Bootcamp 134 | Python

Course 19 | Advanced Content

Amir Hossein Chegouniyan

Head of the Technical Team at Dariche Tejarat

Lecturer of Python – Django at Maktab Sharif

Amirhossein-chegounian

Content

Python Scripting

- Understanding Python Scripts.
- Difference between Modules and Scripts.
- The if __name__ == "__main__": construct.
- Organizing and structuring Python scripts for reusability and clarity.

Python Scripting | How to Create?

- Write a python file (For example: test.py)
- Add shebang (#!) at first of file (#!/usr/bin/env python3)
- Write your code!
- Run file in terminal (./test.py)
- Add output of script to a file with:
  - > output.txt
- Append output of script to a file with:
  - >> output.txt

Argument Parsing with argparse

- Overview of sys.argv vs argparse.
- Adding arguments: positional and optional.
- Argument types and validation (choices, type, required).
- Help messages and default values.
- Subcommands and mutually exclusive arguments.

Argument Parsing with argparse | How to Work?

```
import argparse

parser = argparse.ArgumentParser()      # create parser

# add arguments
parser.add_argument("name")   # positional argument
parser.add_argument("--age", type=int, default=18, help="Your age")

args = parser.parse_args()      # parse arguments

print(f"Hello {args.name}, you are {args.age} years old.")
```

Argument Parsing with argparse | Explain

- *parser.add_argument(name, help, choices, type, action)*
  - *name:*
    - *The name of the argument.*
    - *Can be positional ("filename") or optional with -- ("--verbose").*
  - *help: A description shown when the user runs --help.*
  - *choices: Restrict the input to a predefined set of values.*
  - *type: Convert the input to a specific type (default: str).*
  - *action: Defines what to do when the argument is encountered.*

Argument Parsing with argparse | action arg 1

- *action (common values):*
  - *“store” (default) → store the value*
    - *parser.add_argument("--name", action="store")*
  - *“store_true” → set to True if the flag is present*
  - *“store_false” → set to False if the flag is present*
    - *parser.add_argument("--verbose", action="store_true")*
  - *“append” → allow multiple uses, values stored in a list*
    - *parser.add_argument("--tag", action="append")*
    - *python script.py --tag python --tag ai*

Argument Parsing with argparse | action arg 1

- *action (common values):*
  - *“extend” → Extends a list with multiple values at once.*
    - *parser.add_argument("--nums", action="extend", nargs="+", type=int)*
    - *python script.py --nums 1 2 3 --nums 4 5*
  - *“count” → Counts how many times the flag appears.*
    - *parser.add_argument("-v", "--verbose", action="count", default=0)*
    - *python script.py -vvv*
  - *“version” → Prints the program version and exits.*
    - *parser.add_argument("--version", action="version", version="%(prog)s 1.0")*

Date & Time Handling in Python

- Working with the datetime and time modules.
- Getting and formatting the current date and time: strftime() and strptime().
- Calculations using timedelta.
- Converting timestamps to datetime and vice versa.

Date & Time Handling in Python | Datetime

- Import datetime
  - datetime.datetime.now()
  - datetime.date.today() # with .year, .month, .day, .hour, .minute, .second
  - datetime.datetime
  - datetime.date(year, month, day)
  - datetime.time(hour, minute, second)
  - datetime.datetime(year, month, day, hour, minute, second, microsecound)
  - delta = datetime.timedelta(days=5, hours=3)

Date & Time Handling in Python | Options

- datetime.datetime.now().strftime(pattern) # convert datetime to string
- datetime.datetime.strptime(string, pattern) # convert string to datetime
- datetime.datetime.fromtimestamp(number) # convert timestamp to datetime
- datetime.datetime.timestamp(datetime) # convert datetime to timestamp

Date & Time Handling in Python | Time

- import time
  - time.time()
  - time.ctime(second)
  - time.sleep(second)
  - …

Logging in Python

- The importance of logging over print().
- Logging levels: DEBUG, INFO, WARNING, ERROR, CRITICAL.
- Setting up and customizing the logging format.
- Writing logs to a file.
- Exception logging for debugging.

Any question?

Next course

- Setup and Configuration
- Working with Files
- Branching and Merging
- Remote Repositories
- Git Help