CODE STANDARDS & BEST PRACTICES

Md. Ibrahim Rashid [irashid.com]

Good Code Vs. Bad Code

Good Code Vs. Bad Code

- "The best applications are coded properly"
- This sounds like an obvious statement, but by 'properly', I mean that the code not only does its job well, but is also easy to add to, maintain and debug.

Ask Yourself?

- Is your code well organized and maintainable?
- Is you code well documented?

General Practices

- Naming Conventions
- Indentation
- Brace Style
- Commenting
- Code consistency
- Readability Vs. Compression

What should coding standards provide?

- File, class, variable naming conventions
- Code formatting conventions
- Guidelines for consistency across the code
- Uniformity

Naming Conventions

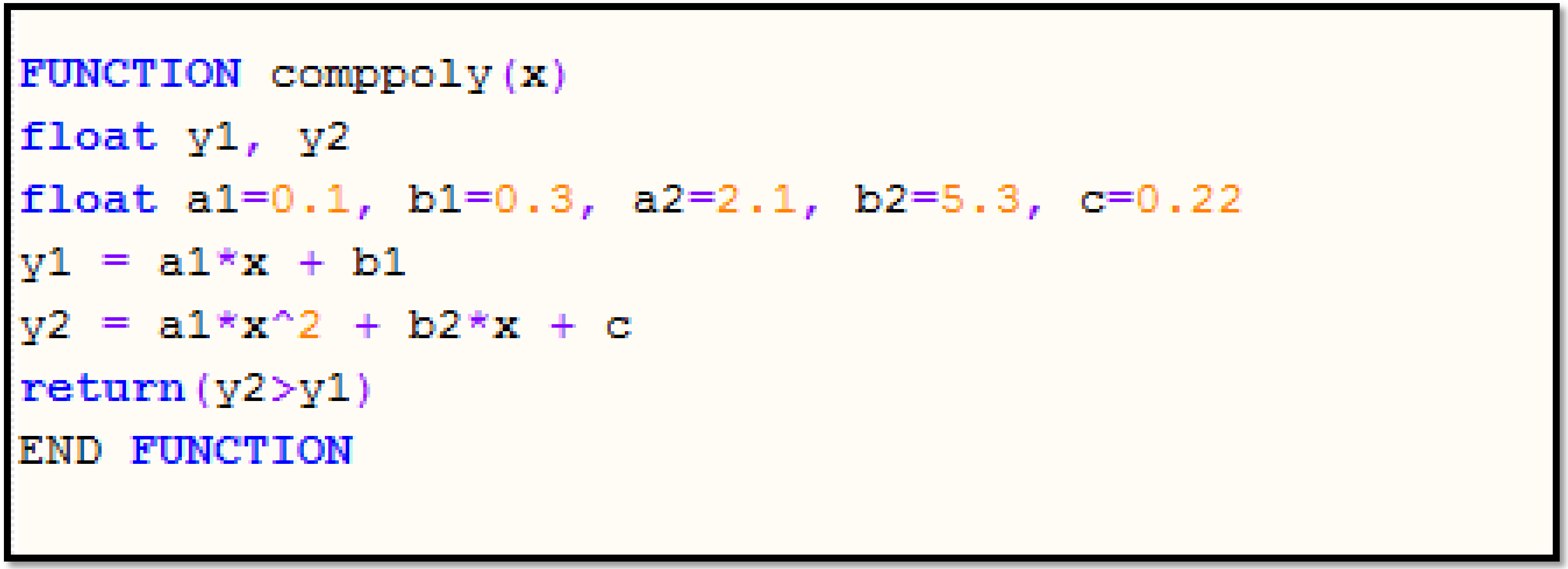
- Class names are MixedCase
  - [ex. MyClass]
- Method names are camelCase
  - [ex. myMethod()]
- Constants are ALL_CAPS
  - [MY_CONSTANT]
- Properties and variables are camelCase
  - [ex. myMethod()]
- Non-public class members are _underscorePrefixed
  - [Ex. _myPrivateVariable]

Various Conventions

Class names	MyClass		
Method names	my_function()	myFunction()	MyFunction
Constants	MY_CONSTANT		
Properties and variables	my_variable	myVariable	
Non-public class members	_my_private_variable	_myPrivateVariable	
Filenames	MyFile.php	myFile.php	my_file.php
Class Filenames	ClassMyFile.php	classMyFile.php	class_my_file.php

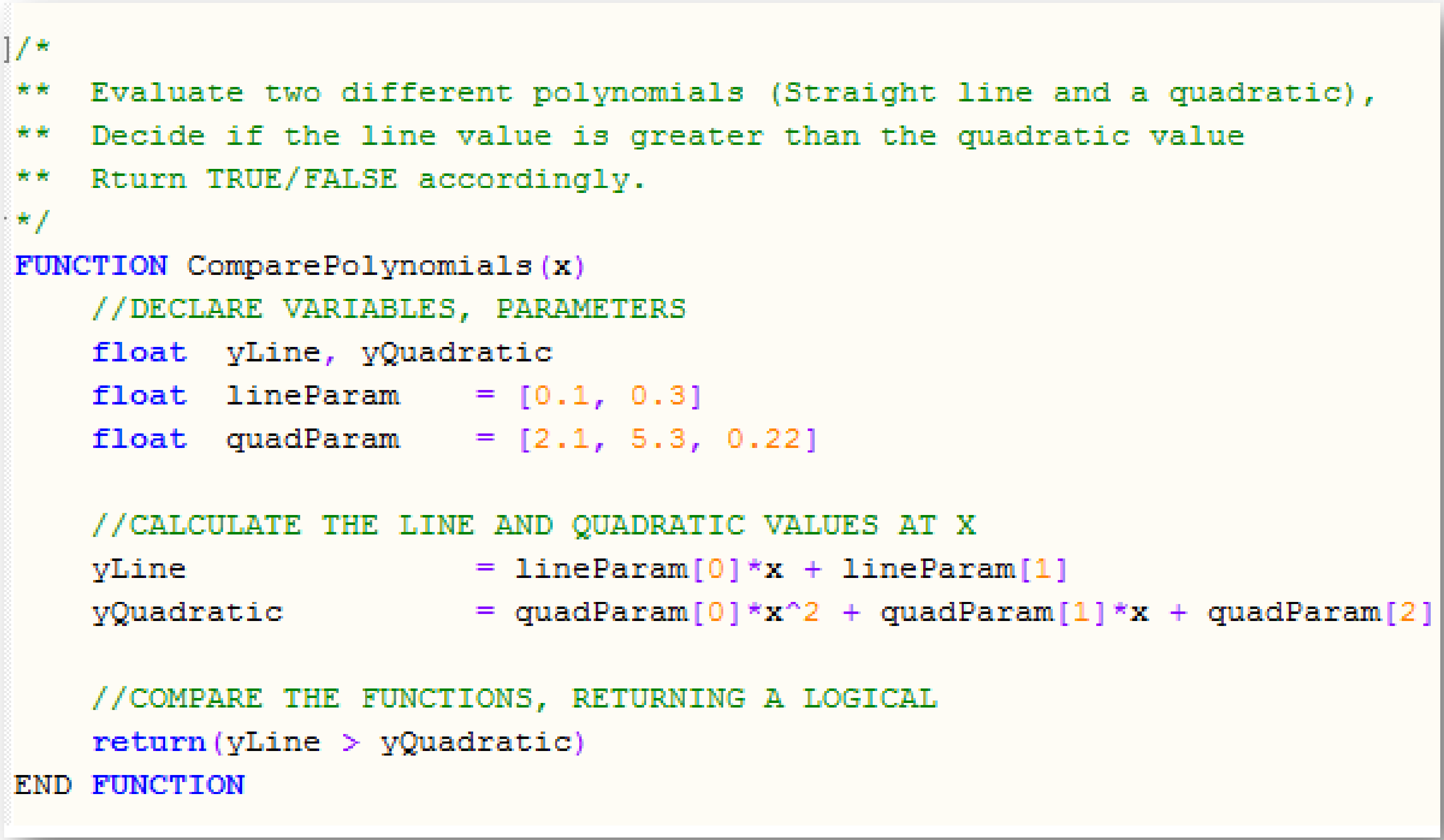
Case insensitive : MyFile.php and myfile.php are same in windows

