Python CLI

Peb Ruswono Aryan

15.02.2020 @EOX

who is peb?

Research assistant

TU Wien Informatics

Information and software engineering

Semantics systems research

- Semantic Web
- Ontology
- Linked data
- Knowledge graph

Explainability in Cyber-Physical Systems

former Project assistant/Software Developer

TU Wien Geodesy

Photogrammetry

- Geometric spatial data
- Coordinate Reference System
- Map projection
- Processing workflow
- Machine learning

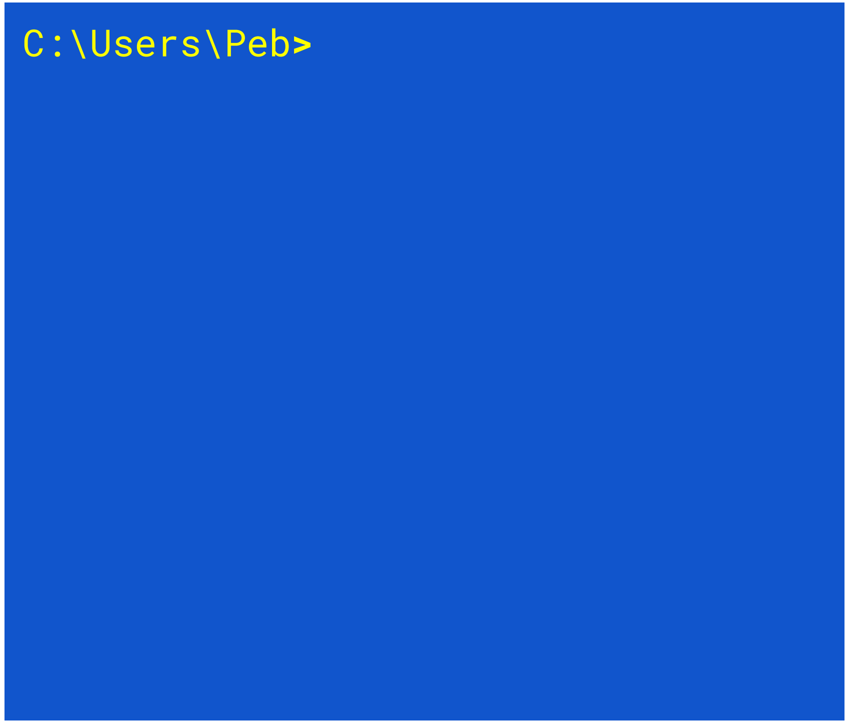
Plan

Command Line Interface

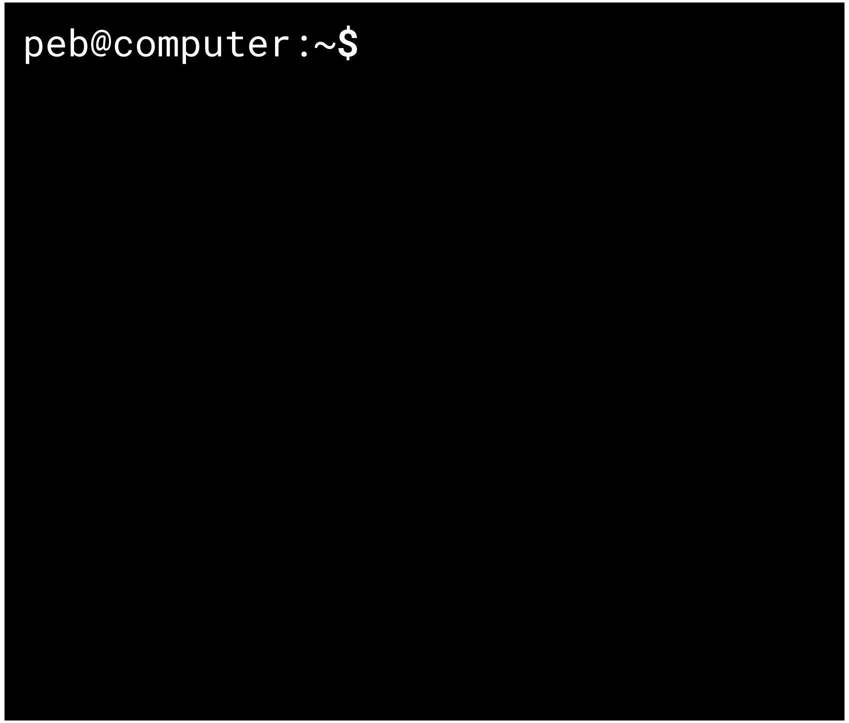
Argument Parsing

Exercise : filename matching using `glob` module

Command Line Interface



```
C:\Users\Peb>
```



```
peb@computer:~$
```

program.py - what’s in a __name__

```
if __name__ == “__main__”:
    print(“Hi There!”)
```

```
$ python program.py
Hi There!
$ python
>>>import program
>>>
```

arg00.py - passing arguments to a program

```python
from sys import argv

if __name__ == "__main__":

    for i in range(len(argv)):

        print(argv[i])
```

```
$ python arg00.py

arg00.py

$ python arg00.py 1 2 3

arg00.py

1

2

3
```

