Bootcamp 134 | Python

Course 18 | RegEx

Amir Hossein Chegouniyan

Head of the Technical Team at Dariche Tejarat

Lecturer of Python – Django at Maktab Sharif

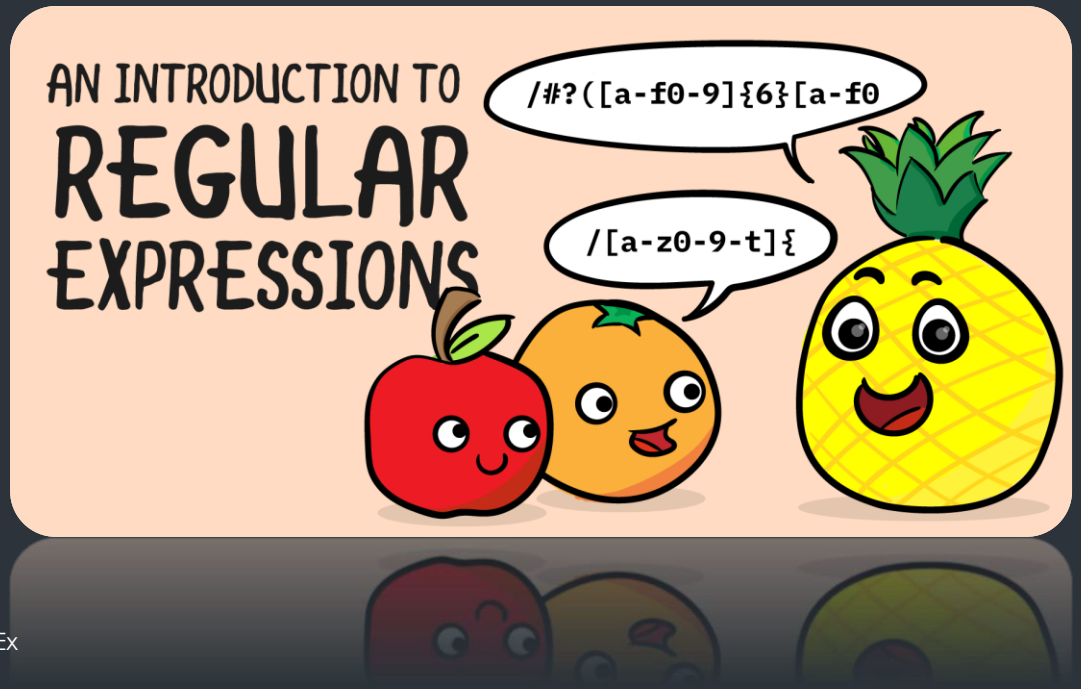
Amirhossein-chegounian

Content

- Introduction to Regex
- Python Regex Library (re)
- Understanding Python Packages
- Working with Third-Party Libraries
- Advanced Function Concepts
- What is Serialization?
- Pickle Module

Introduction to Regex

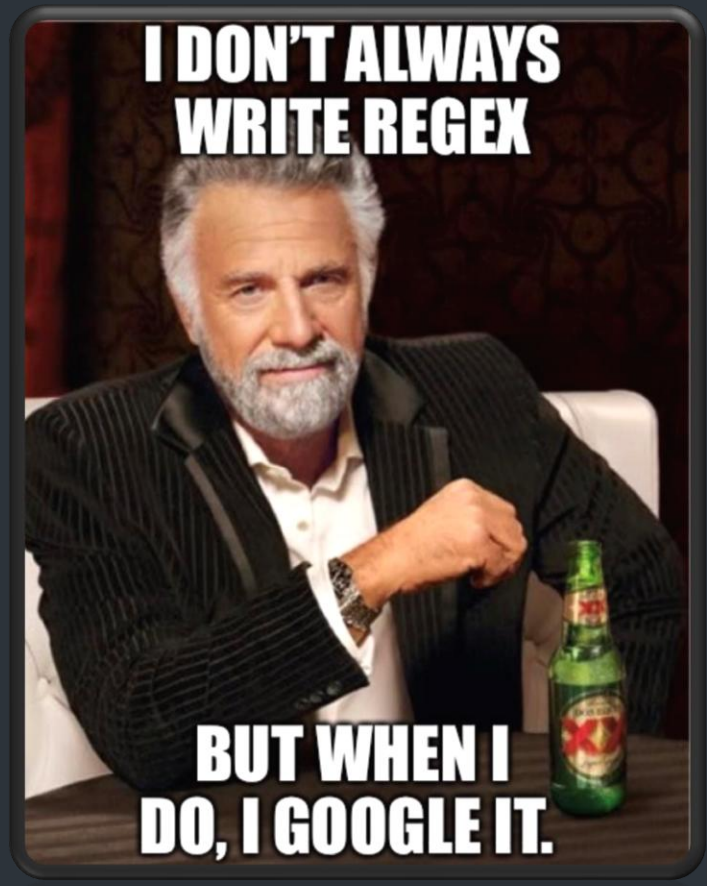
- Definition: A sequence of characters defining a search pattern.
- Applications: Pattern matching, validation, extraction, and substitution.