Example



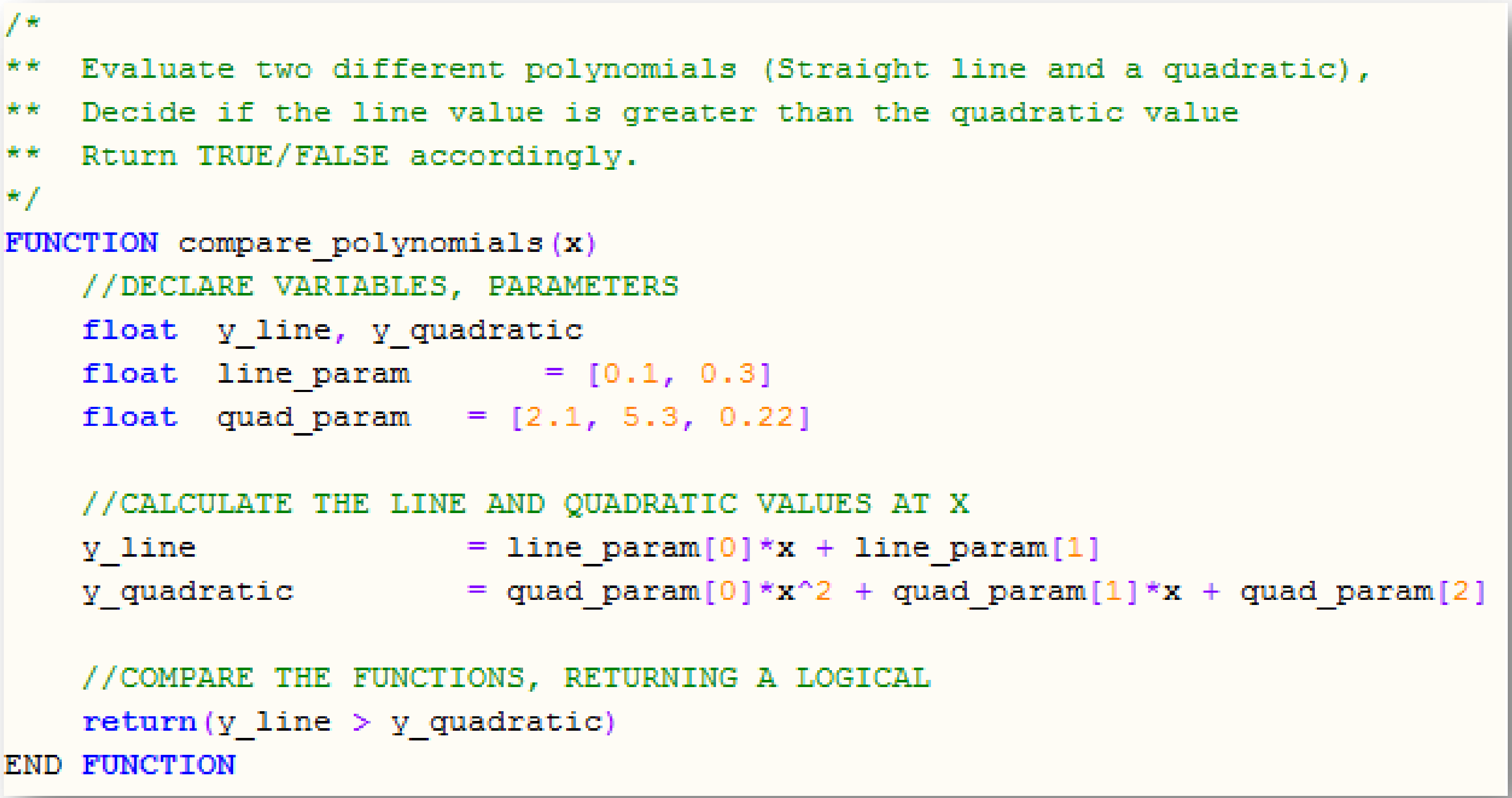
```
FUNCTION comppoly(x)
float y1, y2
float a1=0.1, b1=0.3, a2=2.1, b2=5.3, c=0.22
y1 = a1*x + b1
y2 = a1*x^2 + b2*x + c
return(y2>y1)
END FUNCTION
```

Scenario 1



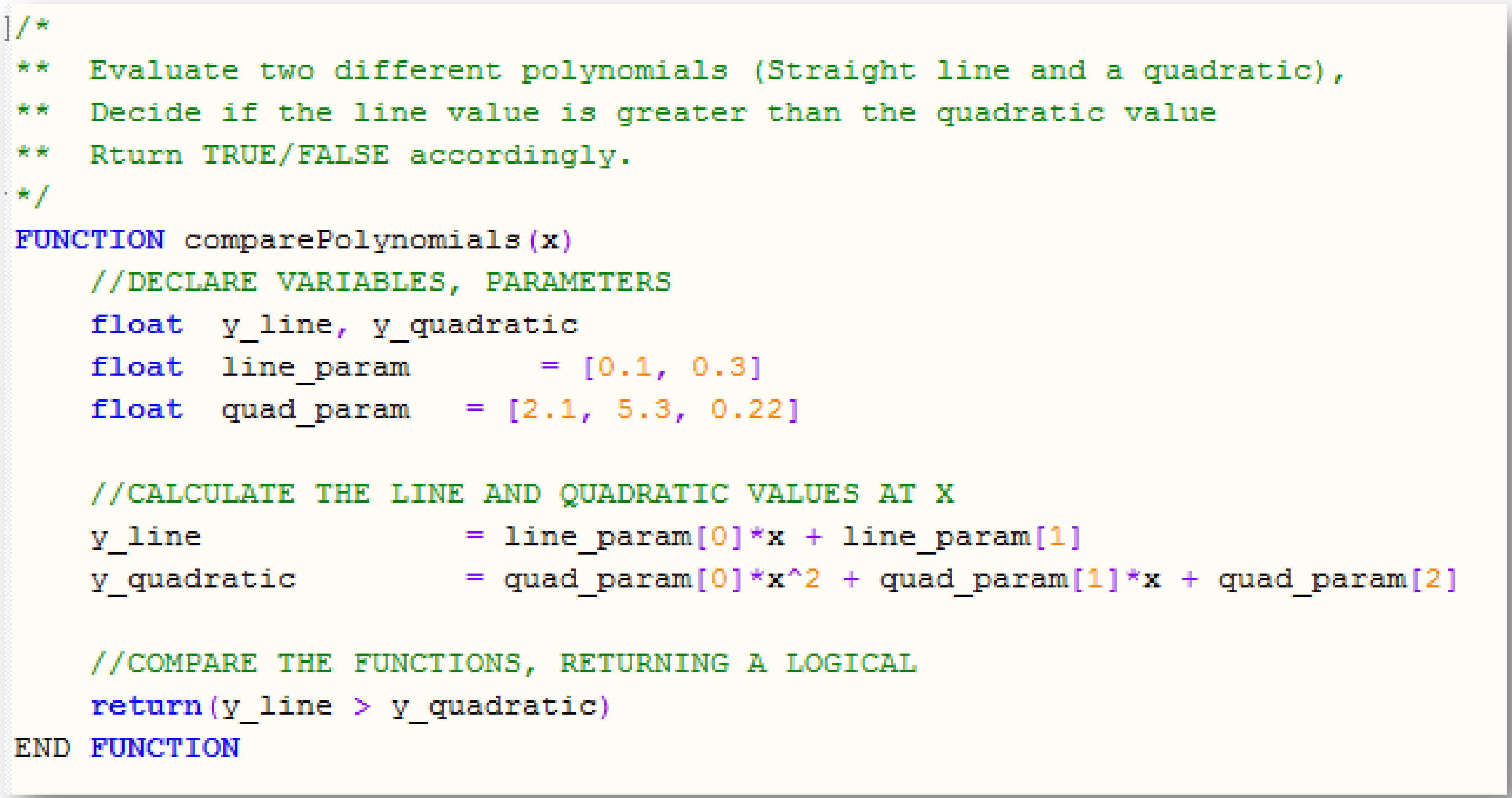
```
/*
**  Evaluate two different polynomials (Straight line and a quadratic),
**  Decide if the line value is greater than the quadratic value
**  Rturn TRUE/FALSE accordingly.
*/
FUNCTION ComparePolynomials(x)
    //DECLARE VARIABLES, PARAMETERS
    float  yLine, yQuadratic
    float  lineParam    = [0.1, 0.3]
    float  quadParam    = [2.1, 5.3, 0.22]

    //CALCULATE THE LINE AND QUADRATIC VALUES AT X
    yLine               = lineParam[0]*x + lineParam[1]
    yQuadratic          = quadParam[0]*x^2 + quadParam[1]*x + quadParam[2]

    //COMPARE THE FUNCTIONS, RETURNING A LOGICAL
    return(yLine > yQuadratic)
END FUNCTION
```

