Python Secure Code Cheat Sheet – Checklist

Task/Vector	In-secure	Secure	Caveat/Other
Serialization/Deserialization	`pickle(), cPickle(), yaml.load()`	`yaml.SafeLoader`	For trusted inputs, pickle, yaml.load are still fine.
Secrets handling	`random()`	`secrets()`	
Injection	`eval(), exec(), input(), popen(), subprocess(), os.system()`		User input is evil for all of them, validate strictly.
Test code	`assert()`	`If/else`	Assert statements are skipped for compiled code, don't use for logic.
String formatting	`f-strings, str.format()`	`Template class`	Issues arises with unvalidated user input only.
XML Parsing	`sax, etree, minidom, pulldom, xmlrpc`	`defusedxml`	Issues arises with maliciously crafted data only.
Imports typo-squatting	`acquisition, bzip, crypt, setup-tools, urlib3, diango, python3-dateutil, jeIlyfish`	`Acquisition, bz2file, crypto, setuptools, urllib3, Django, dateutil, jellyfish`	Those are just a few examples. Import statements execute the code, so be careful with imports.

Task/Vector	In-secure	Secure	Caveat/Other
Dependency management	`pip install/pip freeze`	`pipenv`	Nothing insecure with pip but for better management, use pipenv to lock dependencies and work in virtual env.
Creating temporary files	`tempfile.mktemp()`	`tempfile.mkstemp()`	More of a race condition issue which could lead to undesired output.