比率，預設 128k，可設定 32k、96k、128k、192k、256k、320k
tags	夾帶在聲音中的標籤，使用字典格式
cover	夾帶在聲音中的預覽圖，支援 jpg、png、bmp 或 tiff 格式

Pydub套件切割音訊

- AudioSegment的物件可以直接對它做像陣列的操作
  - 讀入音訊時自動轉成以毫秒為單位存入
- 切割音訊
  - audobj[begin:end]

AudioSegment物件

Pydub套件操作音訊

- 將兩段音訊串接
  - audobj1+ audobj2
- 將一段音訊重複幾次並串接在一起
  - audobj*次數
- 將audobj這個物件的音訊反轉過來
  - audobj.reverse()

Pydub套件操作音訊

- 調整音訊的音量
  - 在音訊處理中調整音量就是調整振幅
- 將所有AudioSegment陣列中的資料增加或減少
  - 增加表示將音量變大
    - audobj+一個數值
  - 減少表示將音量變小
    - audobj-一個數值
- 使用audobj.apply_gain(數值)
  - 數值為正就是將音量變大，負的就是變小

Pydub套件操作音訊

- 調整音訊的開頭音量淡入
  - audobj.fade_in(毫秒數)
- 調整音訊的結尾音量淡出
  - audobj.fade_out(毫秒數)
- fade(to_gain=數值, start=毫秒數, duration=毫秒數)
  - 提供可指定時間的調整方式
  - to_gain表示要淡入或淡出的音量大小，淡入為正數，淡出為負數
  - start表示開始的時間點，單位為毫秒
  - duration表示持續的時間，單位為毫秒

Pydub套件混合音訊

- sound1.overlay(sound2, position, gain_during_overlay, loop, times)
  - 要用主要音訊來呼叫overlay函數來混合次要音訊
  - 總共5個參數
    - 第一個參數sound2為要混合的次要音訊
    - 第二個參數指定從主要音訊的哪個時間點開始混合，單位為毫秒
    - 第三個參數為一個數值，在混合時主要音訊的音量調整(正數增加或負數減少)
    - 第四個參數為如果次要音訊不夠長時，是否要重複(True/Flase)
    - 第五個參數為一個數值，指定次要音訊要重複幾次

練習

- 將兩段音樂做串接
  - 第一段音樂的奇數秒
  - 第二段音樂的偶數秒
  - 輸出合成後的音訊

第一段音訊的第1秒	第二段音訊的第2秒	第一段音訊的第3秒	第二段音訊的第4秒	第一段音訊的第5秒	第二段音訊的第6秒	第一段音訊的第7秒	第二段音訊的第8秒	第一段音訊的第9秒	…

- 將多段音訊進行合成
  - 將每一段音樂和任意一段音效進行混合
  - 輸出合成後的音訊