Forritunarmálið Python

Day 5

I/O and Files

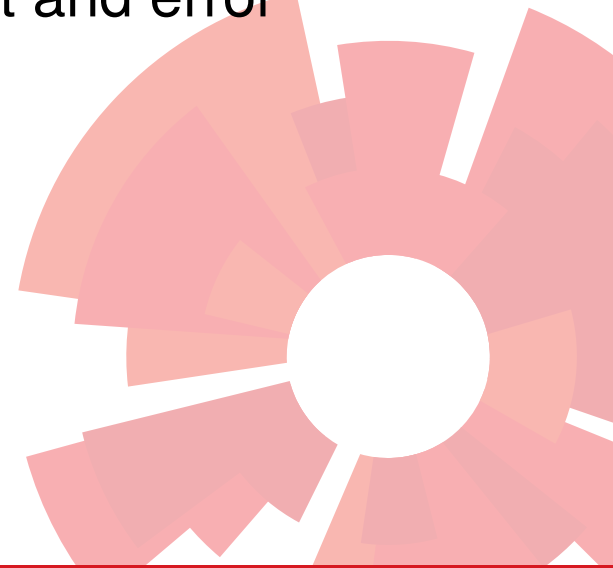
Hjalti Magnússon

1. desember 2017

I/O

Standard input/output

- `input`
  - Read a line from input
- `sys.stdin/stdout/stderr`
  - Work with OS's standard input, output and error



File encodings

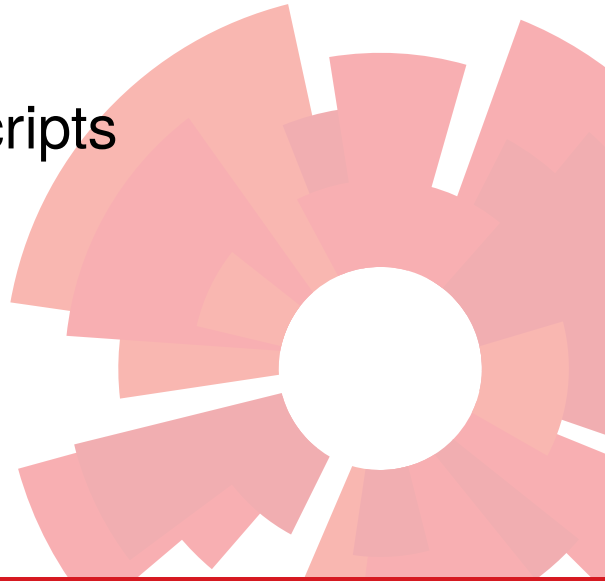
- A files is just a sequence of bits
  - We like to group them into chunks of 8 and call them bytes
- We categorize files into text files and binary files

Encoding

- Text files use the bytes to represent letters
- Originally, ASCII was used
  - Bytes 32–127 were used to represent letters
  - 0–32 were used for special (non-printable) characters
  - the last bit (bytes 128–255) was reserved for error detection
- Technology spread around the world and countries started using the last bit to denote their own letters
  - More than 128 non-English letters around the world
  - Encodings started popping up
  - Chaos ensued

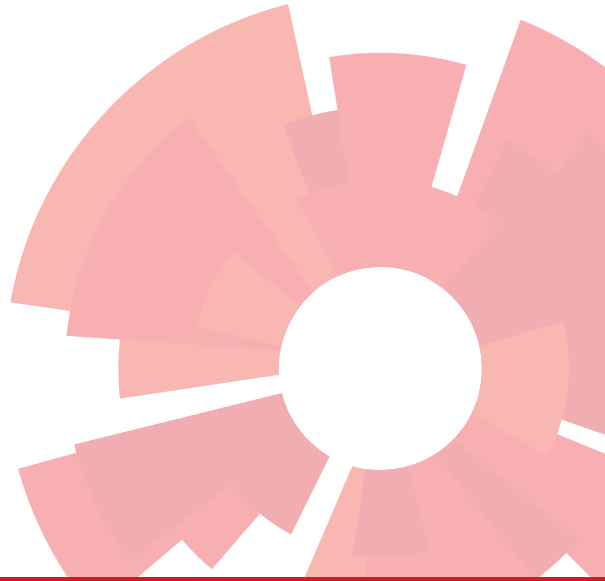
Unicode

- To unite the world in one encoding
- Has a few implementations
  - UTF-8 (most common)
  - UTF-16
  - UTF-32
- Has 136,755 characters
  - Covers modern languages, historic scripts
  - . . . and emojis



