# Final Review

# Disclamer

- This review is an attempt to highlight the major parts of the semester
- Something being on this review does not garuntee it will be on the test
- Something not being on this review does not mean it will not be on the test

# Introduction

- What is a scripting language
  - How is it different than a compiled language
- What are some common uses for scripting languages

# Regular Expressions

- What are they
  - What are some uses of them?
- What is the syntax?
  - Character Class Symbols (`\w`, `\W`, etc.)
  - Dot character `.`
  - Quantifiers (`?`, `*`, `+`, `{n}`, `{n,}`, `{n,m}`)
    - Greedy and non-greedy
  - Anchors (`^`, `$`, `\b`, `\B`)

In [ ]:

```
%%perl
use feature qw(say);
$string = "a b c" ;
say $string if $string=~ /\w+/;
```

In [ ]:

```
%%perl
use feature qw(say);
$string = "abc" ;
say $string if $string=~ /\W+/;
$string = ",." ;
say $string if $string=~ /\W+/;
```

In [ ]:

```
%%perl
use feature qw(say);
$string = "abba" ;
say $string if $string=~ /^a.+a$/;
$string = "aa" ;
say $string if $string=~ /^a.+a$/;
```

# Regular Expressions

- Syntax Continued
  - Grouping `()`
  - Capture References `\1`
  - Pattern Modifiers `i`, `g`
  - Custom Character classes `[]`
  - Alternation `|`

In [ ]:

```
%%perl
use feature qw(say);
$string = "abba" ;
say $string if $string =~ /(\w)(\w)\2\1/
```

In [ ]:

```
%%perl
use feature qw(say);
$string = "ABBA" ;
say $string if $string =~ /(a)(b)\2\1/i
```

In [ ]:

```
%%perl
use feature qw(say);
$string = "10 GB";
say $string if $string =~ /\d+ [GKMT]B/
```

In [ ]:

```
%%perl
use feature qw(say);
$string = "10 PB";
say $string if $string =~ /\d+ [^GKMT]B/
```

# Regular Expressions

- Subsitution
  - Substitution in PERL `s///`
  - Replacing Text
  - Using Capture Buffers (`$1` in replacement)
- What is PCRE, how is this useful in other languages?

In [ ]:

```
%%perl
use feature qw(say);
$string = "Hello Class! Class!";
$string =~ s/Class!/World!/g;
say $string;
```

In [ ]:

```
%%perl
use feature qw(say);
$string = "410-455-1000";
$string =~ s/(\d\d\d)-(\d\d\d-\d\d\d\d)/($1) $2/;
say $string;
```

# Shells

- What is a shell?
- What are some basic unix commands?
  - What is the philosphy behind having so many commands?
- Shell script basics
  - Shebang line
  - Comments
  - Executing Them

In [ ]:

```
%%bash
cat to_sort1.txt
sort to_sort1.txt
```

In [ ]:

```
%%bash
head -n2 to_sort1.txt
tail -n2 to_sort1.txt
```

# Shell Variables

- Declaring Variables (no space between =)
  - Arrays
- Reference Variables
- Interpolation
- Command Line Arguments
- Globbing

In [ ]:

```
%%bash
var=10
str="A String"
echo $var $str
```

In [ ]:

```
%%bash
arr=(10 20 30 40 50)
echo $arr ${arr[0]}
echo ${arr[@]}
```

In [ ]:

```
%%bash
end="the end"
echo "I am interpolating ${end}ing."
```

# Streams

- Redirection
- Pipes
- Here Docs
- Process Substitution `<(command)` Syntax

In [ ]:

```
%%bash
ls -lh *.ipynb > ls.txt
head ls.txt
```

In [ ]:
```
%%bash
ls *.ipynb | sort | head
```

# Control Structures

- If statements
  - `if then else fi`
  - Using `[[ ]]` or `[ ]`
  - Basic tests (`-eq`, `-ne`, `=`, `!=`, etc.)
- Switch Statement
  - `case pattern) ;; esac`

In [ ]:

```
%%bash
a=10
if [[ $a -gt 5 ]]; then
    echo "YAY"
else
    echo "NAY"
fi
```

In [ ]:

```
%%bash
b="Hello"
case $b in
    *a)
        echo "Ends with an A"
        ;;
    He*)
        echo "Contains He"
        ;;
    *)
        echo "Default"
        ;;
esac
```

