PHP

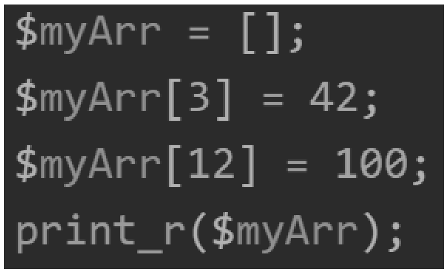
(Part 2)

ICS4U - Mr. Emmell

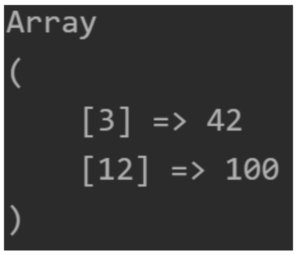
Array Indices!

- Arrays do not have to have sequential indices (0,1,2,etc...)

Ex:



```php
$myArr = [];
$myArr[3] = 42;
$myArr[12] = 100;
print_r($myArr);
```



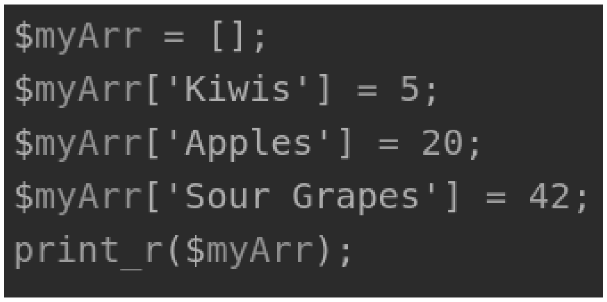
```
Array
(
    [3] => 42
    [12] => 100
)
```

There are NO OTHER indices other than these two.

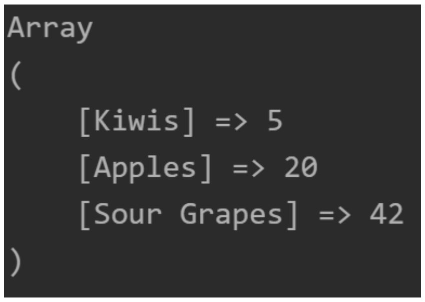
echo $myArr[0]; // would throw an error

Array Indices!

- Arrays do not even need NUMERIC indices!



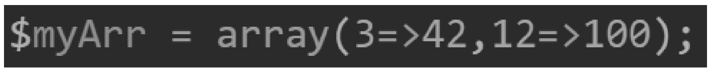
```
$myArr = [];
$myArr['Kiwis'] = 5;
$myArr['Apples'] = 20;
$myArr['Sour Grapes'] = 42;
print_r($myArr);
```



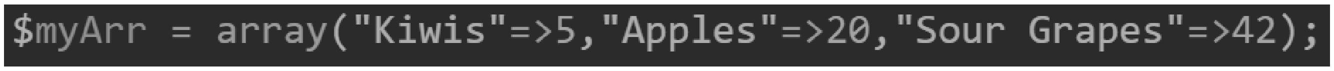
```
Array
(
    [Kiwis] => 5
    [Apples] => 20
    [Sour Grapes] => 42
)
```

Array Indices!

- Both of these can be defined by creating the array with key=>value pairs:
  - key refers to the index
  - value refers to the value at that index



```
$myArr = array(3=>42,12=>100);
```



```
$myArr = array("Kiwis"=>5,"Apples"=>20,"Sour Grapes"=>42);
```

Then how in the world do we iterate through them?

The wonderful 'foreach' loop

No matter what the indexes are, the foreach loop will iterate through the entire array.

```
foreach($arrayToInterateThrough as $eachItem) {
		echo $eachItem '';
}
```

The wonderful 'foreach' loop

You can even ask it to tell you what the indexes are:

```
Fruit Array
(
	[kiwi] => 20
	[oranges] => 12
	[apples] => 9
	[bananas] => 42
)
```

```
foreach($fruit as $kind => $quant) {
		echo "You have $quant $kind\n";
}
```

Output

```
You have 20 kiwi
You have 12 oranges
You have 9 apples
You have 42 bananas
```

The wonderful 'foreach' loop

If we have a multidimensional array like this:

```
$students....
array(
		[5] = array(
				[firstName] = "Jack"
				[lastName]  = "Sparrow"
				)
)
```

How do we iterate through it?

```
foreach($students as $stud) {
		echo $stud['firstName']." ".$stud['lastName']."\n";
}
```

Output

```
Jack Sparrow
```

The problem with quotes…

The problem with quotes…

How would we echo the following string:

```
<input type="number" name="var1" id="var1"/>
```

It uses double quotes! The echo function <u>also</u> uses double quotes…

```
echo "<input type="number" name="var1" id="var1"/>";
```

The problem with quotes...

How would we echo the following string:

```
<input type="number" name="var1" id="var1"/>
```

It uses double quotes! The echo function also uses double quotes...

echo **"<input type="**number" name="var1" id="var1"/>";

echo will think that it only needs to print out
<input type=
and then the rest will error.

Two solutions

- Escape the quotes you want to actually print out
  - `echo "<input type=\"number\" name=\"var1\" id=\"var1\"/>";`
- Switch to single quotes where possible
  - `echo "<input type='number' name='var1' id='var1'/>";`
- Sometimes you will need to use both approaches!

File I/O with PHP

Mostly the same as C, which is nice!

Process is the same:

- Open the file (fopen)
- Use that open file to read/write
- Close the file (fclose)

File I/O Examples:

```
$file = fopen("myfile.csv","r"); //only use "r" or "a" for
                                                     the upcoming assignment

// ... do some stuff


fclose($file);
```

File I/O Examples (fputcsv):

```
$student = array("Jack","Sparrow","53%");
fputcsv($file,$student);        //will write out array as one line
```

or

```
$students = array(
                    array("Jack","Sparrow","53%"),
                    array("Johnny","Appleseed","87%")
                    );
for ($x=0;$x<count($students);$x++) {
        fputcsv($file,$students[$x]);
}
```

File I/O Examples (fgetcsv):

How about reading?

```
$lineArr = fgetcsv($file);
```

or

```
while ($lineArr = fgetcsv($file)) {
        print_r($lineArr);
}
```