```php
<?php
/**
* Creates thumbnail images in the form of squares from the images in the
* current directory, the thumbnails retain their aspect ratio but have a
* specific background colour. The size of the square is either 132 (default)
* or passed into the script using the parameter of "size". The colour of the
* background defaults to #996699 (rgb(153,102,153)) or is passed in using the
* "colour" parameter.
* Thus: http://<domain name>/images/makeSquare.php?colour=FF0000&size=100
*               - would create 100px square thumbnails with a red background.
* @author Dominic Myers <drmsite.com>
*/
if (!isset($_GET['colour'])) {
    $_GET['colour'] = "996699";
}
if (!isset($_GET['size'])) {
    $_GET['size'] = 132;
}
$files = scandir(".");
foreach($files as $file) { // Work through the files in the directory.
    if ($file == ".") {    // Quit if pointer to current directory.
        continue;
    }
    if ($file == "..") {   // Quit if pointer to directory above.
        continue;
    }
    // This regular expression matches file names starting with:
    // <a number greater than 1><underscore><a hex colour>
    $regex = "/^[0-9]+[_][0-9a-fA-F]{6}/";
    // Quit if we find a previously generated image as we don't want to
    // create thumbnails of other thumbnails.
    if (preg_match($regex, $file)) {
        continue;
    }
    if (substr($file, -3, 3) == "php") { // Quit if a php file.
        continue;
    }
    // This is the probable future file name, in the form of:
    // <image size><underscore><colour><underscore><original image name>
    $probableName = $_GET['size']."_".$_GET['colour']."_".$file;
    if (in_array($probableName, $files)) { // Quit if previously generated.
        continue;
    }
    $sourceImage = imagecreatefromjpeg($file); // Create a jpg from image.
    $sourceWidth = imagesx($sourceImage); //  Get its width.
    $sourceHeight = imagesy($sourceImage); // Get its height.
    if ($sourceWidth == $sourceHeight) { // If the image is already square.
        $destinationWidth = $_GET['size'];
        $destinationHeight = $_GET['size'];
        $destinationOffsetX = 0;
        $destinationOffsetY = 0;
    }
```

```php
    // The next two conditions make use of some simple maths:
    // 1st: For example, to calculate the correct height for a landscape
    //      orientated image we need to calculate the ratio between the
    //      height and the width of the image and then multiply this ratio
    //      by the width we want the final image to be.
    // 2nd: For example: to calculate the position where the new landscape
    //      orientated image should be placed on the square we need to
    //      calculate an downwards offset from the (0,0) position (or top-
    //      left point of the square). We do this by subtracting the new
    //      height from the height of the square and then dividing the result
    //      by 2.
    // For Portrait orientated images the process is reversed.
    if ($sourceWidth > $sourceHeight) { // Landscape orientated.
        $destinationWidth = $_GET['size'];
        $destinationHeight = ($sourceHeight / $sourceWidth) * $_GET['size'];
        $destinationOffsetX = 0;
        $destinationOffsetY = round(($_GET['size'] - $destinationHeight) /
          2);
    }
    if ($sourceWidth < $sourceHeight) { // Portrait orientated.
        $destinationWidth = ($sourceWidth / $sourceHeight) * $_GET['size'];
        $destinationHeight = $_GET['size'];
        $destinationOffsetX = round(($_GET['size'] - $destinationWidth) /
          2);
        $destinationOffsetY = 0;
    }
    // Create the square background image.
    $destinationImage = imagecreatetruecolor($_GET['size'], $_GET['size']);
    // Allocate a colour to the square.
    $backgroundColor = imagecolorallocate($destinationImage,
      hexdec(substr($_GET['colour'], 0, 2)),  /* Red   */
      hexdec(substr($_GET['colour'], 2, 2)),  /* Blue  */
      hexdec(substr($_GET['colour'], 4, 2))); /* Green */
    // Fill the Box with that colour.
    imagefilledrectangle($destinationImage, 0, 0, ($_GET['size']-1),
      ($_GET['size']-1), $backgroundColor);
    // imagecopyresampled() copies a rectangular portion of one image to
    // another image, smoothly interpolating pixel values so that, in
    // particular, reducing the size of an image still retains a great deal
    // of clarity.
    imagecopyresampled($destinationImage, $sourceImage, $destinationOffsetX,
      $destinationOffsetY, 0, 0, $destinationWidth, $destinationHeight,
      $sourceWidth, $sourceHeight);
    // Save the result as the best quality.
    imagejpeg($destinationImage, $probableName, 100);
    // Clean up after ourselves.
    imagedestroy($sourceImage);
    imagedestroy($destinationImage);
}
?>
```