# Looping

- Count Style For Loops
  - `for ((x = S; x < N; x++)); do ... done`
- For-each style `for f in *.html`
  - Brace Expansion

In [ ]:

```
%%bash
for ((x = 10; x < 20; x++)); do
    echo $x
done
```

In [ ]:

```
%%bash
for f in *2{0..9}.html; do
    wc -l "$f"
done
```

# Looping

- While Loop

```
while CONDITION; do
#CODE_HERE
done
```

# I/O

- User input using `read`
- Reading a file using `read` and `while`
- `echo` and `printf`
- Sourcing a file using `.`

In [ ]:

```
%%bash
while read line; do
    echo $line
done < to_sort1.txt
```

# Math

- `(( ))` Syntax
  - Don't escape variables inside
- `$(( ))` to save results
- Can't do floating point without external program

In [ ]:

```
%%bash
x=0
#((x=0))
((x++))
echo $((x + 2))
```

# Grep/Sed/Awk

- What they are used for
- Basic use of grep
  - Using Perl style Regexes
  - How to count number of matches

In [ ]:

```
%%bash
grep -Pc "\d\d\d-\d\d\d-\d\d\d\d" *.html | grep -P ":[^0]$"
```

# Functions

- Declaring Functions
  - Accessing Function Parameters
- Calling Functions
- Scope
- Returning from functions
  - What does the `return` keyword actually do

In [ ]:

```
%%bash
function mystery(){
    diff $1 $2
    if [[ $? -ne 0 ]]; then
        echo "The files $1 and $2 are different"
    else
        echo "The files $1 and $2 are the same"
    fi
}

mystery "Lecture20.html" "Lecture20.html"
```

# R

- What it is used for
- The assignment operator (<–)
- Data types
  - No Scalars
  - List vs Vector
  - Matrix vs DataFrame

In [ ]:

```
%%script R --no-save -q
a <- 4
print(a)
```

# 1D Structures

- Creating a vector using `c`
- Creating a list using `list`

In [ ]:

```
%%script R --no-save -q
vec <- c(1,2,3,4)
vec2 <- c(1,"2",3,4)
print(vec)
print(vec2)
```

In [ ]:

```
%%script R --no-save -q
l1 <- list(1,2,3,4)
l2 <- list(1,"2",3,4)
print(l1)
print(l2)
```

# Matrices

- Creating using `matrix`
- Creating from vectors

In [ ]:

```
%%script R --no-save -q
m <- matrix(c(1,2,3,4,5,6),nrow=2)
print(m)
```

In [ ]:

```
%%script R --no-save -q
v2m <- c(1,2,3,4,5,6)
print(v2m)
dim(v2m) <- c(2,3)
print(v2m)
```

# DataFrames

- Creating from vectors
- Reading in data
- Common functions

In [ ]:

```
%%script R --no-save -q
df1 <- data.frame(int=c(1,2,3,4),float=c(1.01,2.0,3.0,4.0),str=c("One","Two","Thre
e","Four"))
print(df1)
```

In [ ]:

```
%%script R --no-save -q
df2 <- read.csv("usm.csv")
print(df2)
```

# Math in R

- Can be on a high dimensional object

In [ ]:

```
%%script R --no-save -q
m <- matrix(c(1,2,3,4,5,6),nrow=2)
print(m + 5)
```

In [ ]:

```
%%script R --no-save -q
m <- matrix(c(1,2,3,4,5,6),nrow=2)
print(m * m)
```

# Subsetting

- Indexing starts at 1
- Postive vs Negative Intenger Subsetting
- Boolean Indexing
- `[]` and `[[]]` operations
- `$` operator for Data Frames

In [ ]:

```
%%script R --no-save -q
v2 <- c('a','b','c','d')
print(v2[1])
```

In [ ]:

```
%%script R --no-save -q
v3 <- list('a','b','c','d')
print(v3[1])
print(v3[[1]])
```

In [ ]:

```
%%script R --no-save -q
v4 <- data.frame(letters=c('a','b','c','d'),numbers=c(1,2,3,4),symbols=c
('!','#','$','@'))
print(v4$letters)
```

In [ ]:

```
%%script R --no-save -q
v5 <- c(1,2,3,4,5)
print(v5[-1])
print(v5[v5 %% 2 == 1])
```

# Control Structures

- If, else if
- `for ... in`
  - Rarely Seen in R
- `lapply`

In [ ]:

```
%%script R --no-save -q
a <- 1
if(a > 0)
{
    print("This is Positive")
} else {
    print("This is Negative")
}
```

