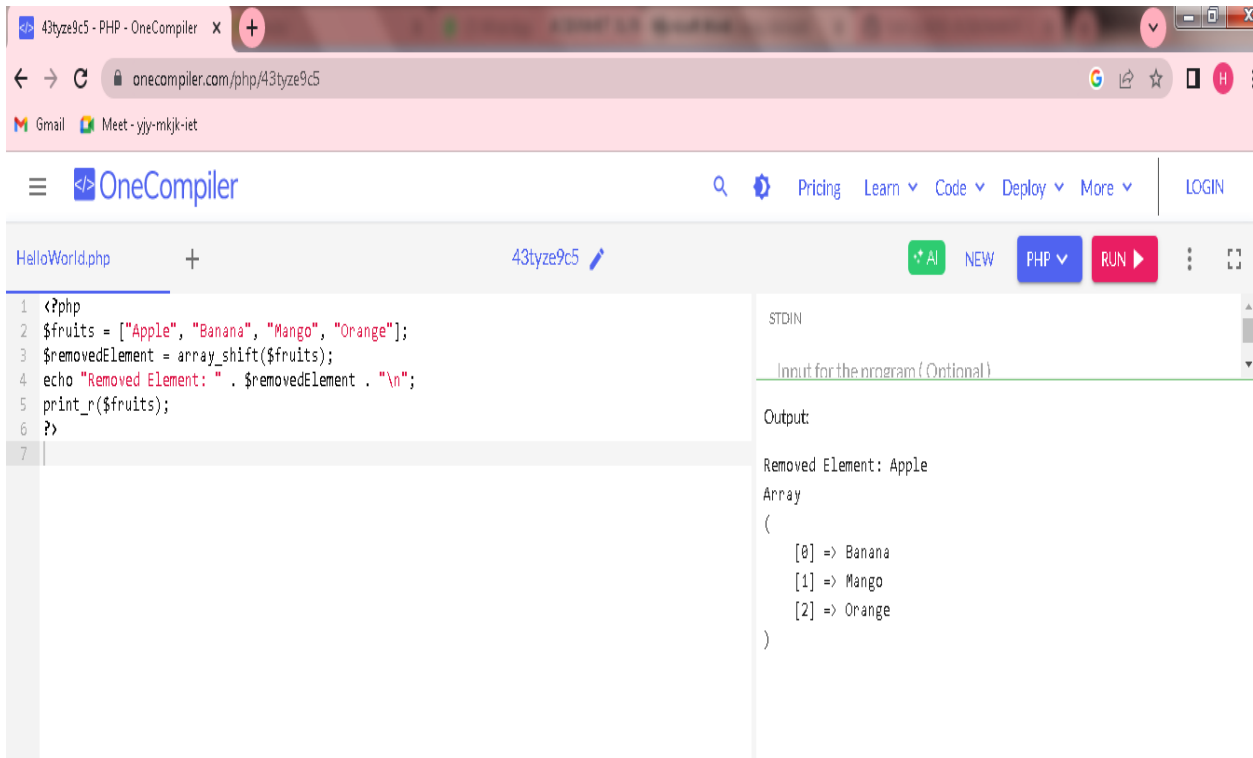


31)



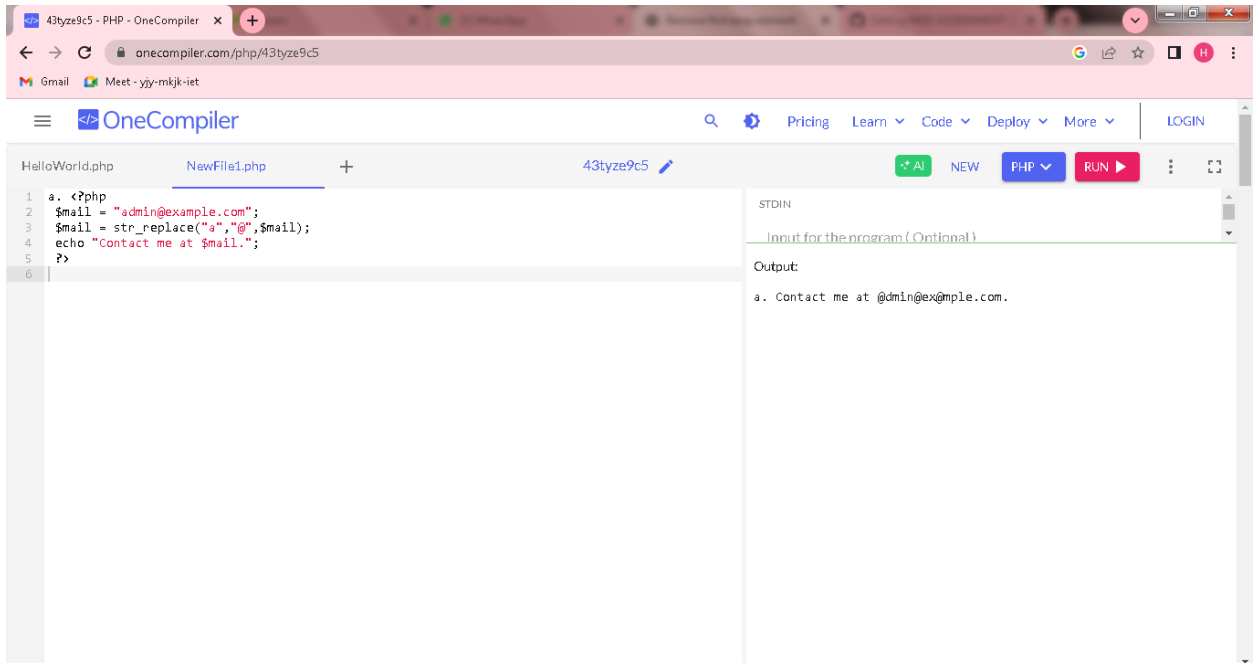
The screenshot shows the OneCompiler web interface. The browser address bar displays 'onecompiler.com/php/43tyze9c5'. The code editor contains a PHP script that initializes an array of fruits, removes the first element ('Apple'), and prints the remaining elements. The output panel shows the result of the execution.

```
1 <?php
2 $fruits = ["Apple", "Banana", "Mango", "Orange"];
3 $removedElement = array_shift($fruits);
4 echo "Removed Element: " . $removedElement . "\n";
5 print_r($fruits);
6 ?>
```

Output:

```
Removed Element: Apple
Array
(
    [0] => Banana
    [1] => Mango
    [2] => Orange
)
```

32)



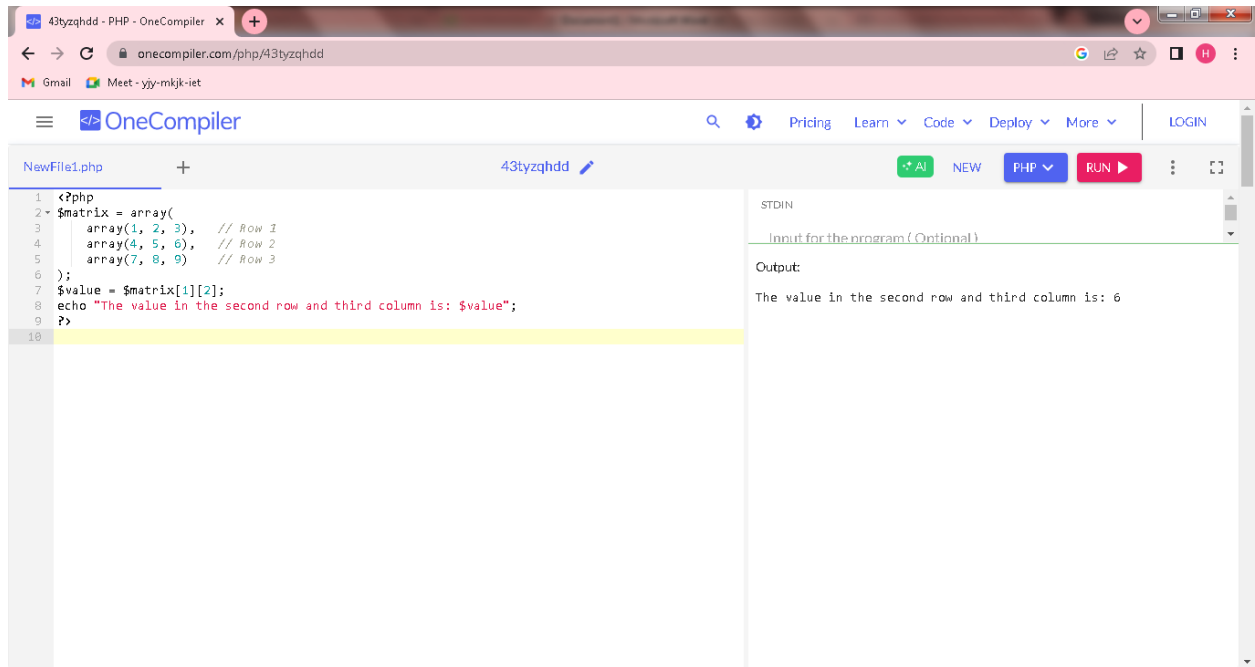
The screenshot shows the OneCompiler web interface with a new file named 'NewFile1.php'. The code editor contains a PHP script that replaces the '@' symbol in an email address with 'a' and then echoes the result. The output panel shows the result of the execution.

```
1 a. <?php
2 $mail = "admin@example.com";
3 $mail = str_replace("a","@",$mail);
4 echo "Contact me at $mail.";
5 ?>
```