Scenario 2



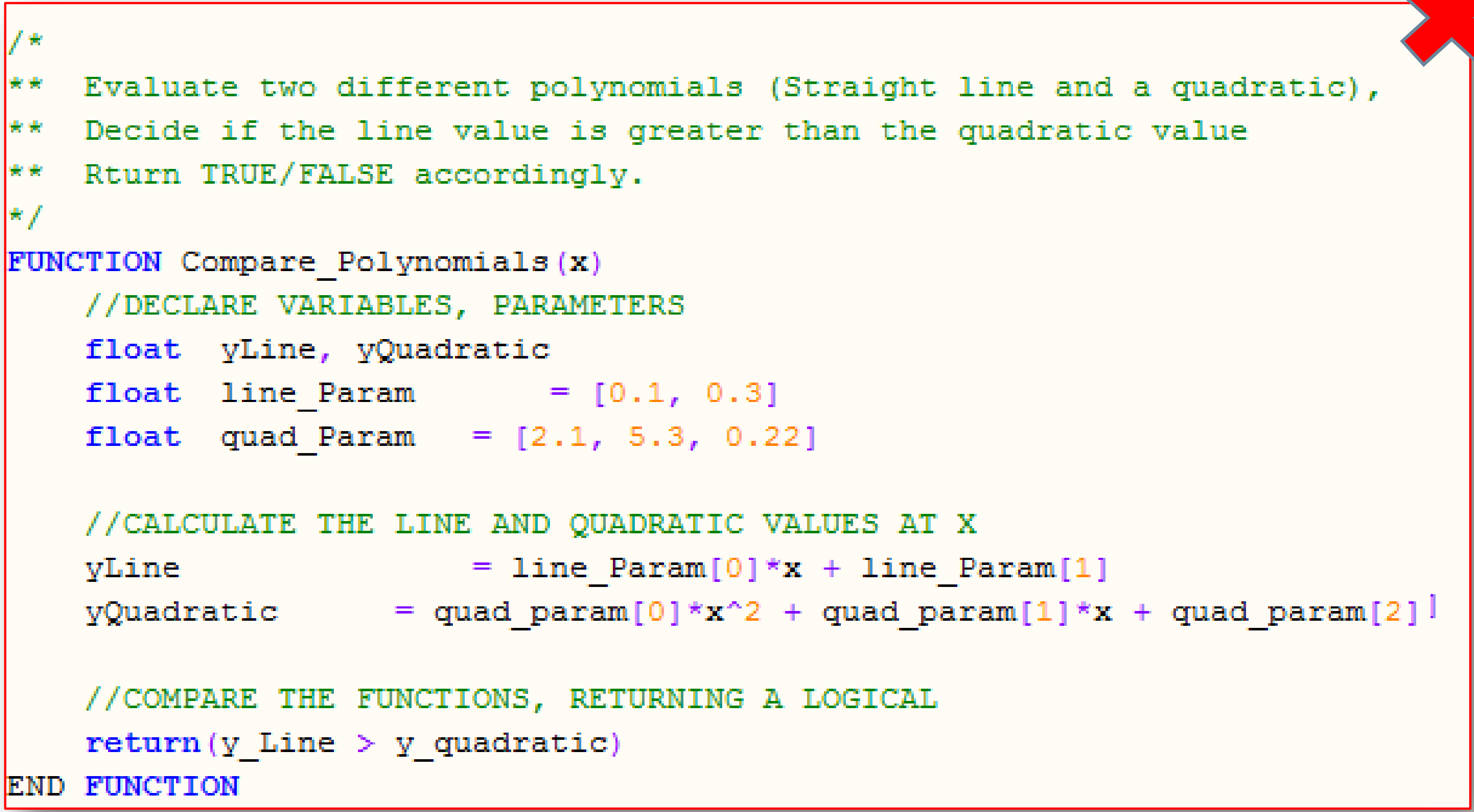
```
/*
**  Evaluate two different polynomials (Straight line and a quadratic),
**  Decide if the line value is greater than the quadratic value
**  Rturn TRUE/FALSE accordingly.
*/
FUNCTION compare_polynomials(x)
    //DECLARE VARIABLES, PARAMETERS
    float  y_line, y_quadratic
    float  line_param       = [0.1, 0.3]
    float  quad_param   = [2.1, 5.3, 0.22]

    //CALCULATE THE LINE AND QUADRATIC VALUES AT X
    y_line              = line_param[0]*x + line_param[1]
    y_quadratic         = quad_param[0]*x^2 + quad_param[1]*x + quad_param[2]

    //COMPARE THE FUNCTIONS, RETURNING A LOGICAL
    return(y_line > y_quadratic)
END FUNCTION
```

Scenario 3



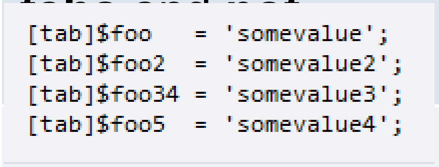
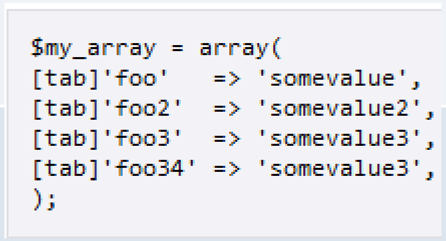
```
/*
**  Evaluate two different polynomials (Straight line and a quadratic),
**  Decide if the line value is greater than the quadratic value
**  Rturn TRUE/FALSE accordingly.
*/
FUNCTION comparePolynomials(x)
    //DECLARE VARIABLES, PARAMETERS
    float  y_line, y_quadratic
    float  line_param       = [0.1, 0.3]
    float  quad_param   = [2.1, 5.3, 0.22]

    //CALCULATE THE LINE AND QUADRATIC VALUES AT X
    y_line              = line_param[0]*x + line_param[1]
    y_quadratic         = quad_param[0]*x^2 + quad_param[1]*x + quad_param[2]

    //COMPARE THE FUNCTIONS, RETURNING A LOGICAL
    return(y_line > y_quadratic)
END FUNCTION
```

Senerio 4(!BAD Don’t MIX)



```
/*
**   Evaluate two different polynomials (Straight line and a quadratic),
**   Decide if the line value is greater than the quadratic value
**   Rturn TRUE/FALSE accordingly.
*/
FUNCTION Compare_Polynomials(x)
    //DECLARE VARIABLES, PARAMETERS
    float  yLine, yQuadratic
    float  line_Param       = [0.1, 0.3]
    float  quad_Param   = [2.1, 5.3, 0.22]

    //CALCULATE THE LINE AND QUADRATIC VALUES AT X
    yLine              = line_Param[0]*x + line_Param[1]
    yQuadratic        = quad_param[0]*x^2 + quad_param[1]*x + quad_param[2]

    //COMPARE THE FUNCTIONS, RETURNING A LOGICAL
    return(y_Line > y_quadratic)
END FUNCTION
```

