# PHP CodeSniffer

CodeSniffer is a PHP application, so you must have the PHP CLI installed on your system. I tested with PHP 8.1.2 (the standard PHP CLI from the Ubuntu repository)

## Installation on Windows using Composer

Taken from this guide on pixelspress.com

1. If not already installed on your system, run the Composer installer from https://getcomposer.org/Composer-Setup.exe

Make sure Composer is installed correctly by opening a command window and using the command

```
composer help
```

2. Use Composer to globally install phpcs

```
composer global require "squizlabs/php_codesniffer=*"
```

Check that phpcs is installed correctly by opening a command window and using the command

```
phpcs -i
```

which should produce output showing the default coding standards.

## Installation on Ubuntu 22.04 using Composer

1. If you have installed phpcs from the Ubuntu repository, remove that version

```
sudo apt remove --purge php-codesniffer
```

2. If you have not yet installed Composer, install from the Ubuntu repository

```
sudo apt install composer
```

3. Use Composer to globally install phpcs

```
composer global require "squizlabs/php_codesniffer=*"
```

Add a symlink in .**`$HOME/.config/composer/vendor/squizlabs/php_codesniffer/`** called **`scripts`** as a symlink to the folder **`bin`** (This was necessary on my system as without it phpcs reported an error that the scripts folder did not exist. Your mileage may vary.)

Add the composer local bin folder to the PATH. Add to end of **`$HOME/.profile`**

```
export PATH="$PATH:$HOME/.config/composer/vendor/bin"
```

Log out and back in and ensure phpcs is now on the PATH by running

```
phpcs -i
```

which should produce output showing the default coding standards.

## Adding the extra coding standards (Windows or Linux)

Working in a terminal or command window, use Composer to add the additional coding standards scripts ("sniffs"):

1. Add the PHP compatibility sniff

```
composer global require phpcompatibility/php-compatibility
```

2. Add the WordPress sniffs

```
composer global require wp-coding-standards/wpcs:"^3.0"
```

3. Check default configuration

```
    phpcs -i
```

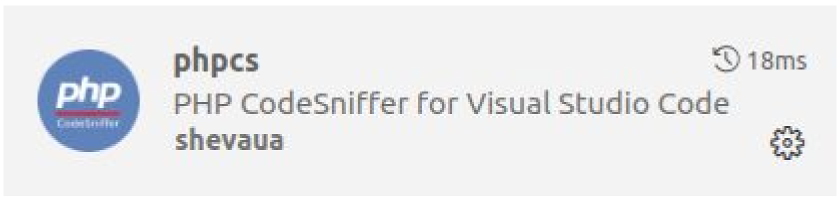
should produce this output:

```
    The installed coding standards are MySource, PEAR, PSR1, PSR2,
    PSR12, Squiz, Zend, PHPCompatibility, Modernize, NormalizedArrays,
    Universal, PHPCSUtils, WordPress, WordPress-Core, WordPress-Docs
    and WordPress-Extra
```

PHP Code Sniffer is now set up to check files to a range of standards.

## Using phpcs in VSCode

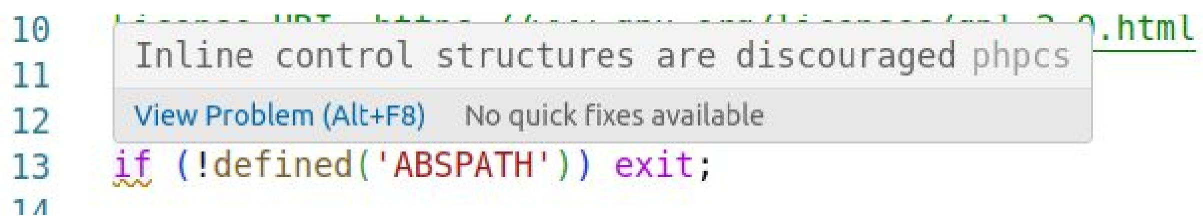
In VSCode, install the **phpcs** extension from **shevaua**



phpcs 18ms
PHP CodeSniffer for Visual Studio Code
shevaua

The default configuration which the extension sets up is fine.
When working on a plugin the default rules from the global phpcs settings will be used, unless overridden by a phpcs.xml configuration file.

Issues identified by phpcs will be shown with a wavy red underscore, and the hover info will show the issue:



```
Inline control structures are discouraged phpcs
View Problem (Alt+F8)    No quick fixes available
if (!defined('ABSPATH')) exit;
```

### Using a local phpcs configuration file

You can override the global default settings for phpcs by putting a phpcs.xml configuration file in the top folder of the code your are working on (ie at the same level as the .git and .vscode folders).

The phpcs extension in VSCode will use the settings in the phpcs.xml file, if present.

## Using phpcs on the command line

Some examples:

**phpcs *path/to/filename.php***
will check a file using the global default settings

**phpcs --standard=*PEAR*  *path/to/filename.php***
will check a file using the specified standard

**phpcs -s *path/to/filename.php***
will check a file using the global default settings and show the rule name for each issue found

**phpcs *path/to/filename.php***
in a folder containing phpcs.xml will check a file using the rules in the file

**phpcs -i**
will show which standards are available

**phpcs --help**
list all available commands