Output:

```
a. Contact me at @admin@example.com.
```

33)



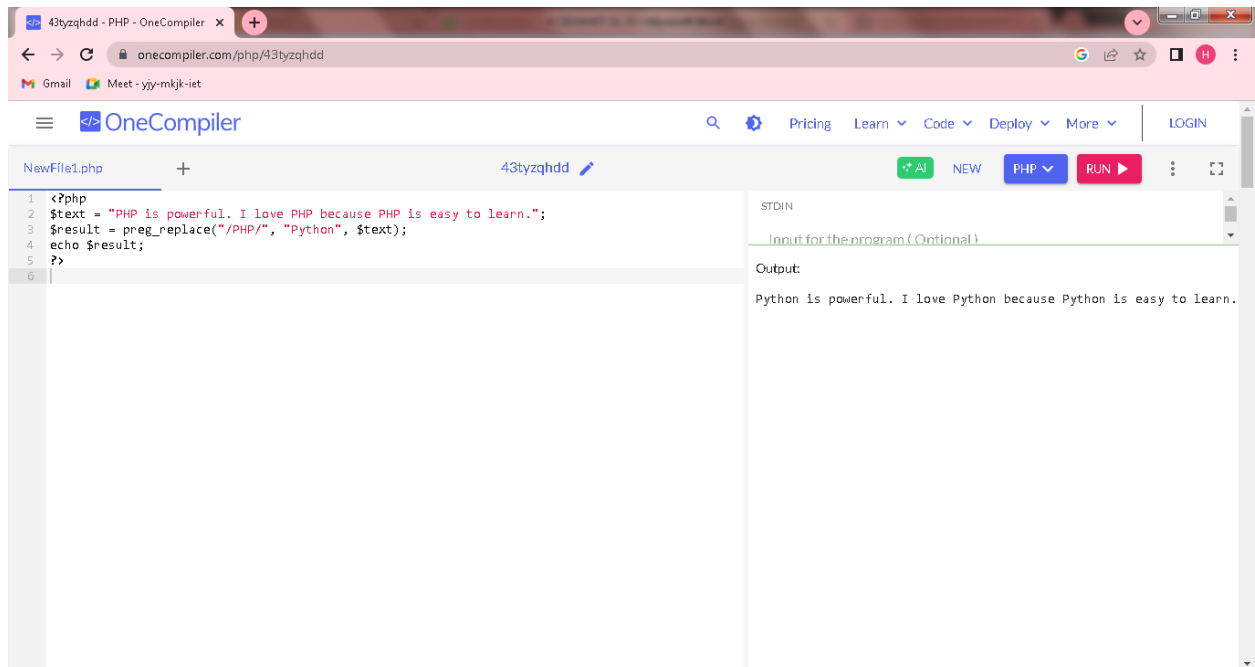
The screenshot shows the OneCompiler web interface. The code editor contains the following PHP code:

```
1 <?php
2 $matrix = array(
3     array(1, 2, 3), // Row 1
4     array(4, 5, 6), // Row 2
5     array(7, 8, 9) // Row 3
6 );
7 $value = $matrix[1][2];
8 echo "The value in the second row and third column is: $value";
9 ?>
```

The output panel on the right shows the result of the execution:

Output:
The value in the second row and third column is: 6

34)



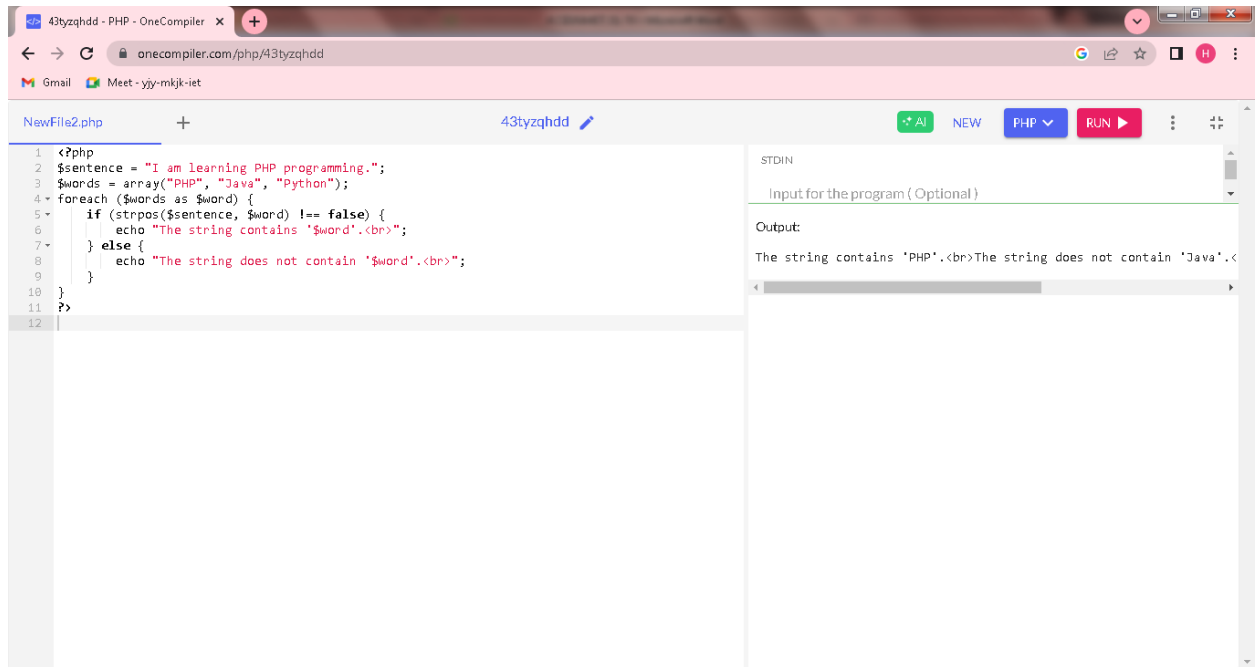
The screenshot shows the OneCompiler web interface. The code editor contains the following PHP code:

```
1 <?php
2 $text = "PHP is powerful. I love PHP because PHP is easy to learn.";
3 $result = preg_replace("/PHP/", "Python", $text);
4 echo $result;
5 ?>
```

The output panel on the right shows the result of the execution:

Output:
Python is powerful. I love Python because Python is easy to learn.

35)