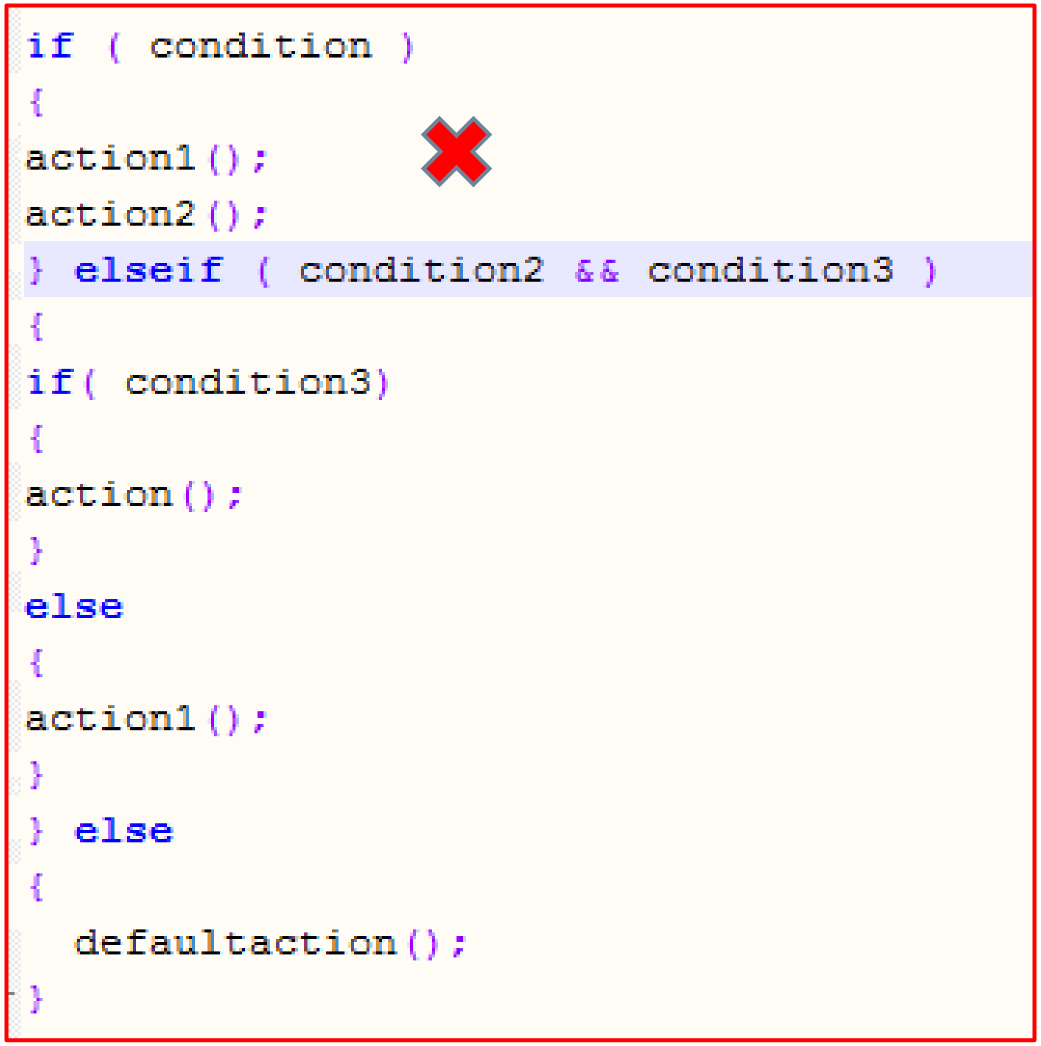
Indentation

PHP(Drupal)	Wordpress(PHP)	C(K&R standard)	
Use an indent of 2 spaces, with no tabs	Use **real** tabs and not  	Tab = 4 spaces	

```
[tab]$foo   = 'somevalue';
[tab]$foo2  = 'somevalue2';
[tab]$foo34 = 'somevalue3';
[tab]$foo5  = 'somevalue4';
```

```
$my_array = array(
[tab]'foo'   => 'somevalue',
[tab]'foo2'  => 'somevalue2',
[tab]'foo3'  => 'somevalue3',
[tab]'foo34' => 'somevalue3',
);
```

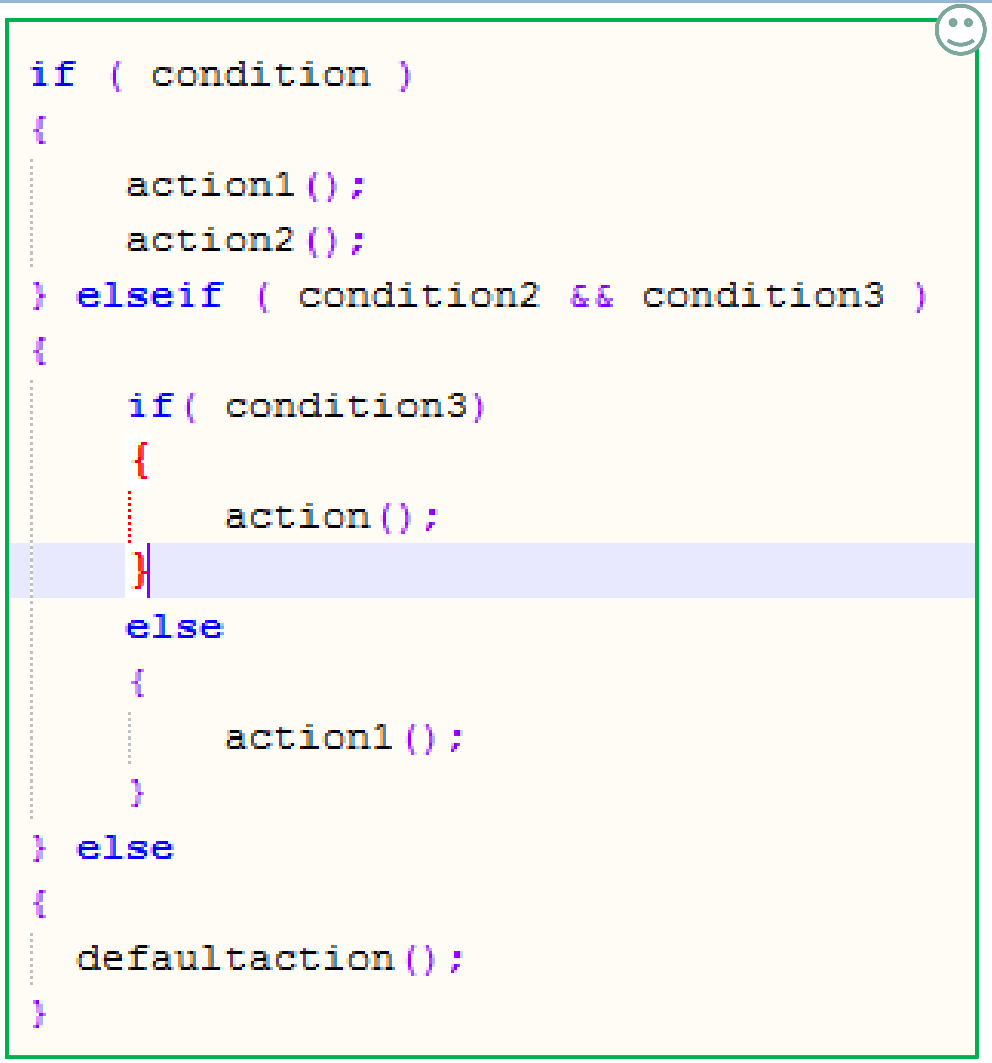
Indentation



```
if ( condition )
{
action1();
action2();
} elseif ( condition2 && condition3 )
{
if( condition3)
{
action();
}
else
{
action1();
}
} else
{
  defaultaction();
}
```

Indentation

- Always Make Proper Indent.



```
if ( condition )
{
    action1();
    action2();
} elseif ( condition2 && condition3 )
{
    if( condition3)
    {
        action();
    }
    else
    {
        action1();
    }
} else
{
  defaultaction();
}
```

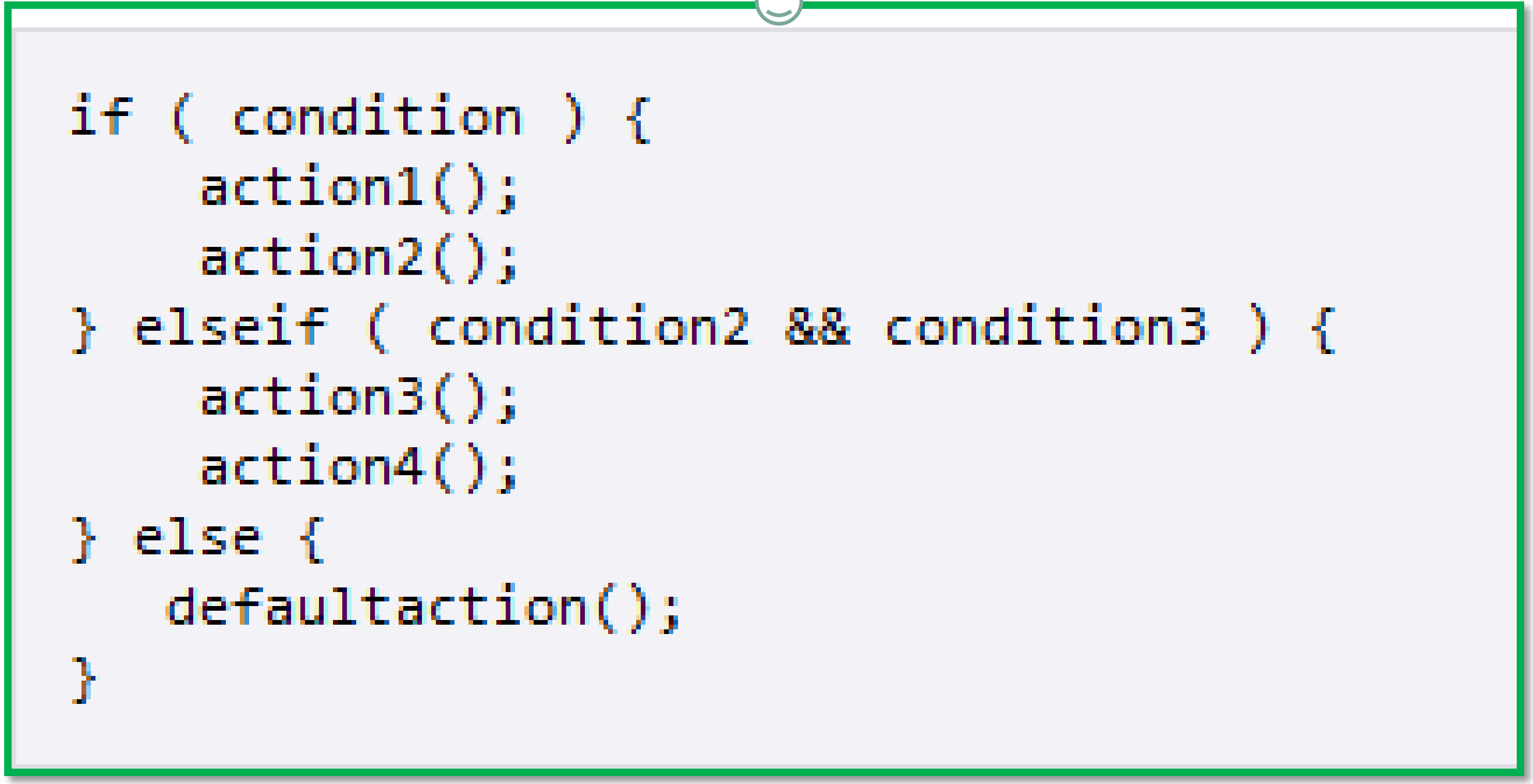
Indentation

- Use of Real Tabs, 4 space as Tabs, only spaces is

  controversial.
- It’s because same source code loaded into different editors with distinct setting will not look alike.
- Use lines less than 80 characters.

