PYTHON 2.7 → PYTHON 3.X CHEAT SHEET

PYTHON 2.7 → PYTHON 3.X CHEAT SHEET (for “The Python Standard Library by Example”, 2011)

CORE SYNTAX
- print → function: print(x, y, sep=" ", end="\n")
- Division: / = true division, // = floor division
- Exceptions: except ValueError as e:
- input(): old raw_input(); raw_input removed
- range/map/filter/zip return iterators (wrap with list(...) to materialize)
- next(obj) replaces obj.next()
- Comparisons: no cmp; use key=cmp_to_key for sorting with old-style cmp
- Strings: str is Unicode; bytes is binary data (b"...")
- exec is a function; long type removed; u'' prefix unnecessary

TEXT VS BYTES (THE BIG ONE)
- Decode at edges → process as text (str) → encode at edges
- File I/O: open(path, mode, encoding="utf-8", newline="")
- Network/crypto/compression APIs expect bytes: payload.encode("utf-8")
- Regex: text patterns yield str matches; bytes patterns yield bytes matches

DICT/LIST/ITERATORS
- d.keys()/items()/values() → views (dynamic); wrap with list(...) if needed
- d.has_key() removed → use key in d
- iteritems()/itervalues() → items()/values()
- List comprehensions have their own scope; loop vars don’t leak

RENAMED/STDLIB MODULES
- urlparse → urllib.parse
- urllib, urllib2 → urllib.request, urllib.error, urllib.parse
- httplib → http.client
- BaseHTTPServer, SimpleHTTPServer, CGIHTTPServer → http.server
- Cookie → http.cookies; cookielib → http.cookiejar
- robotparser → urllib.robotparser
- SocketServer → socketserver
- ConfigParser → configparser (lowercase)
- Queue → queue (lowercase)
- StringIO / cStringIO → io.StringIO (text), io.BytesIO (bytes)
- SimpleXMLRPCServer → xmlrpc.server; xmlrpclib → xmlrpc.client
- imp (deprecated) → importlib, importlib.resources, importlib.metadata

FILES & PATHS
- Prefer pathlib: from pathlib import Path
  Path("file.txt").read_text(encoding="utf-8")
  Path("file.txt").write_text(data, encoding="utf-8")
- For CSV: open(..., newline="", encoding="utf-8")

DATES & TIME
- Use datetime, timezone.utc, and zoneinfo (3.9+) for IANA time zones
- fromisoformat()/isoformat() make parsing/formatting easy

CONCURRENCY & PROCESSES
- subprocess.run([...], check=True, text=True, capture_output=True)
- concurrent.futures: ThreadPoolExecutor / ProcessPoolExecutor
- asyncio: async/await for high-concurrency I/O

EMAIL & NETWORK
- Build messages with email.message.EmailMessage; policy=default
- HTTP client: urllib.request for stdlib; requests (third-party) is common

• Simple server: from http.server import SimpleHTTPRequestHandler, HTTPServer

MODERN PYTHON SUPERPOWERS
• f-strings: f"{name=} {value:.2f}"
• dataclasses: from dataclasses import dataclass
• typing: list[str], dict[str, int], TypedDict, Protocol
• enum: from enum import Enum, IntFlag
• statistics: mean, median, quantiles
• lzma module for compression; pathlib everywhere

COMMON GOTCHAS WHEN PORTING
• UnicodeError on I/O → always specify encoding="utf-8"
• CSV double newlines on Windows → pass newline=""
• HTTP/text confusion → .read() gives bytes; decode before JSON/str ops
• Sorting with cmp → use functools.cmp_to_key
• Relative imports in packages → from .submodule import thing

MINI BEFORE/AFTER
• HTTP GET
```
  # Py2: resp = urllib2.urlopen(url).read()
  # Py3:
  from urllib.request import urlopen
  with urlopen(url) as r:
      body_bytes = r.read()
      text = body_bytes.decode("utf-8")
```

• Simple HTTP server
```
  # Py3:
  from http.server import SimpleHTTPRequestHandler, HTTPServer
```

• Queue
```
  import queue
  q = queue.Queue()
```

• Configs
```
  import configparser
```

• Email (modern)
```
  from email.message import EmailMessage
  msg = EmailMessage(); msg["Subject"] = "Hello"; msg.set_content("Hi")
```