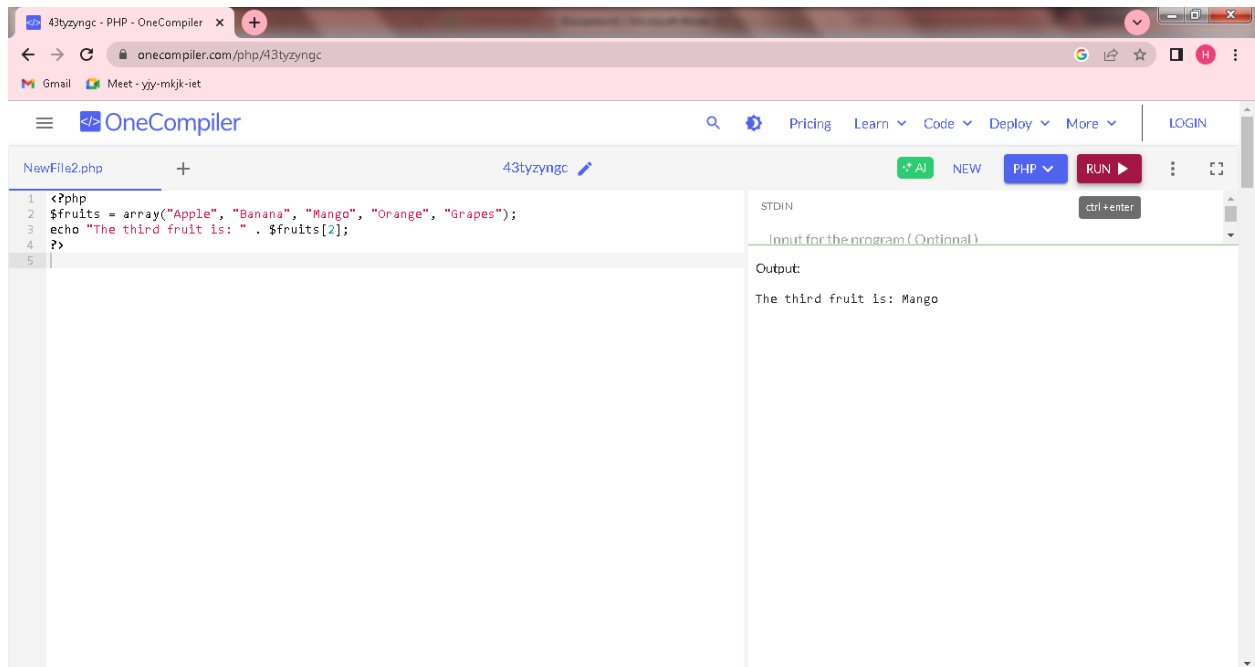


The screenshot shows the OneCompiler web interface for PHP. The code editor contains a PHP script that checks if a sentence contains any words from a given array. The output shows that the sentence contains 'PHP' but not 'Java'.

```
<?php
1 $sentence = "I am learning PHP programming.";
2 $words = array("PHP", "Java", "Python");
3
4 foreach ($words as $word) {
5     if (strpos($sentence, $word) !== false) {
6         echo "The string contains '$word'.<br>";
7     } else {
8         echo "The string does not contain '$word'.<br>";
9     }
10 }
11 ?>
```

Output:
The string contains 'PHP'.
The string does not contain 'Java'.

36)

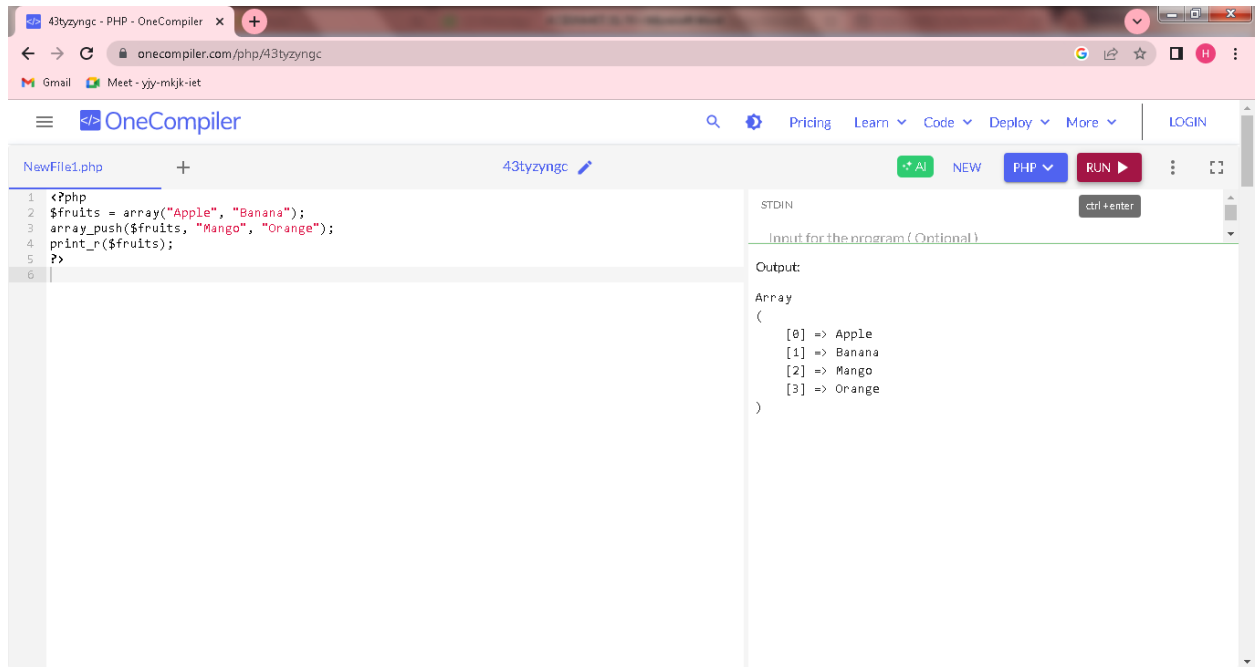


The screenshot shows the OneCompiler web interface for PHP. The code editor contains a PHP script that prints the third element of an array of fruits. The output shows 'Mango'.

```
<?php
1 $fruits = array("Apple", "Banana", "Mango", "Orange", "Grapes");
2
3 echo "The third fruit is: " . $fruits[2];
4 ?>
```

Output:
The third fruit is: Mango

37)



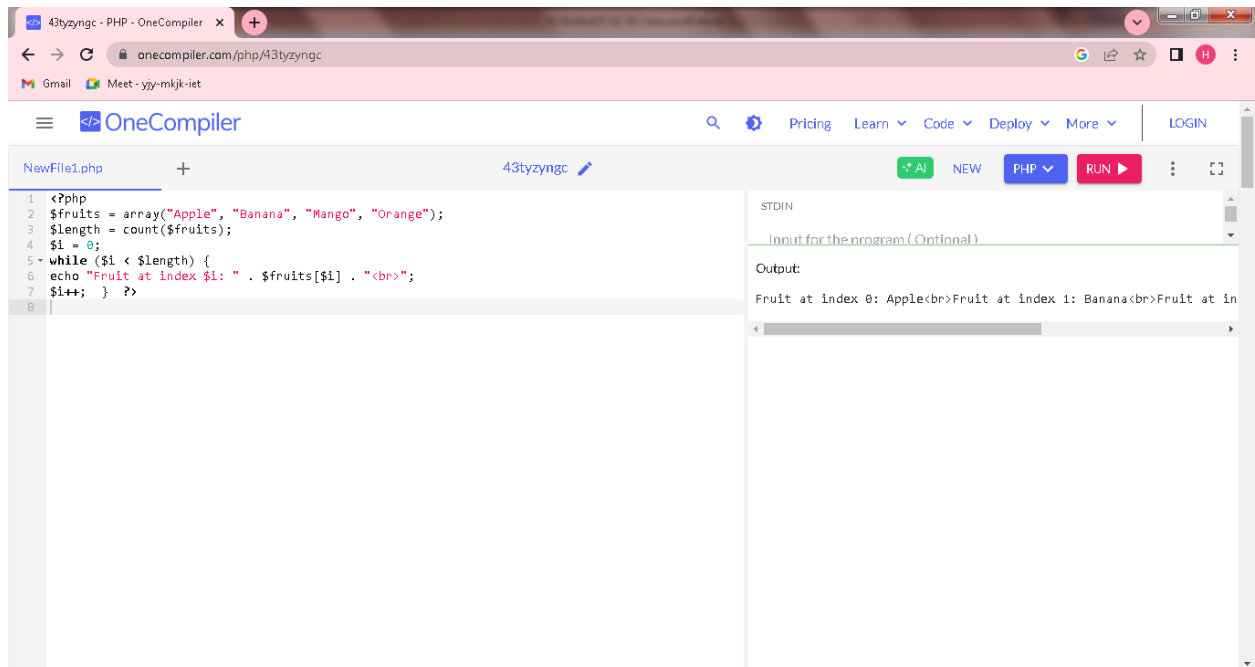
The screenshot shows the OneCompiler web interface. The code editor on the left contains the following PHP code:

```
1 <?php
2 $fruits = array("Apple", "Banana");
3 array_push($fruits, "Mango", "Orange");
4 print_r($fruits);
5 ?>
6
```

The right-hand panel shows the execution results. Under the 'Output:' section, it displays the output of the `print_r` function:

```
Array
(
    [0] => Apple
    [1] => Banana
    [2] => Mango
    [3] => Orange
)
```

38)



