Python: From Command Line

Prashant K. Sharma
Diptesh Kanojia

e-Yantra Team
ERTS Lab, IIT Bombay

IIT Bombay
April 30, 2021

Agenda for Discussion

Why Command-Line Arguments ?

For exploration of data and prototyping code, IDEs, Jupyter Notebooks are useful. But, when it is required that we execute our Python code as script or as a part of a pipeline, we need our Python codes to work like other command line programs in UNIX-environment.

Command-Line Program

Write a Python program that prints all the command line arguments.

Command-Line Program

Write a Python program that prints all the command line arguments.

Using default **sys** module

```
import sys
print("Arguments: ", sys.argv)
```

Command-Line Program

Write a Python program to take 1 command line argument (eg. let's say its user's first name) and prints the script name and command line argument.

Command-Line Program

Write a Python program to take 1 command line argument (eg. let's say its user's first name) and prints the script name and command line argument.

Using default **sys** module

```
import sys
args = sys.argv
print("Script Name: ", args[0])
print("First Name: ", args[1])
```

Command-Line Program

Using main() function, write a Python program to take two command line arguments (i.e two numbers). and print their addition.

Command-Line Program

Using main() function, write a Python program to take two command line arguments (i.e two numbers). and print their addition.

Using default **sys** module

```
import sys

def main():
    args = sys.argv
    num1 = float(args[1])
    num2 = float(args[2])
    add = num1 + num2
    print("Addition: ", add)

if __name__ == "__main__":
    main()
```

Argparse

The **argparse** module is the default Python library which is recommended for creating user-friendly command line interfaces in Python.

Argparse

The **argparse** module is the default Python library which is recommended for creating user-friendly command line interfaces in Python.

Key features are:

- The program will define what arguments it requires and **argparse** will parse those out of **sys.argv**.
- This module automatically generates help and usage messages and issues error when user supplies invalid arguments.

Argparse: Positional Arguments

Using **argparse** module, write a program to take a positional argument "echo" and print the values.

Argparse: Positional Arguments

Using **argparse** module, write a program to take a positional argument "echo" and print the values.

Using **argparse** module

```
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("echo", help="echo the string")
args = parser.parse_args()
print(args.echo)
```

Argparse: Positional and Optional Arguments

Using **argparse** module, write a program to take one positional argument called "num" and an optional argument "verbosity" and print the square of the number.

Argparse: Positional and Optional Arguments

Using **argparse** module, write a program to take one positional argument called "num" and an optional argument "verbosity" and print the square of the number.

Using **argparse** module

```
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("num", type=int, help="input the number")
parser.add_argument("--verbosity", action="store_true",
                     help="readable")
args = parser.parse_args()
if args.verbosity:
    print("The square is: ", args.num**2)
else:
    print(args.num**2)
```

References

- argparse — Parser for command-line options, arguments and sub-commands
- Python Command Line arguments

Thank You!

Author: Prashant Sharma
Contributor: Diptesh Kanojia

Post your queries at: resources@e-yantra.org