

```

PPOptional.add_argument('-v', dest='verbose', action='count', default=0, help='increase verbosity level')
PPOptional.add_argument('--version', action='version', version='Version ' + str(constant.CURRENT_VERSION), help='laZagne version')

PWrite = argparse.ArgumentParser(add_help=False, formatter_class=lambda prog: argparse.HelpFormatter(prog, max_help_position=constant.MAX_HELP_POSITION))
PWrite._optionals.title = 'output'
PWrite.add_argument('-w', dest='write', action='store_true', help='write a text file on the current directory')

all_subparser = []
for c in category.keys():
    category[c]['parser'] = argparse.ArgumentParser(add_help=False, formatter_class=lambda prog: argparse.HelpFormatter(prog, max_help_position=constant.MAX_HELP_POSITION))
    category[c]['parser']._optionals.title = category[c]['help']

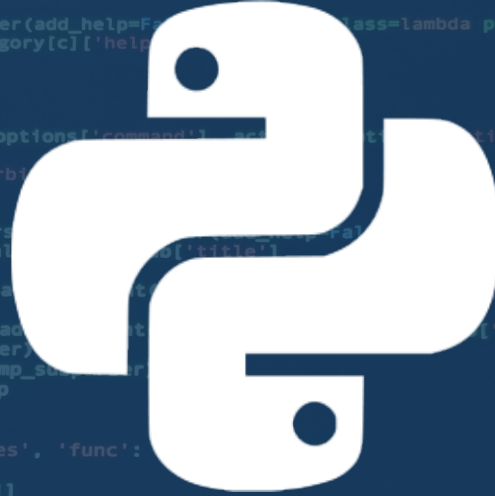
    category[c]['subparser'] = []
    for module in modules[c].keys():
        m = modules[c][module]
        category[c]['parser'].add_argument(m.options['command'], dest=m.options['dest'], help=m.options['help'])

        if m.suboptions and m.name != 'thunderbolt':
            tmp = []
            for sub in m.suboptions:
                tmp_subparser = argparse.ArgumentParser(add_help=False, formatter_class=lambda prog: argparse.HelpFormatter(prog, max_help_position=constant.MAX_HELP_POSITION))
                tmp_subparser._optionals.title = sub['title']
                if 'type' in sub:
                    tmp_subparser.add_argument(sub['command'], dest=sub['dest'], help=sub['help'], action=sub['action'], dest=sub['dest'], help=sub['help'])
                else:
                    tmp_subparser.add_argument(sub['command'], dest=sub['dest'], help=sub['help'], action=sub['action'], dest=sub['dest'], help=sub['help'])
                tmp.append(tmp_subparser)
            all_subparser.append(tmp_subparser)
            category[c]['subparser'] += tmp

parents = [PPOptional] + all_subparser + [PWrite]
dic = {'all': {'parents': parents, 'help': 'Run all modules', 'func': run_all_modules}}
for c in category.keys():
    parser_tab = [PPOptional, category[c]['parser']]
    if 'subparser' in category[c]:
        if category[c]['subparser']:
            parser_tab += category[c]['subparser']
    parser_tab += [PWrite]
    dic_tmp = {c: {'parents': parser_tab, 'help': 'Run %s module' % c, 'func': run_module}}
    dic = dict(dic.items() + dic_tmp.items())

subparsers = parser.add_subparsers(help='laZagne subcommands')
for d in dic.keys():
    subparsers.add_parser(d, parents=dic[d]['parents'], help=dic[d]['help']).set_defaults(func=dic[d]['func'], auditType=d)

```



# SALIDA DE DATOS

## PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

# CONCEPTO Y DEFINICIÓN

---

- ⦿ Cada programa debe poder comunicarse con su entorno
- ⦿ Para este propósito nacieron las funciones de entrada/salida
- ⦿ En el video anterior aprendieron cómo ingresar datos
- ⦿ Imprimiremos los resultados de nuestros programas en pantalla

# CONCEPTO Y DEFINICIÓN

---

- ⦿ La función `print()` es la solución de Python para esta situación

- Permite imprimir directamente un texto

```
>>> print("Hola Eduvolucion!")
```

```
Hola Eduvolucion!
```

- Puede imprimir el valor de una variable

```
>>> x = 3
```

```
>>> print(x)
```

```
3
```

# CONCEPTO Y DEFINICIÓN

---

⦿ La función `print()` es la solución de Python para esta situación

- Podemos imprimir varios valores separados por coma

```
>>> a = 5
```

```
>>> b = 7
```

```
>>> print(a, b)
```

- Permite mezclar impresión de textos y valores de variables

```
>>> nombre = "Carlos"
```

```
>>> print("Tu nombre es " + nombre)
```

# CONSIDERACIONES ESPECIALES

---

- ⦿ Si la variable es de tipo numérica, debemos utilizar la función `str()` para convertir su valor en una cadena

```
>>> edad = 35
```

```
>>> print("Tu edad es " + str(edad))
```