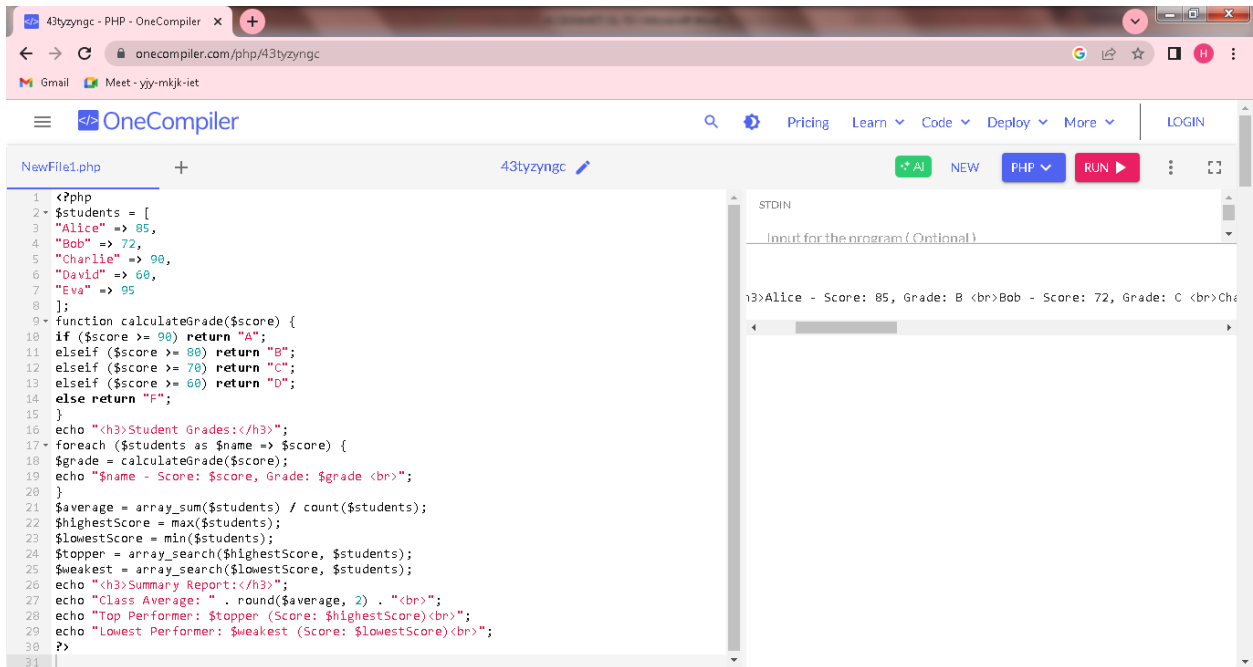
The screenshot shows the OneCompiler web interface with a different PHP code example. The code editor on the left contains the following PHP code:

```
1 <?php
2 $fruits = array("Apple", "Banana", "Mango", "Orange");
3 $length = count($fruits);
4 $i = 0;
5 while ($i < $length) {
6     echo "Fruit at index $i: " . $fruits[$i] . "<br>";
7     $i++; } ?>
8
```

The right-hand panel shows the execution results. Under the 'Output:' section, it displays the output of the `while` loop:

```
Fruit at index 0: Apple<br>Fruit at index 1: Banana<br>Fruit at in
```

39)



The screenshot shows the OneCompiler web interface with a PHP file named 'NewFile1.php'. The code defines an array of student names and scores, a function to calculate grades based on scores, and logic to calculate class statistics like average, highest, and lowest scores. The output on the right shows the calculated grades for each student.

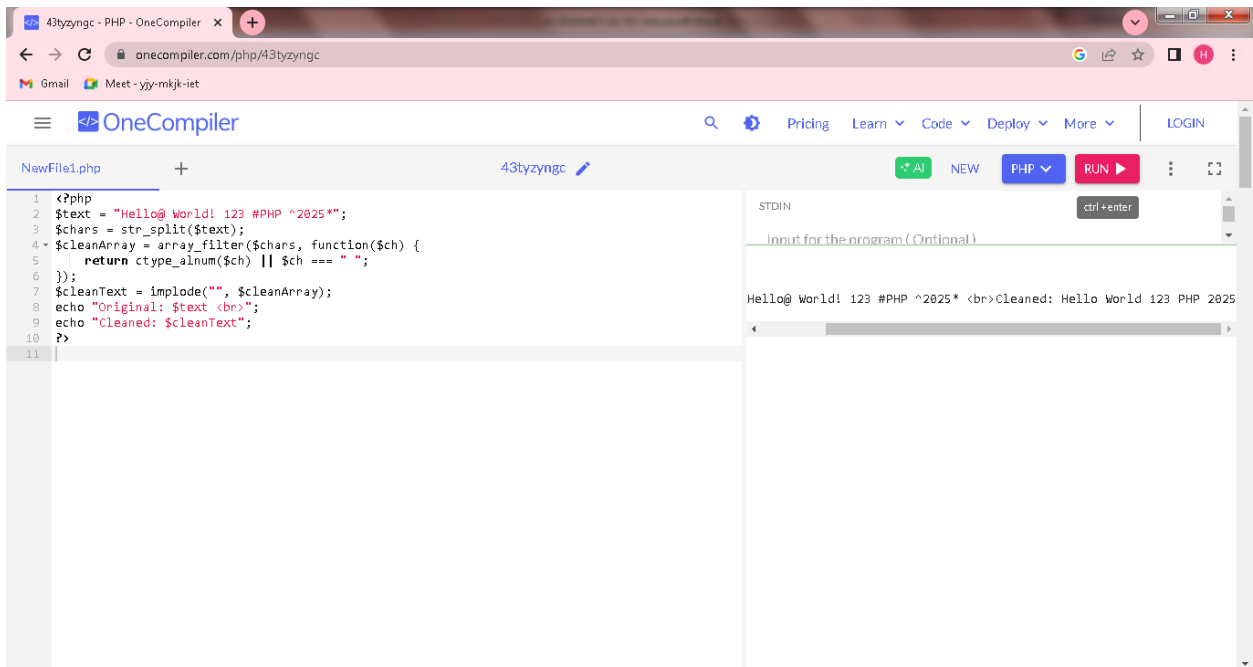
```
1 <?php
2 $students = [
3     "Alice" => 85,
4     "Bob" => 72,
5     "Charlie" => 90,
6     "David" => 60,
7     "Eva" => 95
8 ];
9 function calculateGrade($score) {
10     if ($score >= 90) return "A";
11     elseif ($score >= 80) return "B";
12     elseif ($score >= 70) return "C";
13     elseif ($score >= 60) return "D";
14     else return "F";
15 }
16 echo "<h3>Student Grades:</h3>";
17 foreach ($students as $name => $score) {
18     $grade = calculateGrade($score);
19     echo "$name - Score: $score, Grade: $grade <br>";
20 }
21 $average = array_sum($students) / count($students);
22 $highestScore = max($students);
23 $lowestScore = min($students);
24 $topper = array_search($highestScore, $students);
25 $weakest = array_search($lowestScore, $students);
26 echo "<h3>Summary Report:</h3>";
27 echo "Class Average: " . round($average, 2) . "<br>";
28 echo "Top Performer: $topper (Score: $highestScore)<br>";
29 echo "Lowest Performer: $weakest (Score: $lowestScore)<br>";
30 ?>
```

Output:

```
STDIN
Input for the program (Optional)

3> Alice - Score: 85, Grade: B <br> Bob - Score: 72, Grade: C <br> Charlie - Score: 90, Grade: A <br> David - Score: 60, Grade: D <br> Eva - Score: 95, Grade: A
```

40)



The screenshot shows the OneCompiler web interface with a PHP file named 'NewFile1.php'. The code takes a string with special characters and spaces, splits it into an array, filters out non-alphanumeric characters, and then joins the array back into a string. The output shows the original and cleaned versions of the string.