UTF-8

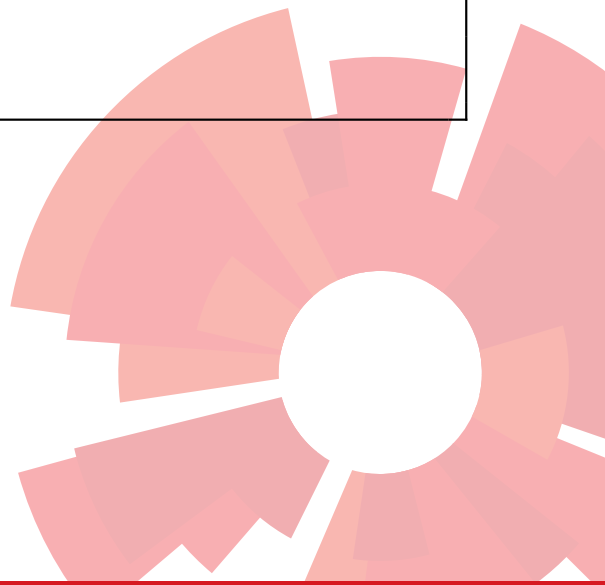
- Variable length encoding
- The first 127 bytes are identical to ASCII



Strings and encoding

- A string is never just a string
  - Behind a string is an encoding
- Python strings are UTF-8 encoded

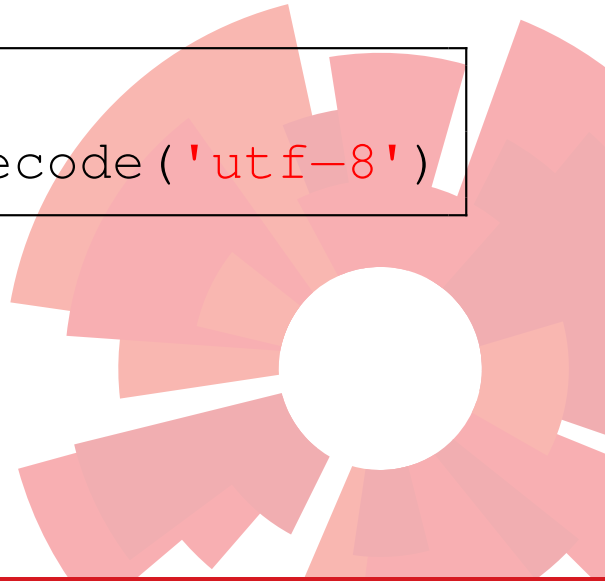
```
>>> 'Thad er fjor'.encode()
b'Thad er fjor'
```



Bytes

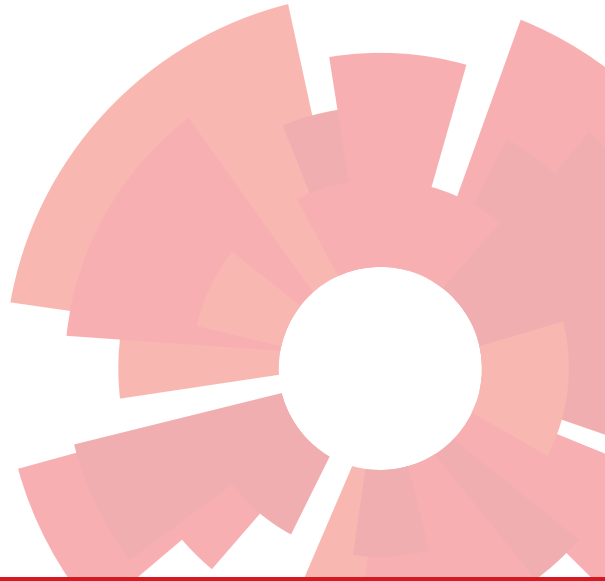
- `bytes` is a read-only sequence of bytes
  - Essentially ASCII strings
- **`bytearray`** is a mutable sequence of bytes
- Provided an encoding, byte sequences can be decoded to UTF-8 strings

```
b'\xF0\x9F\x98\x81'.decode('utf-8')
b'\xc3\x9ea\xc3\xb0 er fj\xc3\xb6r'.decode('utf-8')
```



open

- mode
  - `'r'`, `'w'`, `'rb'`, `'wb'`, `'a'`, `'r+'`
- `read`
  - errors
- streams
  - `read`
  - `write`
  - `append`



A little extra

Meta

- `eval`
- `exec`