Linter - Pylint

“É uma ferramenta de análise de código estática usada para sinalizar erros de programação, bugs, erros estilísticos e construções suspeitas.”

https://coodesh.com/blog/dicionario/o-que-e-lint/

Adicionar Pylint

setup.py

```
1  from setuptools import setup, find_packages
2
3  setup(
4      name='severino',
5      version='1.0.0',
6      packages=find_packages(include=['source.*', 'source.excecoes.*', 'source.habilidades.*', 'source.helpers.*', 'source.modelos.*']),
7      install_requires=[
8          'mypy',
9          'pytest',
10         'pytest-cov',
11         'requests',
12         'types-requests',
13+        'pylint',
14     ]
15 )
```

* make setup

  Gerar arquivo de configurações:

* venv/bin/pylint --generate-rcfile > configs/.pylintrc

Atualizar Makefile #1

Makefile

```
 1  python = $(shell which python3)
 2  VENV = venv
 3  PIP = $(VENV)/bin/pip
 4  PYTHON = $(VENV)/bin/python
 5  MYPY = $(VENV)/bin/mypy
 6  PYTEST = $(VENV)/bin/pytest
 7+PYLINT = $(VENV)/bin/pylint
 8
 9  all:
10      $(PYTHON) source/main.py
11  setup:
12      $(python) -m venv $(VENV)
13      $(PIP) install --upgrade setuptools
14      $(PIP) install -e .
15  mypy:
16      $(MYPY) source tests --config-file configs/mypy.ini
17  pytest:
18      $(PYTEST) --cov-report term-missing:skip-covered --cov=source
19+pylint:
20+     $(PYLINT) source/* tests/* --rcfile=configs/.pylintrc
```

* make pylint

Atualizar Pylint

configs/.pylintrc

max-line-length

200

min-public-methods

0

check-quote-consistency

yes

ignore-none

no

Atualizar Pylint

configs/.pylintrc

[MESSAGES CONTROL]

disable

```
disable=raw-checker-failed,
        bad-inline-option,
        locally-disabled,
        file-ignored,
        suppressed-message,
        useless-suppression,
        deprecated-pragma,
        use-symbolic-message-instead,
        use-implicit-booleaness-not-comparison-to-string,
-       use-implicit-booleaness-not-comparison-to-zero
+       use-implicit-booleaness-not-comparison-to-zero,
+       raise-missing-from,
```

Atualizar Pylint

Comentários

configs/.pylintrc

[MESSAGES CONTROL]

disable

```
disable=raw-checker-failed,
        bad-inline-option,
        locally-disabled,
        file-ignored,
        suppressed-message,
        useless-suppression,
        deprecated-pragma,
        use-symbolic-message-instead,
        use-implicit-booleaness-not-comparison-to-string,
        use-implicit-booleaness-not-comparison-to-zero,
        raise-missing-from,
+       missing-module-docstring,
+       missing-class-docstring,
```

Atualizar Pylint

Funções

`max-args`

4

`max-nested-blocks`

1

`configs/.pylintrc`

Arrumando erros

```
************* Module api_cotacoes
source/helpers/api_cotacoes.py:1:0: C0114: Missing module docstring (missing-module-docstring)
source/helpers/api_cotacoes.py:9:0: C0115: Missing class docstring (missing-class-docstring)
source/helpers/api_cotacoes.py:24:0: C0206: Consider iterating with .items() (consider-using-dict-items)
source/helpers/api_cotacoes.py:24:64: C0201: Consider iterating the dictionary directly instead of calling .keys() (consider-iterating-dictionary)
```

```python
def __transformar_moedas_em_lista(self) -> list[str]:
  return [f'{chave}: {self.__moedas[chave]}' for chave in self.__moedas.keys()]
```

Arrumando erros

```
************* Module api_cotacoes
source/helpers/api_cotacoes.py:1:0: C0114: Missing module docstring (missing-module-docstring)
source/helpers/api_cotacoes.py:9:0: C0115: Missing class docstring (missing-class-docstring)
source/helpers/api_cotacoes.py:24:0: C0206: Consider iterating with .items() (consider-using-dict-items)
source/helpers/api_cotacoes.py:24:64: C0201: Consider iterating the dictionary directly instead of calling .keys() (consider-iterating-dictionary)
```

```python
def __transformar_moedas_em_lista(self) -> list[str]:
  return [f'{chave}: {self.__moedas[chave]}' for chave in self.__moedas.keys()]
```

```python
def __transformar_moedas_em_lista(self) -> list[str]:
  return [f'{chave}: {valor}' for chave, valor in self.__moedas.items()]
```