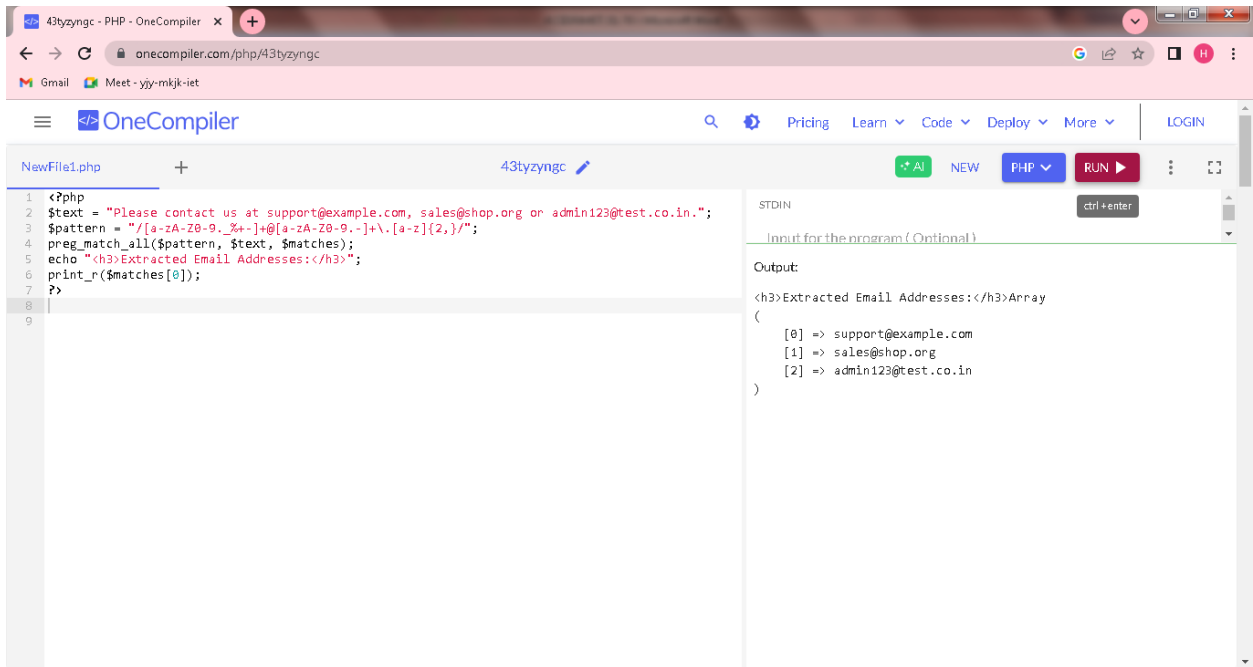
```
1 <?php
2 $text = "Hello@ World! 123 #PHP ^2025*";
3 $chars = str_split($text);
4 $cleanArray = array_filter($chars, function($ch) {
5     return ctype_alnum($ch) || $ch === " ";
6 });
7 $cleanText = implode(" ", $cleanArray);
8 echo "Original: $text <br>";
9 echo "Cleaned: $cleanText";
10 ?>
```

Output:

```
STDIN
Input for the program (Optional)

Hello@ World! 123 #PHP ^2025* <br> Cleaned: Hello World 123 PHP 2025
```

41)



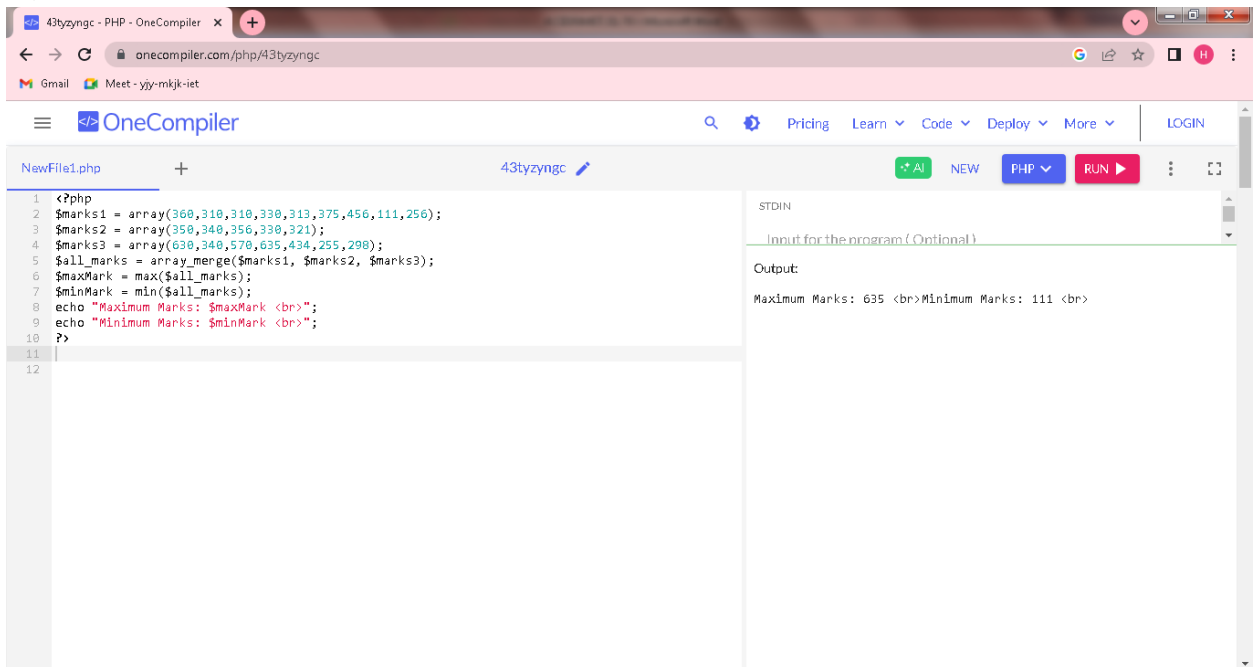
The screenshot shows the OneCompiler web interface with a PHP file named 'NewFile1.php'. The code is a PHP script that uses a regular expression to extract email addresses from a given text. The text is: "Please contact us at support@example.com, sales@shop.org or admin123@test.co.in.". The script uses `preg_match_all` to find all matches of the pattern `/[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-z]{2,}/`. The output is displayed in the right-hand pane, showing the extracted email addresses as an array: `[0] => support@example.com`, `[1] => sales@shop.org`, and `[2] => admin123@test.co.in`.

```
1 <?php
2 $text = "Please contact us at support@example.com, sales@shop.org or admin123@test.co.in.";
3 $pattern = "/[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-z]{2,}/";
4 preg_match_all($pattern, $text, $matches);
5 echo "<h3>Extracted Email Addresses:</h3>";
6 print_r($matches[0]);
7 ?>
```

Output:

```
<h3>Extracted Email Addresses:</h3>Array
(
    [0] => support@example.com
    [1] => sales@shop.org
    [2] => admin123@test.co.in
)
```

42)



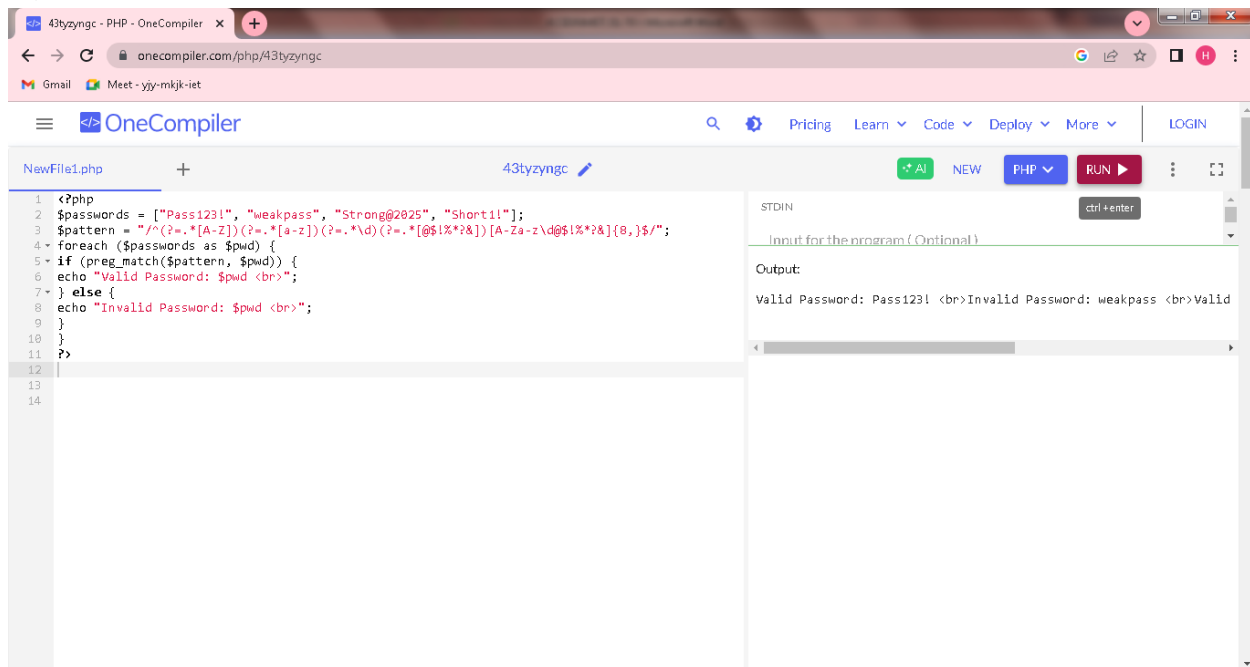
The screenshot shows the OneCompiler web interface with a PHP file named 'NewFile1.php'. The code is a PHP script that manipulates arrays. It defines three arrays: `$marks1`, `$marks2`, and `$marks3`. `$marks1` is `array(360,310,310,330,313,375,456,111,256)`, `$marks2` is `array(350,340,356,330,321)`, and `$marks3` is `array(630,340,570,635,434,255,290)`. The script then merges these arrays into `$all_marks` using `array_merge`. It calculates the maximum mark (`$maxMark`) and the minimum mark (`$minMark`) from the merged array. The output is displayed in the right-hand pane, showing the maximum mark as 635 and the minimum mark as 111.

```
1 <?php
2 $marks1 = array(360,310,310,330,313,375,456,111,256);
3 $marks2 = array(350,340,356,330,321);
4 $marks3 = array(630,340,570,635,434,255,290);
5 $all_marks = array_merge($marks1, $marks2, $marks3);
6 $maxMark = max($all_marks);
7 $minMark = min($all_marks);
8 echo "Maximum Marks: $maxMark <br>";
9 echo "Minimum Marks: $minMark <br>";
10 ?>
```