Indentation

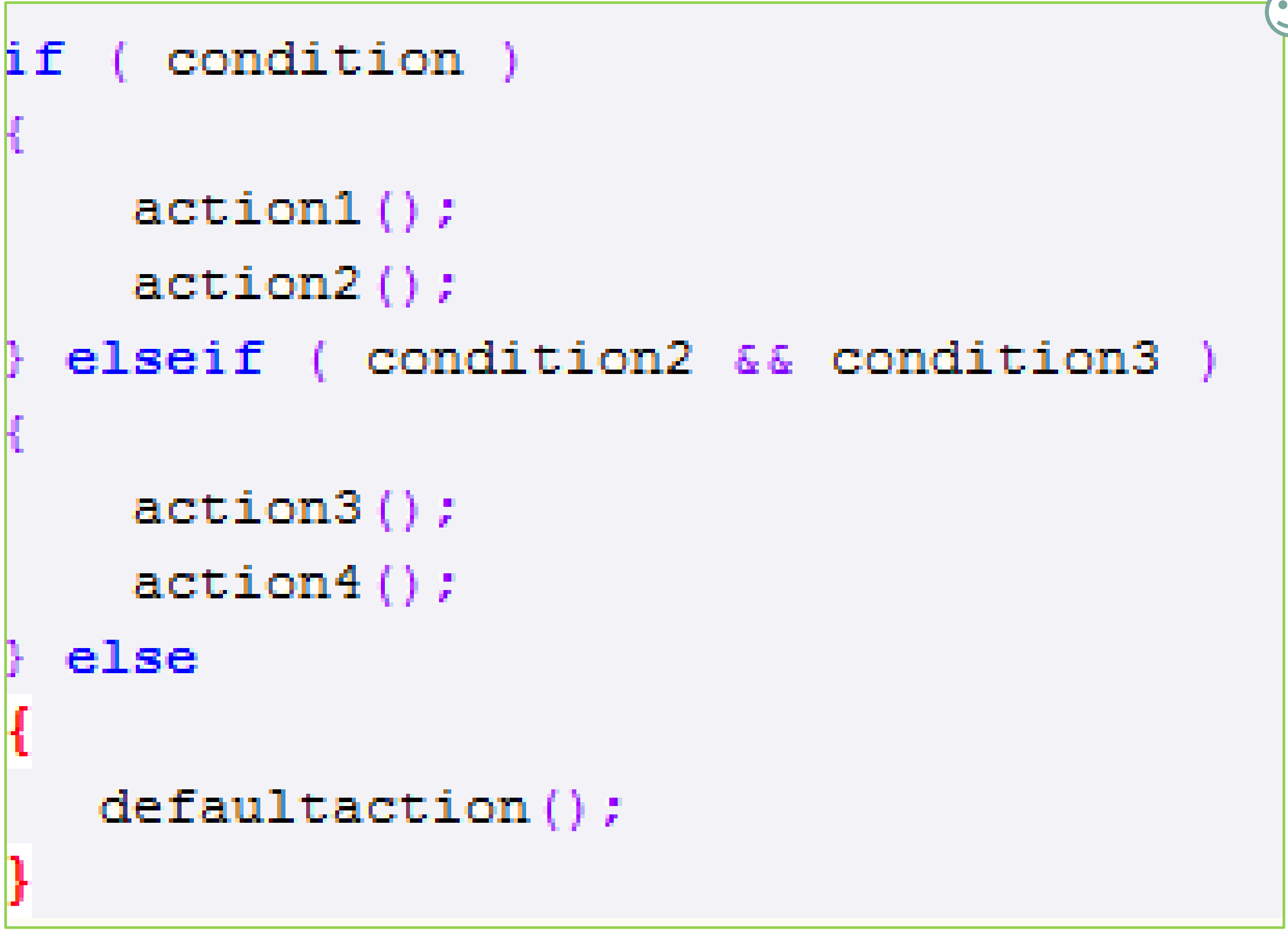
- Brace Style



```
if ( condition ) {
    action1();
    action2();
} elseif ( condition2 && condition3 ) {
    action3();
    action4();
} else {
   defaultaction();
}
```

Indentation

- Brace Style

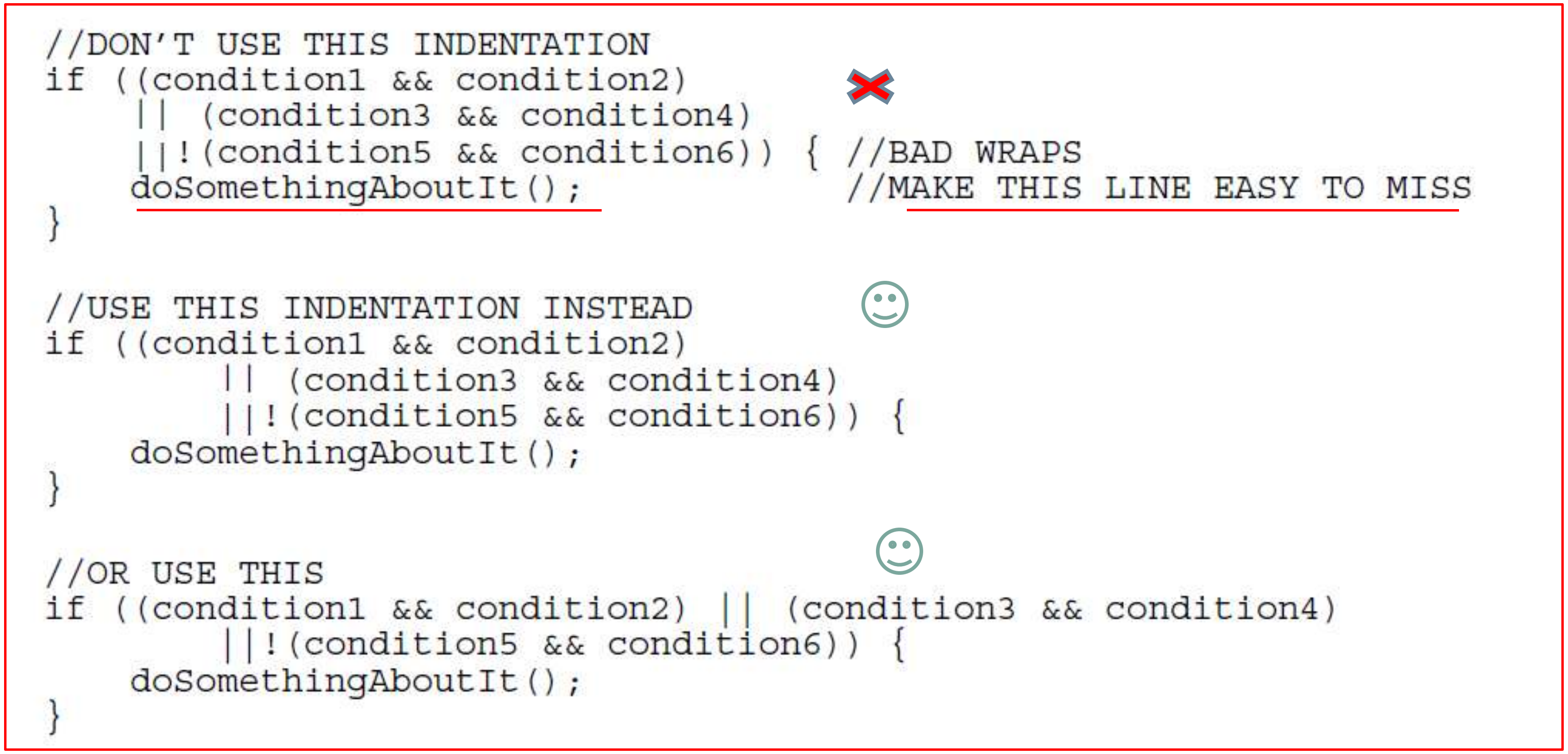


```
if ( condition )
{
    action1();
    action2();
} elseif ( condition2 && condition3 )
{
    action3();
    action4();
} else
{
   defaultaction();
}
```

Indentation

- Use of Real Tabs, 4 space as Tabs, only spaces is controversial.
- It’s because same source code loaded into different editors with distinct setting will not look alike.
- Use lines less than 80 characters.

