

```

PPOptional.add_argument('-v', dest='verbose', action='count', default=0, help='increase verbosity level')
PPOptional.add_argument('--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 != 'thunderbird':
            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 modules %s' % ' '.join(modules[c].keys()), 'func': 'run_modules'}}
    dic = dict(dic.items() + dic_tmp.items())

subparsers = parser.add_subparsers(help='Choose a main tool to run')
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)

```

NÚMEROS

PROGRAMACIÓN DE COMPUTADORAS III

Abdel G. Martínez L.

CATEGORÍAS

⦿ *Enteros*

- Números positivos o negativos sin cifras decimales
- Pueden ser de un tipo: `int`

```
>>> ent = 35
```

- Antes existían los `long`

CATEGORÍAS

⦿ *Enteros*

- Otro uso es para expresar números en octal o hexadecimal

```
>>> oct = 0o20
```

```
>>> hex = 0x17
```

CATEGORÍAS

⊙ *Reales*

- Números que tienen decimales
- Puede ser de un tipo: `float`
- Se expresan de dos maneras:
 - De forma tradicional, con una parte entera y una decimal

```
>>> pi = 3.1416
```

- Utilizando un exponente de base 10

```
>>> celula = 0.1e-5
```

CATEGORÍAS

⦿ *Complejos*

- Números que tienen una parte imaginaria
- Si no los conoce, quizás no tenga que utilizarlos
- Los puede reconocer porque tienen con una j al final
- Su uso más común es en la geología, ingeniería y física

```
>>> latitud = 15 + 3j
```