Output:

```
Maximum Marks: 635 <br>Minimum Marks: 111 <br>
```

43)

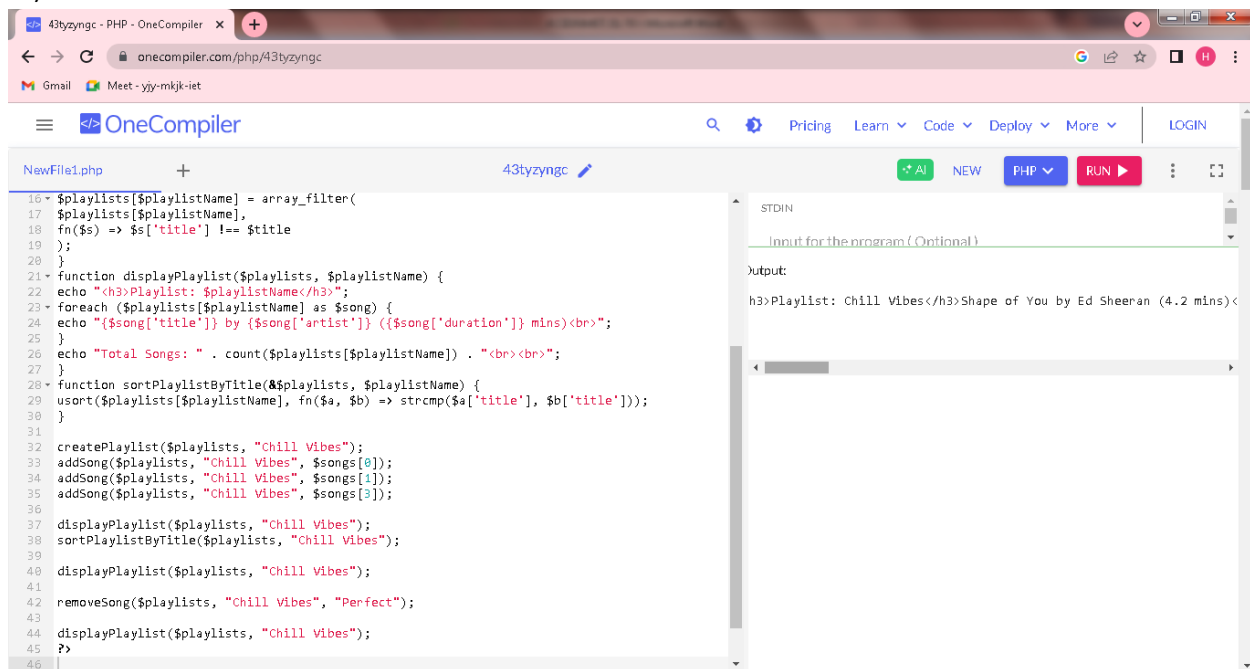


The screenshot shows the OneCompiler web interface with a PHP file named 'NewFile1.php'. The code is a password validation script. It defines an array of passwords, a regex pattern, and a loop that checks each password against the pattern. The output shows 'Valid Password: Pass123!' followed by 'Invalid Password: weakpass' and 'Valid'.

```
1 <?php
2 $passwords = ["Pass123!", "weakpass", "Strong@2025", "Short1!"];
3 $pattern = "/^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/";
4 foreach ($passwords as $pwd) {
5     if (preg_match($pattern, $pwd)) {
6         echo "Valid Password: $pwd <br>";
7     } else {
8         echo "Invalid Password: $pwd <br>";
9     }
10 }
11 ?>
```

Output:
Valid Password: Pass123!
Invalid Password: weakpass
Valid

44)



The screenshot shows the OneCompiler web interface with a PHP file named 'NewFile1.php'. The code is a playlist management script. It defines a function to display a playlist, a function to sort songs by title, and a function to create a playlist. It then creates a playlist named 'Chill Vibes', adds three songs, displays the playlist, sorts it by title, displays it again, and removes a song named 'Perfect'. The output shows the playlist name and the list of songs.

```
10 $playlists[$playlistName] = array_filter(
11     $playlists[$playlistName],
12     fn($s) => $s['title'] !== $title
13 );
14 }
15 function displayPlaylist($playlists, $playlistName) {
16     echo "<h3>Playlist: $playlistName</h3>";
17     foreach ($playlists[$playlistName] as $song) {
18         echo "{$song['title']} by {$song['artist']} ({$song['duration']} mins)<br>";
19     }
20     echo "Total Songs: " . count($playlists[$playlistName]) . "<br><br>";
21 }
22 function sortPlaylistByTitle(&$playlists, $playlistName) {
23     usort($playlists[$playlistName], fn($a, $b) => strcmp($a['title'], $b['title']));
24 }
25 createPlaylist($playlists, "Chill Vibes");
26 addSong($playlists, "Chill Vibes", $songs[0]);
27 addSong($playlists, "Chill Vibes", $songs[1]);
28 addSong($playlists, "Chill Vibes", $songs[2]);
29 displayPlaylist($playlists, "Chill Vibes");
30 sortPlaylistByTitle($playlists, "Chill Vibes");
31 displayPlaylist($playlists, "Chill Vibes");
32 removeSong($playlists, "Chill Vibes", "Perfect");
33 displayPlaylist($playlists, "Chill Vibes");
34 ?>
```

Output:
h3>Playlist: Chill Vibes</h3>Shape of You by Ed Sheeran (4.2 mins)<

45)

The screenshot shows the OneCompiler PHP editor interface. The code in `NewFile1.php` defines a recursive function `array_diff_recursive` that compares two arrays. It then calls this function with two arrays: `$array1` (containing user details) and `$array2` (empty). The output on the right shows the result of the recursive comparison.

```

1 <?php
2 function array_diff_recursive($array1, $array2) {
3     $result = [];
4
5     foreach ($array1 as $key => $value) {
6         if (is_array($value)) {
7             if (!isset($array2[$key]) || !is_array($array2[$key])) {
8                 $result[$key] = $value;
9             } else {
10                 $diff = array_diff_recursive($value, $array2[$key]);
11                 if (!empty($diff)) {
12                     $result[$key] = $diff;
13                 }
14             }
15         } else {
16             if (!isset($array2[$key]) || $array2[$key] !== $value) {
17                 $result[$key] = $value;
18             }
19         }
20     }
21
22     return $result;
23 }
24
25 $array1 = [
26     "name" => "Alice",
27     "age" => 25,
28     "skills" => ["PHP", "MySQL", "JavaScript"],
29     "city" => "New York"
30 ];
31 $array2 = [

```

Output:

```

<pre>Array
(
    [age] => 25
    [skills] => Array
        (
            [2] => JavaScript
        )
    [city] => New York
)
</pre>

```

46)

The screenshot shows the OneCompiler PHP editor interface. The code in `NewFile1.php` defines an `array_search` function to find the index of a value in an array. It then calls this function with the value "Mango" and an array of fruits. The output on the right shows the result of the search.

```

1 <?php
2 $fruits = array("Apple", "Banana", "Mango", "Orange", "Grapes");
3 $search = "Mango";
4 $index = array_search($search, $fruits);
5 if ($index !== false) {
6     echo "The value '$search' is found at index: $index";
7 } else {
8     echo "The value '$search' was not found in the array.";
9 }
10 ?>
11