Arrumando erros

```
************* Module mock_resposta
tests/mock/mock_resposta.py:11:0: C0301: Line too long (361/200) (line-too-long)
tests/mock/mock_resposta.py:1:0: C0114: Missing module docstring (missing-module-docstring)
tests/mock/mock_resposta.py:6:0: C0115: Missing class docstring (missing-class-docstring)
tests/mock/mock_resposta.py:11:4: R0913: Too many arguments (7/5) (too-many-arguments)
tests/mock/mock_resposta.py:11:22: W0613: Unused argument 'cls' (unused-argument)
tests/mock/mock_resposta.py:11:60: W0613: Unused argument 'object_hook' (unused-argument)
tests/mock/mock_resposta.py:11:120: W0613: Unused argument 'parse_float' (unused-argument)
tests/mock/mock_resposta.py:11:169: W0613: Unused argument 'parse_int' (unused-argument)
tests/mock/mock_resposta.py:11:216: W0613: Unused argument 'parse_constant' (unused-argument)
tests/mock/mock_resposta.py:11:268: W0613: Unused argument 'object_pairs_hook' (unused-argument)
tests/mock/mock_resposta.py:11:0: W0613: Unused argument 'kwds' (unused-argument)
```

```python
def json(self, *, cls: type[JSONDecoder] | None = None, object_hook: Callable[[dict[Any,
Any]], Any] | None = None, parse_float: Callable[[str], Any] | None = None, parse_int:
Callable[[str], Any] | None = None, parse_constant: Callable[[str], Any] | None = None,
object_pairs_hook: Callable[[list[tuple[Any, Any]]], Any] | None = None, **kwds: Any) -> Any:
    return self.json_para_retornar
```

Arrumando erros

```
************* Module mock_resposta
tests/mock/mock_resposta.py:11:0: C0301: Line too long (361/200) (line-too-long)
tests/mock/mock_resposta.py:1:0: C0114: Missing module docstring (missing-module-docstring)
tests/mock/mock_resposta.py:6:0: C0115: Missing class docstring (missing-class-docstring)
tests/mock/mock_resposta.py:11:4: R0913: Too many arguments (7/5) (too-many-arguments)
tests/mock/mock_resposta.py:11:22: W0613: Unused argument 'cls' (unused-argument)
tests/mock/mock_resposta.py:11:60: W0613: Unused argument 'object_hook' (unused-argument)
tests/mock/mock_resposta.py:11:120: W0613: Unused argument 'parse_float' (unused-argument)
tests/mock/mock_resposta.py:11:169: W0613: Unused argument 'parse_int' (unused-argument)
tests/mock/mock_resposta.py:11:216: W0613: Unused argument 'parse_constant' (unused-argument)
tests/mock/mock_resposta.py:11:268: W0613: Unused argument 'object_pairs_hook' (unused-argument)
tests/mock/mock_resposta.py:11:0: W0613: Unused argument 'kwds' (unused-argument)
```

```python
def json(self, *, cls: type[JSONDecoder] | None = None, object_hook: Callable[[dict[Any,
Any]], Any] | None = None, parse_float: Callable[[str], Any] | None = None, parse_int:
Callable[[str], Any] | None = None, parse_constant: Callable[[str], Any] | None = None,
object_pairs_hook: Callable[[list[tuple[Any, Any]]], Any] | None = None, **kwds: Any) -> Any:
    return self.json_para_retornar
```

```python
# pylint: disable=line-too-long,unused-argument,too-many-arguments
def json(self, *, cls: type[JSONDecoder] | None = None, object_hook: Callable[[dict[Any,
Any]], Any] | None = None, parse_float: Callable[[str], Any] | None = None, parse_int:
Callable[[str], Any] | None = None, parse_constant: Callable[[str], Any] | None = None,
object_pairs_hook: Callable[[list[tuple[Any, Any]]], Any] | None = None, **kwds: Any) -> Any:
    return self.json_para_retornar
```