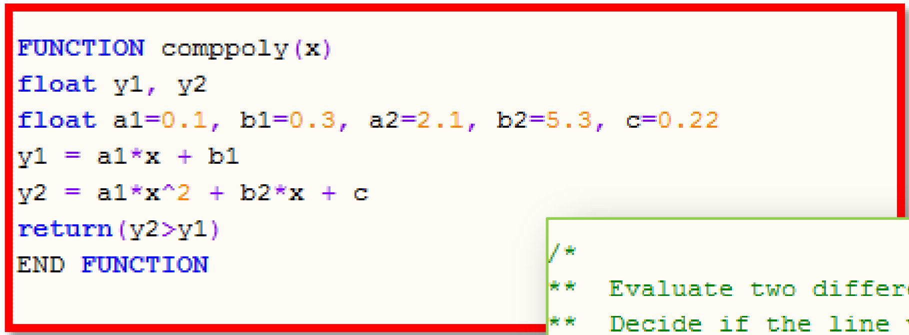
Indentation



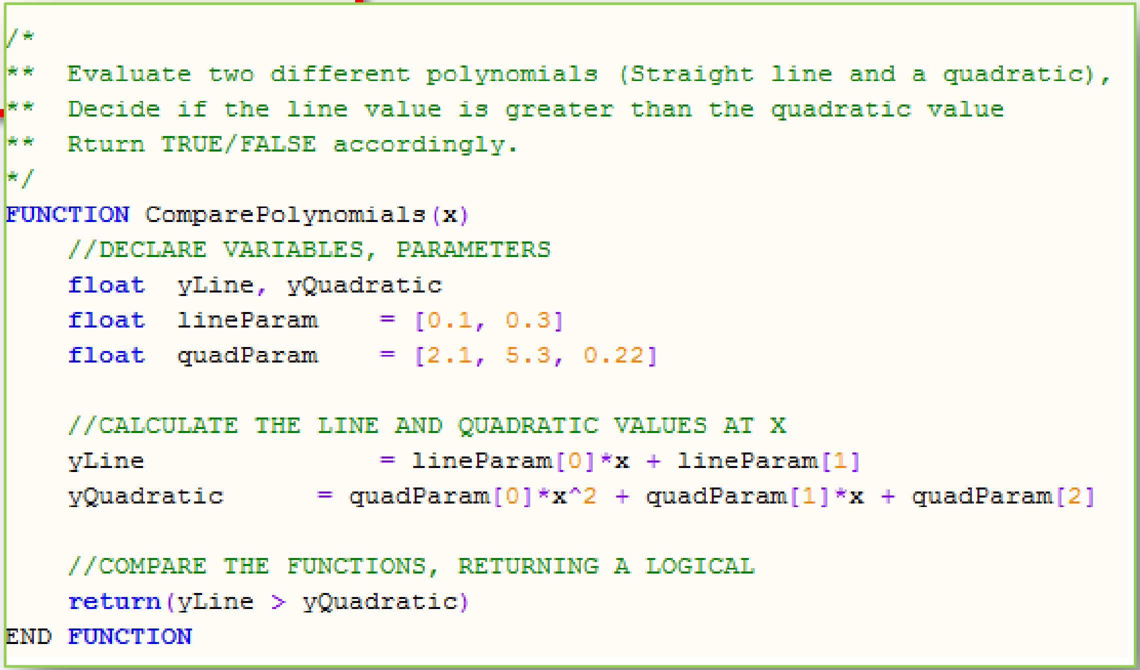
```
//DON'T USE THIS INDENTATION
if ((condition1 && condition2)
    || (condition3 && condition4)
    ||!(condition5 && condition6)) { //BAD WRAPS
    doSomethingAboutIt();            //MAKE THIS LINE EASY TO MISS
}

//USE THIS INDENTATION INSTEAD
if ((condition1 && condition2)
        || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}

//OR USE THIS
if ((condition1 && condition2) || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}
```

Commenting

- Always try to put comments on your code.

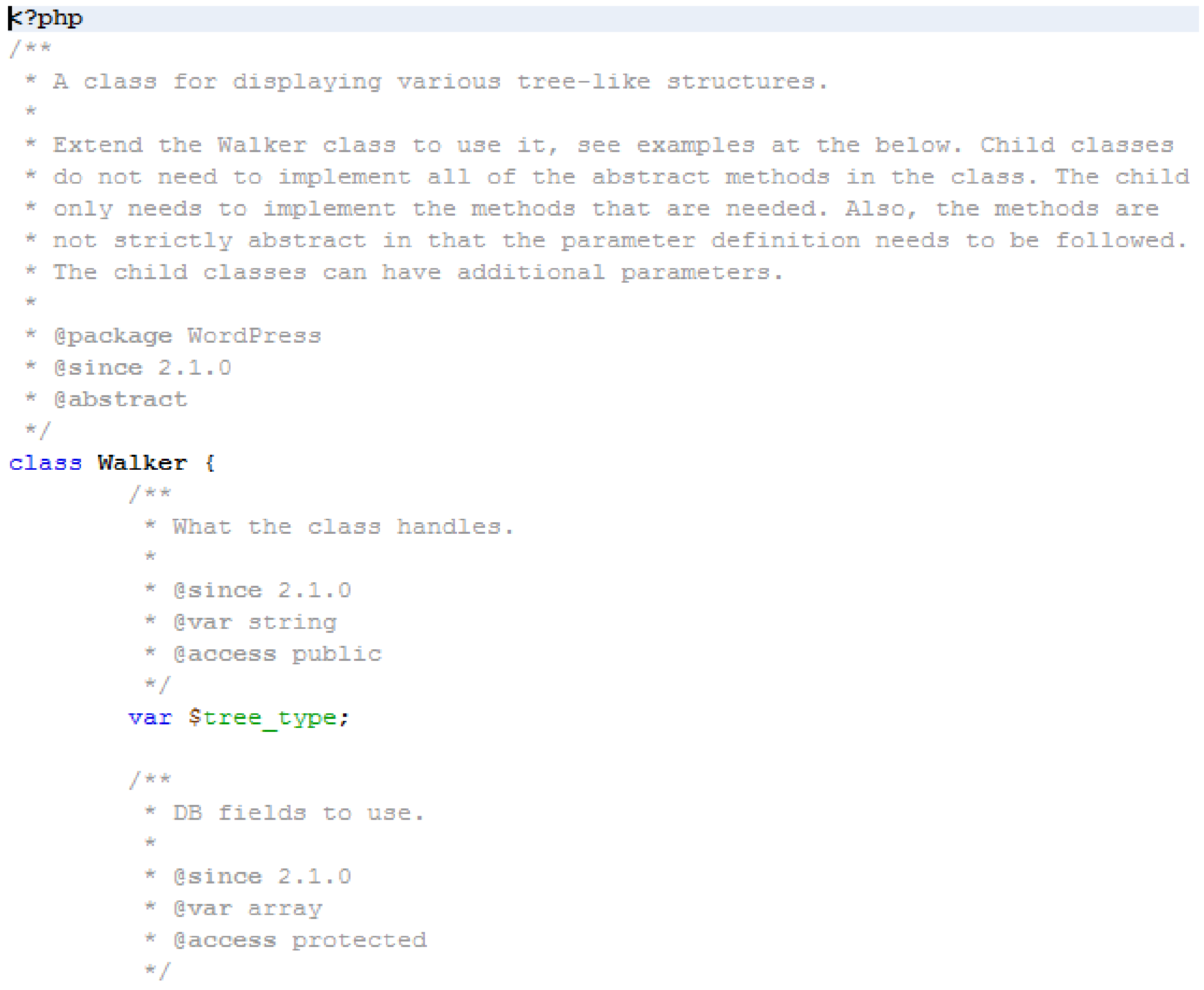


```
FUNCTION comppoly(x)
float y1, y2
float a1=0.1, b1=0.3, a2=2.1, b2=5.3, c=0.22
y1 = a1*x + b1
y2 = a1*x^2 + b2*x + c
return(y2>y1)
END FUNCTION
```