In [ ]:

```
%%script R --no-save -q
b <- c(1,2,3,4,5)
for(x in b){
    print(x)
}
```

# Functions

- How to declare
- How to return from a function
- How are defaul parameters evaluated?
- Lazy Evaluation
- Infix Functions
- Assignment Functions

In [ ]:

```
%%script R --no-save -q
my_f <- function(a,def=1){
    print(def)
    a * a
}
print(my_f(10))
print(my_f(10,d=10))
```

In [ ]:

```
%%script R --no-save -q
`%+%` <- function(a,b){
    a * b
}
print(10 %+% 10)
```

In [ ]:

```
%%script R --no-save -q
`fir<-`<-function(x,value){
    x[1] <- value
    x
}
y <- c(1,2,3,5)
fir(y)<-10
print(y)
```

# Objects

- There are three systems
- S3 basics
  - Use of `structure` and `class`
  - Constructors
- Methods
  - Generic Functions
  - `UseMethod`
  - Naming Conventions

In [ ]:

```
%%script R --no-save -q
per<-function(name,age){
    structure(list(m_name=name,m_age=age),class="per")
}

form<- function(ob){
    UseMethod("form")
}

form.per <- function(obj){
    paste(obj$m_name,"is",obj$m_age)
}

form.default <- function(obj){
    "Not a Person"
}

p <- per("Bob",20)
print(form(p))
print(form(10))
```

# Regex Functions

- `grep` vs `grepl`
- `perl=TRUE`

In [ ]:

```
%%script R --no-save -q
print(grep("\\w\\w\\w",c("AABDF","AG","EOT"),perl=TRUE))
print(grepl("\\w\\w\\w",c("AABDF","AG","EOT"),perl=TRUE))
```

# Web Basics

- How does a HTTP Request Work?
    - Headers, response
- HTML
    - What is it
    - Not directly tested, in JS/PHP questions
- CSS
    - What is it
    - Not directly tested, in JS/PHP questions

# JavaScript

- What is it?
  - Where does it run?
  - What can't it do?
- Embedding JS in HTML
  - `<script>` tag

```
<script src="location.js"></script>
<script src="https://example.com/remove_script.js"></script>
<script>
    //Embed Code
</script>
```

# JavaScript

- Declaring Variables
  - Scope
- Difference between == and ===

In [ ]:

```
%%html
<p id="results">To Be Replaced</p>
<script>
var abc = 1;
function scope_checker(){
   var abc = 2
}
scope_checker()
document.getElementById("results").innerHTML = abc;
</script>
```

In [ ]:

```
%%html
<p id="results1">To Be Replaced</p>
<script>
var abc = 1;
if(true){
   let abc = 2
}
document.getElementById("results1").innerHTML = abc;
</script>
```

In [ ]:

```
%%html
<p id="results2">To Be Replaced</p>
<script>
var s = "10"
var i = 10
if(s === i){
  document.getElementById("results2").innerHTML = "Triple Equal";
}
else if(s == i){
      document.getElementById("results2").innerHTML = "Double Equal";
}
else{
    document.getElementById("results2").innerHTML = "Not Equal";
}

</script>
```

# JavaScript

- Arrays
  - Are Objects
  - Sorting
- Strings
  - Are objects
  - Basic methods, concatenation

In [ ]:

```
%%html
<p id="results3">To Be Replaced</p>
<script>
var arr = new Array();
arr.push(1);
arr.push(2);
document.getElementById("results3").innerHTML = JSON.stringify(arr);
</script>
```

In [ ]:

```
%%html
<p id="results4">To Be Replaced</p>
<script>
var arr = [3,4,5];
document.getElementById("results4").innerHTML = JSON.stringify(arr);
</script>
```

In [ ]:

```
%%html
<p id="results5">To Be Replaced</p>
<script>
var arr = ["Sand","Dirt","Water"];
function s(a,b){
    a.charAt(a.length-3) - b.charAt(b.length-3);
}
document.getElementById("results5").innerHTML = JSON.stringify(arr.sort(s));
</script>
```

## Control Structures in JS

- `if, else if, else`
- `for, while`
- `for .. in` vs `for .. of`

In [ ]:

```
%%html
<p id="results8">To Be Replaced</p>
<script>
s = " ";
arr = [10,9,8,7,6,5];
for(i in arr){
    s += " " + i;
    i++;
}
document.getElementById("results8").innerHTML = JSON.stringify(s);
</script>
```

