PHP Similar to Java

Within <?> brackets

```
$varname = value;
echo $varname; ← print
```

Arrays

- `$arr = [val, val2, val3];`
- `$first = $arr[0];`
- `$arr[3] = new_val;`
- `print_r($arr);` ← recursively prints array
- `count($arr);` ← tells length of arr
- `$first = reset($arr);` ← returns 1st item (resets iterator ptr)
- `$end = end($arr);` ← returns last item

Stacks / Queues - use arrays as stacks/qs

```
$stack = [1,2,3];
array_push($stack, value);
array_pop($stack);
array_merge($arr1, $arr2); ← concatenate
array_slice($arr, 3, 2) ← # of items
```

does not remove

array_splice [0,1,2,3,4,5] → gives 3,4

↑ removes SEGMENT

- Arrays actually are ordered maps
  - is keys & values
- $hash = ["key" => "value"...];
  - or
  - $hash["key"] = "value";
  - if (array_key_exists("key", "value") {..}
  - array_keys($hash); ← returns array of keys

Strings

$s = "string";

$s1 = "hello $s";

$s2 = $s . " " . "$s1";

↑ concatenates

strlen($s); ← len

$e2 = substr($s , strlen($s) - 3);

↑ only gives last 3 letters

$list = explode(" ", $word);

↑ basically split at " " → creates list

↑ used the

$word = implode(",", $list); opposite

for loops

- for ($i = 0; $i < count($arr); $i = $i + 1) { ... }
  ($i++ similar to c++)
- foreach ($array as $var) { echo $var }
- for each ($hash as $name => $val) { ... }

while loop

- while ($counter < 10) { ... }
- can use break/continue

functions

- function name ($param) { ... (no type)
    return $val;
  }
- name ($param_val);
- loading other php files:
  include("filename.php")
  : can call funct from file

Objects | Classes

- PHP object-oriented
- class Student {

```
    //constructor
    public function __construct($f_n, $l_n) {
        $this -> first_name = $f_n;
        $this -> last_name = $l_n;
    }
    public function say_name(){
        echo $this -> first_name . " " . $this ->
                                        last_name;
    }
}

$alex = new Student("Alex", "Jones");
$alex -> say_name();
```

- members - vars that belong to object

Inheritance

class MathStudent extends Student { ... }

(can use methods from)

- public - can use outside class
- private - cannot

Exceptions

- try {...
  } catch (Exception $e) {
    echo "Exception
  } finally {...} ← run after try/catch regardless

- if (some bool) {
    echo "Right" }
  else {
    throw new Exception("bad");
  }

causes fatal error