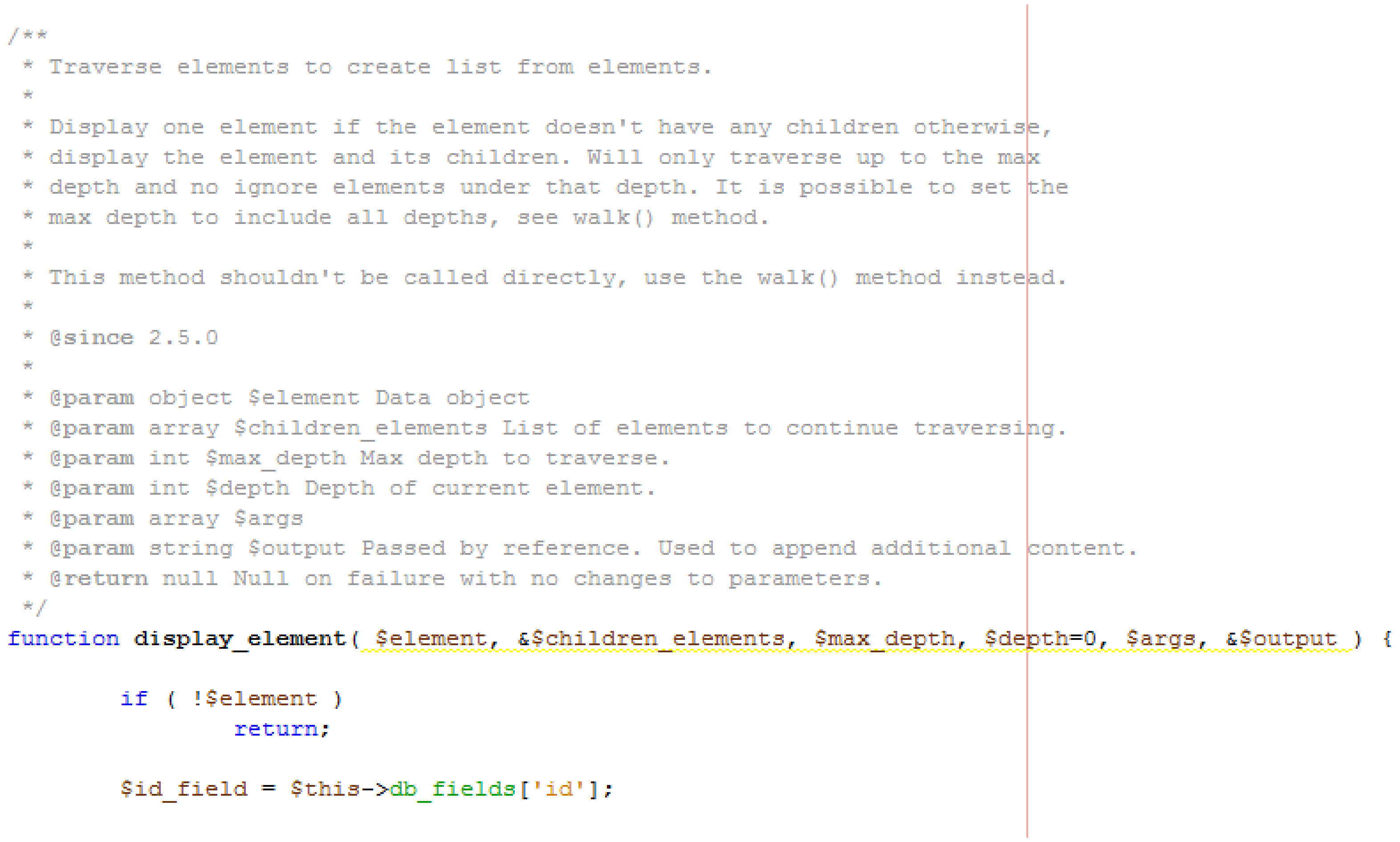
```
/*
**  Evaluate two different polynomials (Straight line and a quadratic),
**  Decide if the line value is greater than the quadratic value
**  Rturn TRUE/FALSE accordingly.
*/
FUNCTION ComparePolynomials(x)
    //DECLARE VARIABLES, PARAMETERS
    float  yLine, yQuadratic
    float  lineParam     = [0.1, 0.3]
    float  quadParam     = [2.1, 5.3, 0.22]

    //CALCULATE THE LINE AND QUADRATIC VALUES AT X
    yLine                = lineParam[0]*x + lineParam[1]
    yQuadratic       = quadParam[0]*x^2 + quadParam[1]*x + quadParam[2]

    //COMPARE THE FUNCTIONS, RETURNING A LOGICAL
    return(yLine > yQuadratic)
END FUNCTION
```

Commenting



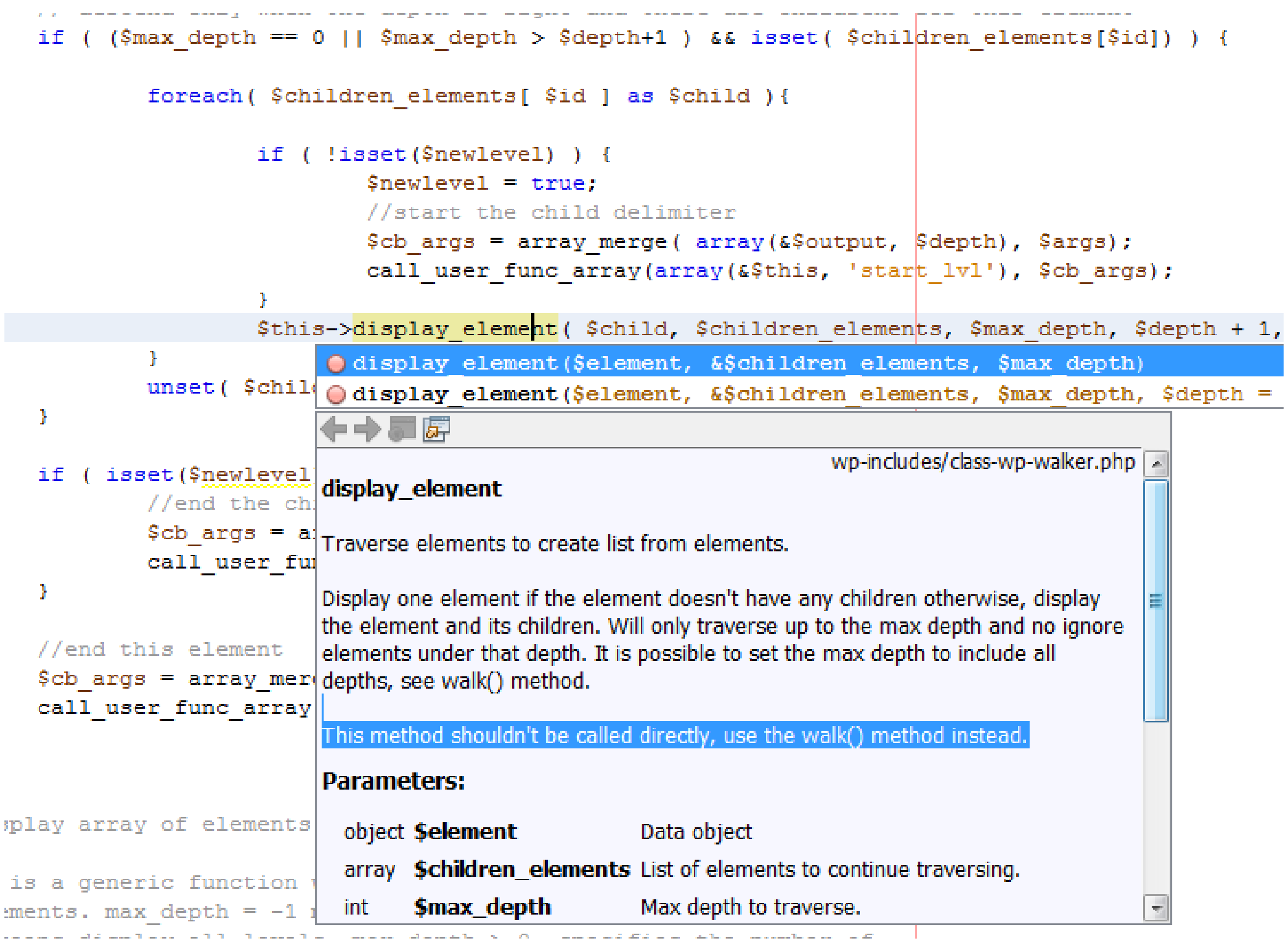
```
<?php
/**
 * A class for displaying various tree-like structures.
 *
 * Extend the Walker class to use it, see examples at the below. Child classes
 * do not need to implement all of the abstract methods in the class. The child
 * only needs to implement the methods that are needed. Also, the methods are
 * not strictly abstract in that the parameter definition needs to be followed.
 * The child classes can have additional parameters.
 *
 * @package WordPress
 * @since 2.1.0
 * @abstract
 */
class Walker {
        /**
         * What the class handles.
         *
         * @since 2.1.0
         * @var string
         * @access public
         */
        var $tree_type;

        /**
         * DB fields to use.
         *
         * @since 2.1.0
         * @var array
         * @access protected
         */
```

Commenting



```
/**
 * Traverse elements to create list from elements.
 *
 * Display one element if the element doesn't have any children otherwise,
 * display the element and its children. Will only traverse up to the max
 * depth and no ignore elements under that depth. It is possible to set the
 * max depth to include all depths, see walk() method.
 *
 * This method shouldn't be called directly, use the walk() method instead.
 *
 * @since 2.5.0
 *
 * @param object $element Data object
 * @param array $children_elements List of elements to continue traversing.
 * @param int $max_depth Max depth to traverse.
 * @param int $depth Depth of current element.
 * @param array $args
 * @param string $output Passed by reference. Used to append additional content.
 * @return null Null on failure with no changes to parameters.
 */
function display_element( $element, &$children_elements, $max_depth, $depth=0, $args, &$output ) {

        if ( !$element )
                return;

        $id_field = $this->db_fields['id'];
```

Commenting



```
if ( ($max_depth == 0 || $max_depth > $depth+1 ) && isset( $children_elements[$id]) ) {

        foreach( $children_elements[ $id ] as $child ){

                if ( !isset($newlevel) ) {
                        $newlevel = true;
                        //start the child delimiter
                        $cb_args = array_merge( array(&$output, $depth), $args);
                        call_user_func_array(array(&$this, 'start_lvl'), $cb_args);
                }
                $this->display_element( $child, $children_elements, $max_depth, $depth + 1,
        }
        unset( $chil
}

if ( isset($newlevel
        //end the ch
        $cb_args = a
        call_user_fu
}

//end this element
$cb_args = array_mer
call_user_func_array


play array of elements

 is a generic function
ments. max_depth = -1
```

```
display_element($element, &$children_elements, $max_depth)
display_element($element, &$children_elements, $max_depth, $depth =
```

wp-includes/class-wp-walker.php

display_element

Traverse elements to create list from elements.