arg01.py

```
from argparse import ArgumentParser

if __name__ == “__main__”:

    parser = ArgumentParser()

    args = parser.parse_args()
```

```
$ python arg01.py

$ python arg01.py -h

usage: arg01.py [-h]

optional arguments:

 -h, --help  show this message and
exit
```

arg02.py - adding more information

```
from argparse import ArgumentParser

if __name__ == “__main__”:

    parser = ArgumentParser(

        description=”hi”

        , epilog=”bye”

    )

    args = parser.parse_args()
```

```
$ python arg02.py

$ python arg02.py -h

usage: arg02.py [-h]

hi

optional arguments:

 -h, --help  show this message and
exit

bye
```

arg03.py - positional argument

```
from argparse import ArgumentParser

if __name__ == “__main__”:

    parser = ArgumentParser()

    parser.add_argument(‘p’)

    args = parser.parse_args()

    print(args)
```

```
$ python arg03.py

usage: arg03.py [-h] p

arg03.py: error: the following
arguments are required: p

$ python arg03.py abc

Namespace(p=’abc’)
```

arg04.py - multiple values in an argument (fixed)

```
from argparse import ArgumentParser

if __name__ == “__main__”:

    parser = ArgumentParser()

    parser.add_argument(‘p’

             , nargs=2)

    args = parser.parse_args()

    print(args)
```

```
$ python arg04.py -h

usage: arg04.py [-h] p p

positional arguments:

 p

$ python arg04.py abc def

Namespace(p=[’abc’, ‘def’])
```

arg05.py - multiple values in an argument (variable)

```
from argparse import ArgumentParser

if __name__ == “__main__”:

    parser = ArgumentParser()

    parser.add_argument(‘p’

                , nargs=’*’)

    args = parser.parse_args()

    print(args)
```

```
$ python arg05.py

Namespace(p=[])

$ python arg05.py abc

Namespace(p=[’abc’])

$ python arg05.py abc def

Namespace(p=[’abc’, ‘def’])
```

arg06.py - help text

```python
from argparse import ArgumentParser

if __name__ == “__main__”:

    parser = ArgumentParser()

    parser.add_argument(‘p’

        , help=’p is any value’)

    args = parser.parse_args()

    print(args)
```

```
$ python arg06.py

usage: arg06.py [-h] p

arg06.py: error: the following
arguments are required: p

$ python arg06.py -h

usage: arg06.py [-h] p

positional arguments:

 p   p is any value
```

arg07.py - typed argument

```
from argparse import ArgumentParser

if __name__ == “__main__”:

    parser = ArgumentParser()

    parser.add_argument(‘p’

        , type=int)

    args = parser.parse_args()

    print(args)
```

```
$ python arg07.py abc

usage: arg07.py [-h] p

arg07.py: error: argument p:
invalid int value: ‘abc’

$ python arg07.py 1

Namespace(p=1)
```

arg08.py - named argument

```
from argparse import ArgumentParser

if __name__ == “__main__”:

    parser = ArgumentParser()

    parser.add_argument(‘-param’)

    args = parser.parse_args()

    print(args)
```

```
$ python arg08.py

Namespace(param=None)

$ python arg08.py -param abc

Namespace(param=’abc’)

$ python arg08.py -p def

Namespace(param=‘def’)
```

arg09.py - default value

```
from argparse import ArgumentParser

if __name__ == “__main__”:

    parser = ArgumentParser()

    parser.add_argument(‘-param’,
default=’xyz’)

    args = parser.parse_args()

    print(args)
```

```
$ python arg09.py

Namespace(param=’xyz’)

$ python arg09.py -param abc

Namespace(param=’abc’)

$ python arg09.py -p def

Namespace(param=‘def’)
```

arg10.py - switch argument

```
from argparse import ArgumentParser

if __name__ == “__main__”:

    parser = ArgumentParser()

    parser.add_argument(‘-param’

        , default=False

        , action=’store_true’)

    args = parser.parse_args()

    print(args)
```

```
$ python arg10.py

Namespace(param=False)

$ python arg10.py -p

Namespace(param=True)
```

arg11.py - mandatory named argument

```
from argparse import ArgumentParser

if __name__ == “__main__”:

    parser = ArgumentParser()

    parser.add_argument(‘-param’

        , required=True)

    args = parser.parse_args()

    print(args)
```

```
$ python arg11.py

$ python arg11.py -p

$ python arg11.py -p xyz
```

arg12.py - predefined values argument

```
from argparse import ArgumentParser

if __name__ == “__main__”:

    parser = ArgumentParser()

    parser.add_argument(‘-param’

        , type=int

        , choices=[0,1])

    args = parser.parse_args()

    print(args)
```

```
$ python arg12.py

$ python arg12.py -p

$ python arg12.py -p xyz

$ python arg12.py -p 2

$ python arg12.py -p 1
```

filematch.py - filename pattern matching

```
from argparse import ArgumentParser
from glob import glob
if __name__ == “__main__”:
    parser = ArgumentParser()
    parser.add_argument(‘pattern’
        , nargs=’+’)
    args = parser.parse_args()
    result = []
    for p in args.pattern:
        result += glob(p)
    for f in result:
        print(f)
```

```
$ python filematch.py arg

$ python filematch.py arg1*

arg10.py

arg11.py

arg12.py

$ python filematch.py -p arg*.py
pro*
```