Introduction to Regex | Regex Syntax Basics

- .: Matches any character except newline.
- ^ and $: Start and end of a string.
- []: Matches any one character within brackets.
- *, +, ?: Quantifiers for repetition.
- |: Alternation (OR operator).
- () Grouping.

Introduction to Regex | For Fun



Python Regex Library (re) | Functions

- import re
- re.findall(pattern, string) # return a list of result
- re.split(pattern, string) # split string by the pattern
- re.match(pattern, string) # return true/false
- re.sub(pattern, replacement, string) # replace a string in other string

Python Regex Library (re) | Flags

- re.IGNORECASE (re.I): Case-insensitive matching.
- re.VERBOSE (re.X): Write readable regex with comments.

Understanding Python Packages

- Definition: A way to organize and distribute reusable code.
- Structure:
  - __init__.py: Marks a directory as a package.
  - Modules (Python files).
  - setup.py: Configuration for package distribution.

Understanding Python Packages | __init__

```
from .math_utils import add
from .string_utils import to_upper

__all__ = ["add", "to_upper"]                    # for when use from mypackage import *
print("mypackage loaded successfully!")          # initializing codes
```

Understanding Python Packages | Setup

```
from setuptools import setup, find_packages

setup(
    name="mypackage",
    version="0.1",
    packages=find_packages(),
    install_requires=[],
)
```

Working with Third-Party Libraries

- Installing Packages: pip install <package_name>.
- Popular Libraries:
  - NumPy: Numerical operations.
  - Pandas: Data manipulation.
  - Requests: HTTP requests.
  - Flask/Django: Web development.

Advanced Function Concepts | Zen of Python

- Principles of writing Pythonic code.

import this

The Zen of Python, by Tim Peters

Beautiful is better than ugly.

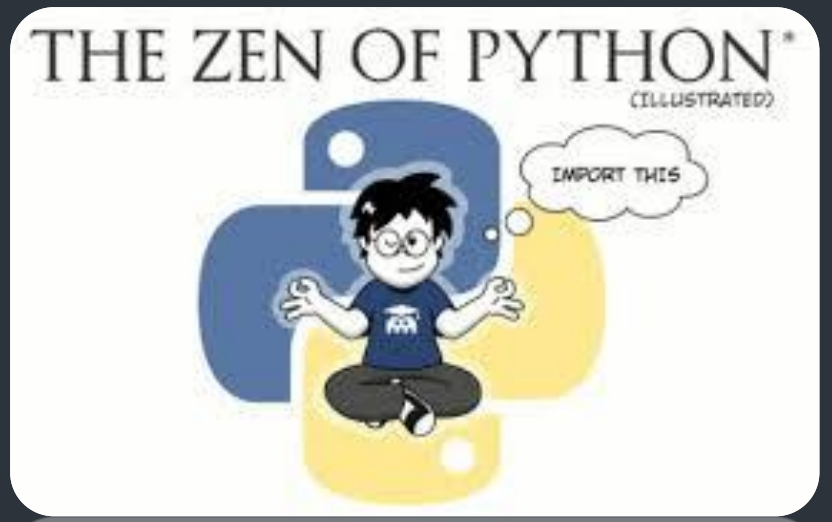
Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