```

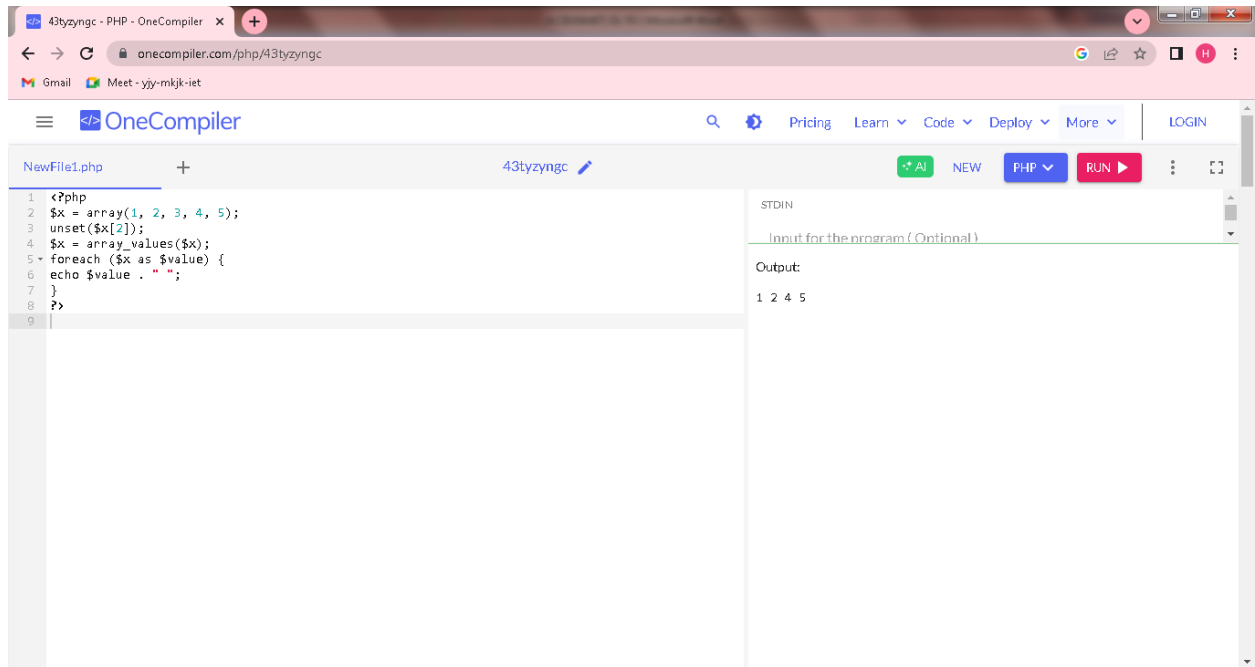
Output:

```

The value 'Mango' is found at index: 2

```


47)

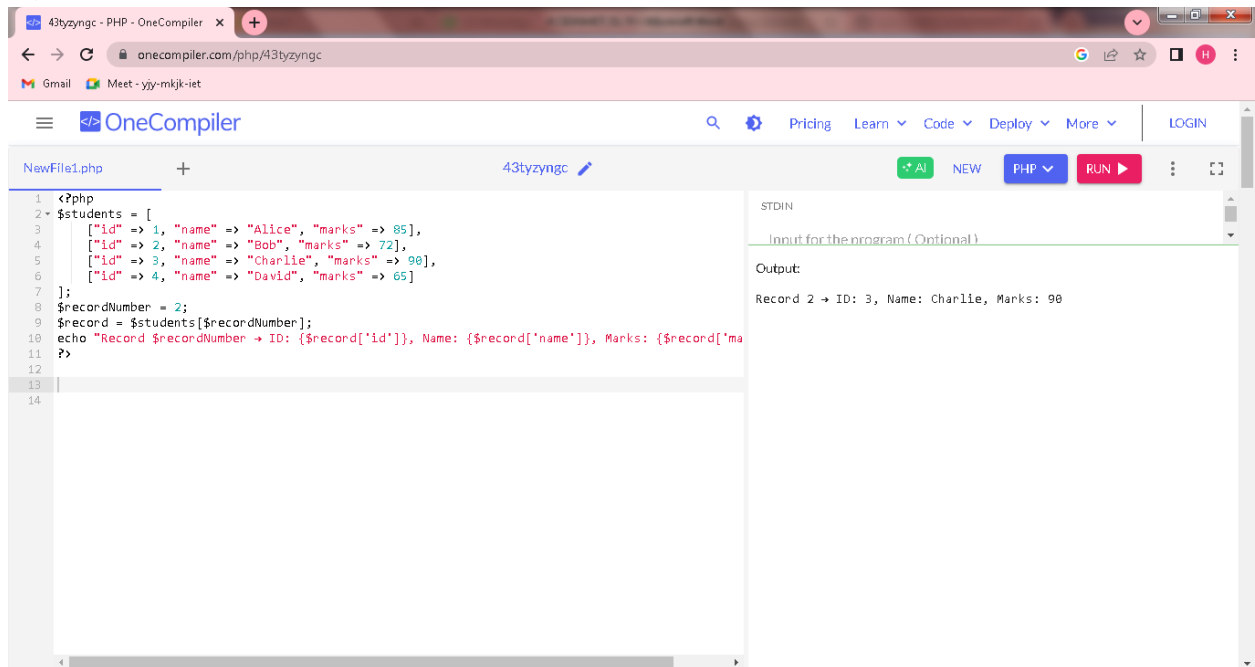


The screenshot shows the OneCompiler web interface. The code editor contains a PHP script that creates an array, unsets an element, and prints the remaining values. The output pane shows the result of the script execution.

```
1 <?php
2 $x = array(1, 2, 3, 4, 5);
3 unset($x[2]);
4 $x = array_values($x);
5 foreach ($x as $value) {
6     echo $value . " ";
7 }
8 ?>
```

Output:
1 2 4 5

48)



The screenshot shows the OneCompiler web interface. The code editor contains a PHP script that defines an array of student records, selects a specific record, and prints its details. The output pane shows the result of the script execution.

```
1 <?php
2 $students = [
3     ["id" => 1, "name" => "Alice", "marks" => 85],
4     ["id" => 2, "name" => "Bob", "marks" => 72],
5     ["id" => 3, "name" => "Charlie", "marks" => 90],
6     ["id" => 4, "name" => "David", "marks" => 65]
7 ];
8 $recordNumber = 2;
9 $record = $students[$recordNumber];
10 echo "Record $recordNumber -> ID: {$record['id']}, Name: {$record['name']}, Marks: {$record['ma'
11 ?>
```

Output:
Record 2 -> ID: 3, Name: Charlie, Marks: 90

43tyzngc - PHP - OneCompiler

onecompiler.com/php/43tyzngc

OneCompiler

NewFile1.php

```
1 <?php
2
3 $students = [
4     ["id" => 1, "name" => "Alice", "marks" => 85],
5     ["id" => 2, "name" => "Bob", "marks" => 72],
6     ["id" => 3, "name" => "Charlie", "marks" => 98],
7     ["id" => 4, "name" => "David", "marks" => 65]
8 ];
9
10 foreach ($students as $index => $student) {
11     echo "Record $index: {"$student['id']} - {"$student['name']} - {"$student['marks']} <br>";
12 }
13 ?>
```

STDIN

Input for the program (Optional)

Output:

Record 0: 1 - Alice - 85
Record 1: 2 - Bob - 72
Record 2:

49)

43tyzngc - PHP - OneCompiler

onecompiler.com/php/43tyzngc

OneCompiler

NewFile1.php

```
1 <?php
2
3 $players = [
4     ["name" => "Alice", "matches" => 10, "goals" => 15, "assists" => 5],
5     ["name" => "Bob", "matches" => 12, "goals" => 10, "assists" => 7],
6     ["name" => "Charlie", "matches" => 8, "goals" => 12, "assists" => 6],
7     ["name" => "David", "matches" => 15, "goals" => 20, "assists" => 10]
8 ];
9
10 foreach ($players as $player) {
11     // Example formula: (Goals * 4 + Assists * 2) / Matches
12     $player['performance_index'] = round(
13         ((($player['goals'] * 4) + ($player['assists'] * 2)) / $player['matches']), 2
14     );
15 }
16
17 usort($players, function($a, $b) {
18     return $b['performance_index'] <=> $a['performance_index'];
19 });
20
21 $totalGoals = array_sum(array_column($players, 'goals'));
22 $totalAssists = array_sum(array_column($players, 'assists'));
23 $totalMatches = array_sum(array_column($players, 'matches'));
24 $avgGoals = round($totalGoals / count($players), 2);
25 $avgAssists = round($totalAssists / count($players), 2);
26
27 echo "<h3>Player Rankings:</h3>";
28 foreach ($players as $rank => $player) {
29     echo ($rank+1) . ". {"$player['name']} - Performance Index: {"$player['performance_index']}<br>";
30 }
31
32 echo "<h3>Team Averages:</h3>";
33 echo "Average Goals per Player: $avgGoals<br>";
34 echo "Average Assists per Player: $avgAssists<br>";
```