Arrumando erros

```
************* Module tests.test_habilidade_conversao_moeda
tests/test_habilidade_conversao_moeda.py:1:0: C0114: Missing module docstring (missing-module-docstring)
tests/test_habilidade_conversao_moeda.py:13:0: C0115: Missing class docstring (missing-class-docstring)
tests/test_habilidade_conversao_moeda.py:69:15: C0121: Comparison 'io_manager_spy.iniciar_chamado == False' should be 'io_manager_spy.iniciar_chamado is False'
 if checking for the singleton value False, or 'not io_manager_spy.iniciar_chamado' if testing for falsiness (singleton-comparison)
```

```
assert io_manager_spy.iniciar_chamado == False
```

Arrumando erros

```
************* Module tests.test_habilidade_conversao_moeda
tests/test_habilidade_conversao_moeda.py:1:0: C0114: Missing module docstring (missing-module-docstring)
tests/test_habilidade_conversao_moeda.py:13:0: C0115: Missing class docstring (missing-class-docstring)
tests/test_habilidade_conversao_moeda.py:69:15: C0121: Comparison 'io_manager_spy.iniciar_chamado == False' should be 'io_manager_spy.iniciar_chamado is False'
 if checking for the singleton value False, or 'not io_manager_spy.iniciar_chamado' if testing for falsiness (singleton-comparison)
```

```
assert io_manager_spy.iniciar_chamado == False
```

```
assert io_manager_spy.iniciar_chamado is False
```

Arrumando erros

```
************* Module tests.test_severino
tests/test_severino.py:1:0: C0114: Missing module docstring (missing-module-docstring)
tests/test_severino.py:12:0: C0115: Missing class docstring (missing-class-docstring)
tests/test_severino.py:63:0: W0613: Unused argument 'args' (unused-argument)
```

```python
def test_get_textos_ajuda_deve_pedir_textos_de_ajuda_para_habilidades(self,
mock_construtor_habilidade_sistema: Mock, *args: tuple[Any]) -> None:
```

Arrumando erros

```
************* Module tests.test_severino
tests/test_severino.py:1:0: C0114: Missing module docstring (missing-module-docstring)
tests/test_severino.py:12:0: C0115: Missing class docstring (missing-class-docstring)
tests/test_severino.py:63:0: W0613: Unused argument 'args' (unused-argument)
```

```python
def test_get_textos_ajuda_deve_pedir_textos_de_ajuda_para_habilidades(self,
mock_construtor_habilidade_sistema: Mock, *args: tuple[Any]) -> None:
```

```python
def test_get_textos_ajuda_deve_pedir_textos_de_ajuda_para_habilidades(self,
mock_construtor_habilidade_sistema: Mock, *_: tuple[Any]) -> None:
```

Como garantir que todos os desenvolvedores irão rodar o Pylint e o Mypy para validar seus códigos?

Como garantir que todos os desenvolvedores irão rodar o Pylint e o Mypy para validar seus códigos?

Hook de pre-commit

Como garantir que todos os desenvolvedores irão rodar o Pylint e o Mypy para validar seus códigos?

Hook de pre-commit

* `ls .git/hooks`

```
applypatch-msg.sample           post-update.sample              pre-merge-commit.sample
        pre-receive.sample              update.sample
commit-msg.sample               pre-applypatch.sample           pre-push.sample
        prepare-commit-msg.sample
fsmonitor-watchman.sample       pre-commit.sample               pre-rebase.sample
        push-to-checkout.sample
```

Todos os momentos que podemos executar um script 👆

A pasta .git é ignorada pelo git, então temos que compartilhar o script de pre-commit de outra maneira

Atualizar Makefile #2

Makefile

```
 1 python = $(shell which python3)
 2 VENV = venv
 3 PIP = $(VENV)/bin/pip
 4 PYTHON = $(VENV)/bin/python
 5 MYPY = $(VENV)/bin/mypy
 6 PYTEST = $(VENV)/bin/pytest
 7 PYLINT = $(VENV)/bin/pylint
 8
 9 all:
10     $(PYTHON) source/main.py
11 setup:
12     $(python) -m venv $(VENV)
13     $(PIP) install --upgrade setuptools
14     $(PIP) install -e .
15+    cp scripts/pre-commit .git/hooks
16+    chmod 777 .git/hooks/pre-commit
17 mypy:
18     $(MYPY) source tests --config-file configs/mypy.ini
19 pytest:
20     $(PYTEST) --cov-report term-missing:skip-covered --cov=source
21     pylint:
22     $(PYLINT) source/* tests/* --rcfile=configs/.pylintrc
```

* make setup