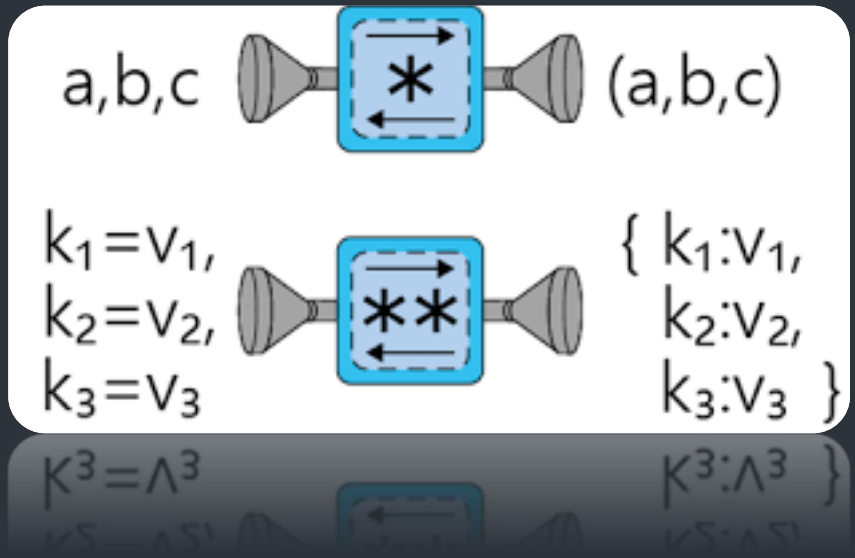


Advanced Function Concepts | PEP 8 - Python Style Guide

- Naming conventions.
- Proper indentation and line breaks.
- Comments and docstrings.

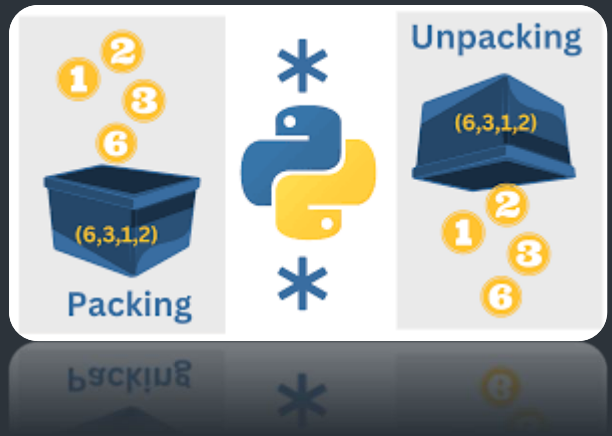
Advanced Function Concepts | Others 1

- *args and **kwargs:
  - Use cases for handling variable-length arguments.
  - Examples of unpacking lists and dictionaries into arguments.



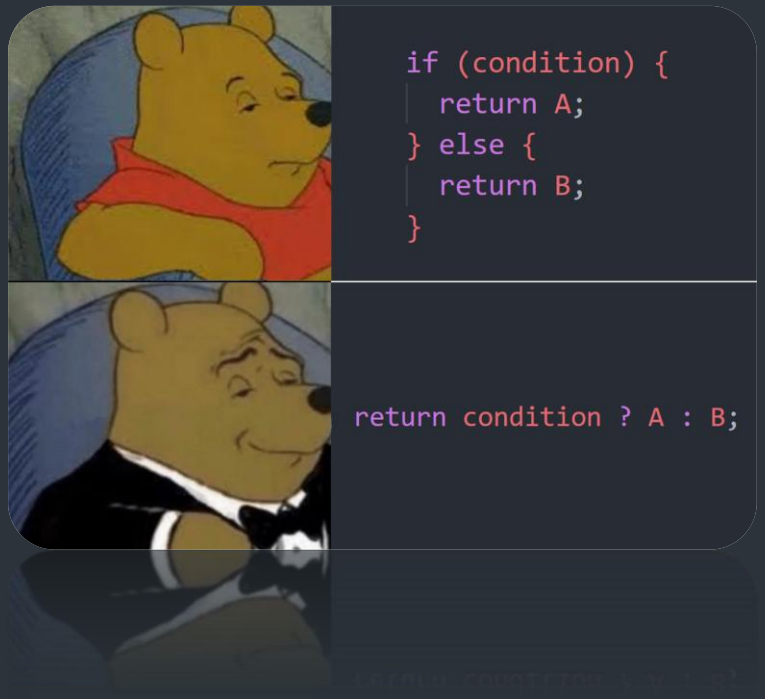
Advanced Function Concepts | Others 2

- Tuple Unpacking:
  - Assign multiple variables in one line.
  - Example: a, b = (1, 2).



Advanced Function Concepts | Others 3

- Ternary Operator:
  - Simplify conditional assignments: x = a if condition else b.



What is Serialization?

- Converting objects into a byte stream for storage or transmission.

pickle Module

- Common Functions:
  - pickle.dumps(data): Serialize.
  - pickle.dump(data, file): Serialize and save to file.
  - pickle.loads(data): Deserializ.
  - pickle.load(data): Deserialize from file.
- Use Cases:
  - Saving user sessions.
  - Persisting model states in machine learning.

pickle Module | Example

- Import pickle

- Data = {}
- with open(“address”, “wb”) as f:
  - Pickle.dump(data, f)

- with open(“address”, “rb”) as f:
  - Loaded_data = pickle.load(f)

Any question?

Next course

- Python Scripting
- Argument Parsing with argparse
- Date & Time Handling in Python
- Logging in Python