```python
class PauloBenatto(object):
    """Paulo Leonardo Benatto is a hobbit developer, sometimes confused with
       Oompa Loompa, from Shire, living and drinking pints in Brighton, UK.
       Yeah, I know, I'm weird.
    """

    def __init__(self):
        self.type = "hobbit"
        self.interests = ["learning", "coding", "python", "go", "linux"]

    def contact(self):
        # douchebags like me feel hacker using b64 =)
        self.email = "YmVuYXR0b0BnbWFpbC5jb20="
        self.linkedin = "https://www.linkedin.com/in/benatto/"
        self.github = "https://github.com/patito"
        self.blog = "http://benatto.xyz"

    def weapons(self):
        """I use different weapons in different wars =). I know Pascal as well
           but I'm emberassed to add in my CV hehe.
        """
        return {
            "python": "I use for task automation, CLIs or running out of time.",
            "golang": "I use for REST APIs and because I love Rob Pike.",
            "c": "I use @?5?h@R?%. Segmentation fault.",
            "javascript": "I think I know how to use it, but I don't",
            "linux": "I use it to have fluxbox running and matrix screensaver.",
            "stack overflow": "This is my teacher, lover and soulmate."
        }

    def battles(self):
        """I have gained some experience being involved in different battles"""

        my_battles = {}
        my_battles["return_of_brandwatch"] = {
            "when": "July 2016 - Present",
            "warrior": "Core Systems Engineer",
            "tasks": "Automating tasks in Python and Go.",
            "weapons": ["linux", "golang", "rest apis"],
        }

        my_battles["sainsburys"] = {
            "when": "Feb 2016 - July 2016",
            "warrior": "Jr. Go Developer",
            "tasks": "Developed Microservices in Go language",
            "weapons": ["linux", "golang", "rest apis"]
        }

        my_battles["brandwatch"] = {
            "when": "July 2014 - Feb 2016",
            "warrior": "Jr. linux System Administrator",
            "tasks": """Keep internal services running.""",
            "weapons": ["linux", "git", "bacula", "ansible", "automation"]
        }

        my_battles["dba"] = {
            "when": "Dec 2013 - Feb 2014",
            "warrior": "Software Engineer Freelancer",
            "tasks": """Develop a system, in C language, to analyse vehicle
                        traffic on Brazilian highways.""",
            "weapons": ["linux", "c", "git", "python", "raspberry pi"]
        }

        my_battles["secplus"] = {
            "when": "Dec 2012 - Dev 2013",
            "warrior": "Software Engineer",
            "tasks": """I have worked in natural disaster monitoring system and
                        C library to malware analysis.""",
            "weapons": ["linux", "c", "git", "python"]
        }

        return my_battles

if __name__ == "__main__":
    pb = PauloBenatto()
    print(pb.battles())
```

```python
# I wil tell you something I wrote the test after finishing my CV code.
# Yeah, I'm that crazy. I know I could test better, but it is just a CV and
# I promisse I will not merge with production branch or deploy in live, and
# I will create a check to monitor on Sensu and Prometheus.

# Run my test: python3 -m unittest -v -b test_cv.py

import unittest
from cv import PauloBenatto


class TestPauloBentto(unittest.TestCase):

    def setUp(self):
        self.plb = PauloBenatto()

    def test_battles(self):
        battles = self.plb.battles()
        self.assertEqual(len(battles), 5)
        self.assertEqual(battles["dba"]["warrior"],
                         "Software Engineer Freelancer")
        self.assertEqual(battles["return_of_brandwatch"]["weapons"],
                         ["linux", "golang", "rest apis"])
        self.assertEqual(len(list(filter(lambda x: x == "c",
                         battles["secplus"]["weapons"]))), 1)
        # Please, be equal 0, please
        self.assertEqual(len(list(filter(lambda x: x == "pascal",
                         battles["brandwatch"]["weapons"]))), 0)

    def test_weapons(self):
        weapons = self.plb.weapons()
        keys = weapons.keys()
        self.assertEqual(len(keys), 6)
        self.assertEqual("python" in keys, True)
        self.assertEqual("linux" in keys, True)
        # please be false _o/
        self.assertEqual("delphi" in keys, False)

    def test_contact(self):
        self.plb.contact()
        # Now you know my email =), but I'm still a douchebag
        import base64
        # converting bytes to str
        self.assertEqual(base64.b64decode(self.plb.email).decode("utf-8"),
                         "benatto@gmail.com")
        self.assertEqual(self.plb.blog, "http://benatto.xyz")
        self.assertEqual(self.plb.linkedin, "https://www.linkedin.com/in/benatto/")
        self.assertEqual(self.plb.github, "https://github.com/patito")

    def test_type(self):
        # please be true =)
        self.assertEqual(self.plb.type == "human", False)
        # please be false =(
        self.assertEqual(self.plb.type == "hobbit", True)

    def test_interest(self):
        self.assertEqual("surf" in self.plb.interests, False)
        self.assertEqual("guitar" in self.plb.interests, False)
        self.assertEqual("rock'n'roll" in self.plb.interests, False)
        # What is wrong with me, if you are reading that u know the
        # answer.
        self.assertEqual("coding" in self.plb.interests, True)
        self.assertEqual("linux" in self.plb.interests, True)


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