Display one element if the element doesn't have any children otherwise, display the element and its children. Will only traverse up to the max depth and no ignore elements under that depth. It is possible to set the max depth to include all depths, see walk() method.

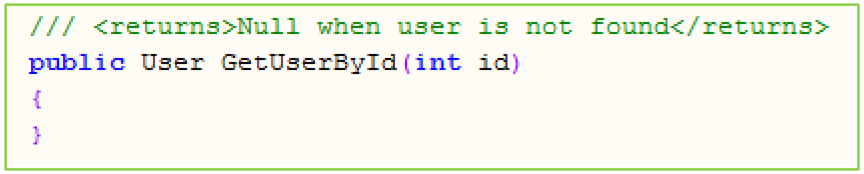
This method shouldn't be called directly, use the walk() method instead.

Parameters:

object	**$element**	Data object
array	**$children_elements**	List of elements to continue traversing.
int	**$max_depth**	Max depth to traverse.

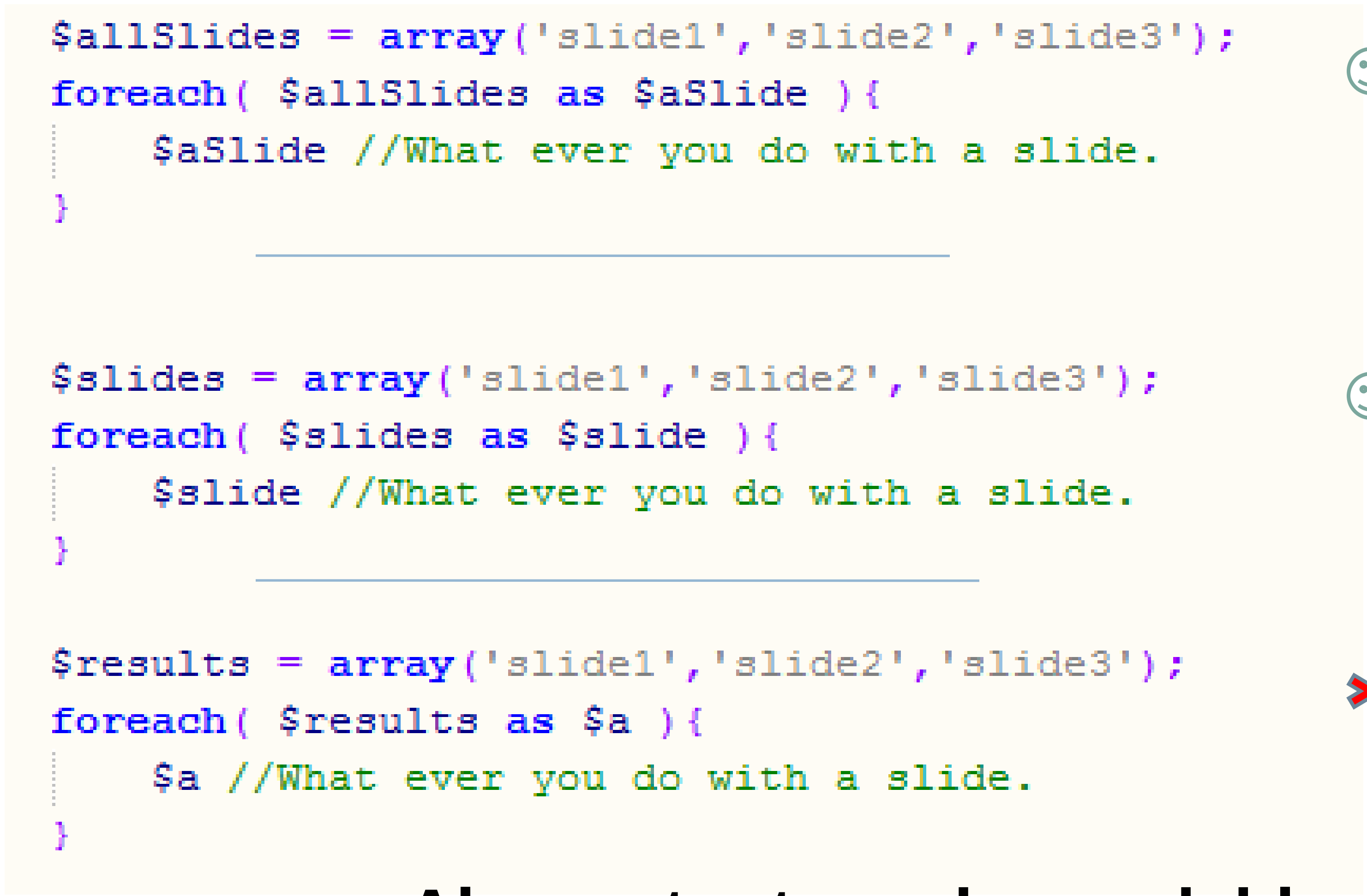
Self documenting code

- Use of Long Method Name that reflects the purpose of the method.
- Still It needs to be commented.



```
/// <returns>Null when user is not found</returns>
public User GetUserById(int id)
{
}
```

Code Readability



```
$allSlides = array('slide1','slide2','slide3');
foreach( $allSlides as $aSlide ){
    $aSlide //What ever you do with a slide.
}
```

☺

```
$slides = array('slide1','slide2','slide3');
foreach( $slides as $slide ){
    $slide //What ever you do with a slide.
}
```

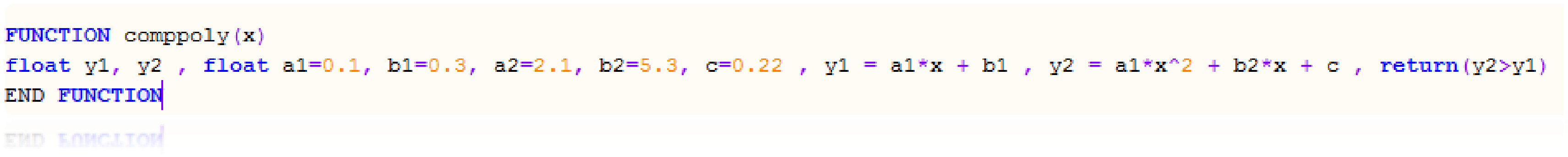
☺

```
$results = array('slide1','slide2','slide3');
foreach( $results as $a ){
    $a //What ever you do with a slide.
}
```

✗

Always try to make readable code.

Readability Vs. Compression



```
FUNCTION compoly(x)
float y1, y2 , float a1=0.1, b1=0.3, a2=2.1, b2=5.3, c=0.22 , y1 = a1*x + b1 , y2 = a1*x^2 + b2*x + c , return(y2>y1)
END FUNCTION
```

YES!! I saved lots of bytes. Code is now compact.

There are lots of tools for making code compact. You don’t have to write in unreadable compact form.

Code consistency

- Let a project has 3 members.
- They watch this slide very carefully , and realized the importance of coding standard & best practices.
- Now, they are told to do the project perfectly.
- Each members uses his/her coding convention and submitted the project.
- What will be the output ?

Code consistency

- Always use same standard throughout a project.
- All members of a project must choose a fixed convention before starting a project.

Learn from Others

- **Don’t invent your own standard. All of the issues have already been debated to death by many others.**
- Use an established standard
  - • Minimize politics by choosing an external standard
  - • Choose a standard compatible with the libraries you use
  - • Use the standard as a requirement when outsourcing
- Stick to the standard you establish, don’t mix

How To Write Unmaintainable Code

Ensure a job for life ;-)

- Read This Site Carefully with negating every concept.
- http://thc.org/root/phun/unmaintain.html

References

- http://wiki.mozilla.org/WebDev:FrontendCodeStandards
- http://na.isobar.com/standards/
- http://en.wikibooks.org/wiki/C%2B%2B_Programming/Programming_Languages/C%2B%2B/Code/Style_Conventions
- http://en.wikipedia.org/wiki/Best_Coding_Practices
- http://codex.wordpress.org/WordPress_Coding_Standards
- http://drupal.org/coding-standards
- Java Code Conventions
- http://www.sitepoint.com/coding-standards/
- http://www.programming4scientists.com/2008/09/26/good-code-bad-code-an-example/
- http://thc.org/root/phun/unmaintain.html

Question ?

Thank You,
Happy Coding!