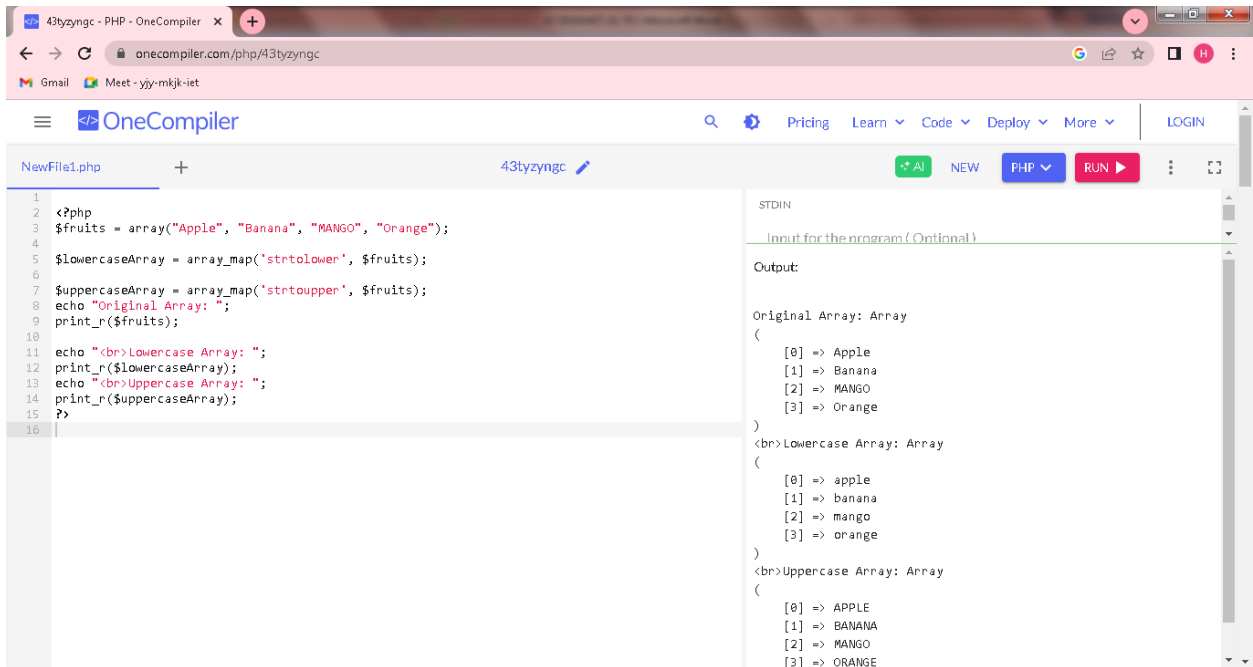
STDIN

Input for the program (Optional)

Output:

3>Player Rankings:</h3>1. Charlie - Performance Index: 7.5
2. Alice - Performance Index: 7.0
3. Bob - Performance Index: 6.7
4. David - Performance Index: 5.3

50)



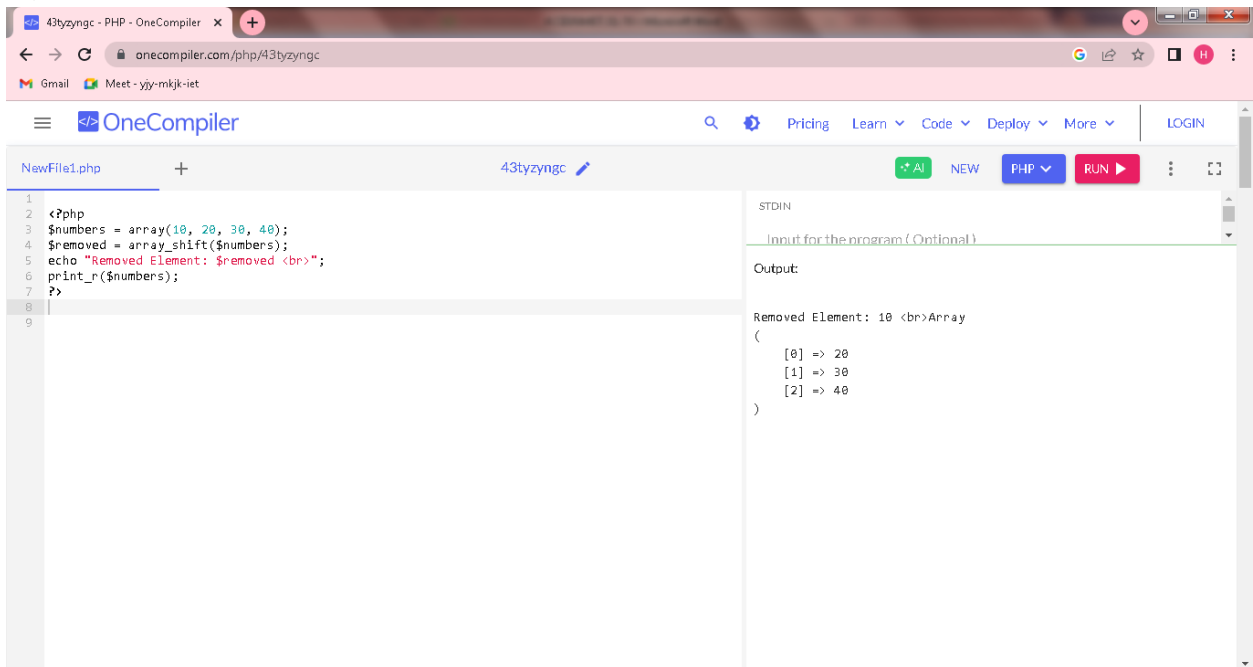
The screenshot shows the OneCompiler web interface with a PHP file named 'NewFile1.php'. The code defines an array of fruits and uses `array_map` to create lowercase and uppercase versions. The output panel shows the original array, the lowercase array, and the uppercase array.

```
1 <?php
2 $fruits = array("Apple", "Banana", "MANGO", "Orange");
3
4 $lowercaseArray = array_map('strtolower', $fruits);
5
6 $uppercaseArray = array_map('strtoupper', $fruits);
7 echo "Original Array: ";
8 print_r($fruits);
9
10 echo "<br>Lowercase Array: ";
11 print_r($lowercaseArray);
12 echo "<br>Uppercase Array: ";
13 print_r($uppercaseArray);
14
15 ?>
```

Output:

```
Original Array: Array
(
    [0] => Apple
    [1] => Banana
    [2] => MANGO
    [3] => Orange
)
<br>Lowercase Array: Array
(
    [0] => apple
    [1] => banana
    [2] => mango
    [3] => orange
)
<br>Uppercase Array: Array
(
    [0] => APPLE
    [1] => BANANA
    [2] => MANGO
    [3] => ORANGE
)
```

51)



The screenshot shows the OneCompiler web interface with a PHP file named 'NewFile1.php'. The code defines an array of numbers and uses `array_shift` to remove the first element. The output panel shows the removed element and the resulting array.

```
1 <?php
2 $numbers = array(10, 20, 30, 40);
3 $removed = array_shift($numbers);
4 echo "Removed Element: $removed <br>";
5 print_r($numbers);
6
7 ?>
```

Output:

```
Removed Element: 10 <br>Array
(
    [0] => 20
    [1] => 30
    [2] => 40
)
```

43tyzngc - PHP - OneCompiler

onecompiler.com/php/43tyzngc

OneCompiler

NewFile1.php

43tyzngc

```
1 <?php
2 $numbers = array(10, 20, 30, 40);
3 $newCount = array_unshift($numbers, 5, 1);
4 echo "New Count: $newCount <br>";
5 print_r($numbers);
6 ?>
```

STDIN

ctrl+enter

Input for the program (Optional)

Output:

New Count: 6
Array

```
(
  [0] => 5
  [1] => 10
  [2] => 20
  [3] => 30
  [4] => 40
)
```

52)

43tyzngc - PHP - OneCompiler

onecompiler.com/php/43tyzngc

OneCompiler

NewFile1.php

43tyzngc

```
1 <?php
2 $stack = array();
3 array_push($stack, "A");
4 array_push($stack, "B");
5 array_push($stack, "C");
6 echo "Stack after pushes: ";
7 print_r($stack);
8 $popped = array_pop($stack);
9 echo "<br>Popped Element: $popped";
10 echo "<br>Stack after pop: ";
11 print_r($stack);
12 ?>
```

STDIN

ctrl+enter

Input for the program (Optional)

Output:

Stack after pushes: Array

```
(
  [0] => A
  [1] => B
  [2] => C
)
```


Popped Element: C
Stack after pop: Array

```
(
  [0] => A
  [1] => B
)
```

53)