In [ ]:

```
%%html
<p id="results9">To Be Replaced</p>
<script>
s = " ";
arr = [10,9,8,7,6,5];
for(i of arr){
    s += " " + i;
    i++;
}
document.getElementById("results9").innerHTML = JSON.stringify(s);
</script>
```

# Functions in JS

- How to declare (the `function` keyword)
  - Where they can be declared
- How to pass functions to other functions
  - Use in sort

In [ ]:

```
%%html
<p id="results10">To be Replaced</p>
<script>
do_something();
function do_something(){
    document.getElementById("results10").innerHTML = "Hoisted";
}
</script>
```

# Objects in JavaScript

- The `this` and `new` keywords
  - How to make a "constructor" function
- Dot syntax
  - Methods are functions stored in keys
- Generic `{ }` syntax
- Prototypes

In [ ]:

```
%%html
<p id="results11">To be Replaced</p>
<script>
function Course(department, number, section){
    this.department = department;
    this.number = number;
    this.section = section;
    this.print = function(){
        return "This is " + this.department + this.number + ",section "  + this.se
ction;
    }
}

today = new Course("CMSC",433,"01");
document.getElementById("results11").innerHTML = today.print();
</script>
```

In [ ]:

```
%%html
<p id="results12">To be Replaced</p>
<script>
Vehicle = {go: function(){
        this.speed = 10;
}}

function Car(color){
    this.color = color;
    this.print = function(){
        return "The " +this.color + ' car is going ' + this.speed + " mph";
    }
}
Car.prototype = Vehicle;
vroom = new Car("red");
vroom.go();
document.getElementById("results12").innerHTML = vroom.print();
</script>
```

# The DOM

- What is the DOM (Document Object Model)
- How to access from JS
- Common properties of DOM objects
- Common methods of DOM objects

In [ ]:

```
%%html
<p class="something"></p>
<div class="something"></div>
<script>
els = document.querySelectorAll(".something");
for(i = 0; i < els.length; i++){
    els[i].innerHTML = "Hello " + i;
}
</script>
```

# Events

- How are events used
- How to bind events to an HTML element
  - What is an event handler?
- Some common events
  - click
  - load
  - blur/focus

In [ ]:

```
%%html
<p id="event">To Respond</p>
<script>
document.getElementById("event").addEventListener('click',
                                function(e){
                                    e.target.innerHTML = "Clicked";
                                });
</script>
```

In [ ]:

```
%%html
<input type="text" id="event2" />
<script>
document.getElementById("event2").addEventListener('focus',
                                                    function(e){
                                                       e.target.value = "In Focus";
                                                    });
document.getElementById("event2").addEventListener('blur',
                                                    function(e){
                                                       e.target.value = "Out of Focu
s";
                                                    });
</script>
```

# AJAX

- What did it originally stand for
- What does it allow
- How to make and handle a request
  - `open`
  - `onreadystatechange`
  - `send`

In [ ]:

```
%%html
<p id="ajaxResult"></p>
<script>
    var xhr = new XMLHttpRequest();
    xhr.open("GET","usm.csv");
    xhr.onreadystatechange = function(){
        if(xhr.readyState == 4){
            if(xhr.status == 200){
                document.getElementById("ajaxResult").innerHTML = xhr.responseText
            }
        }
    };
    xhr.send();
</script>
```

# JSON

- What is it used for
- How to convert from JSON to Javascript object
- How does it relate to AJAX

# PHP

- What is PHP used for?
- The PHP tag (`<?php ?>`)
  - How to use in HTML, ie if statements
- Variables
  - `$`, `$$`, etc.
  - Basic Data Types

In [ ]:

```
%%script php
<?php
$a = "b";
$b = 10;
echo $$a . " " . $b;
```

# PHP

- Control Structures
  - for/ while
  - if, else if, else
  - switch
  - foreach .. as

In [ ]:

```
%%script php
<?php
$a = 10;
if($a > 10){
    ?>
    <strong> 10 is less than 10</strong>
    <?php
} else { ?>
     <strong> 10 is not less than 10</strong>
        <?php }?>
```

In [ ]:

```
%%script php
<?php
$list = [10,20,30,40,50];
foreach($list as $el){
    echo $el * 2;
}
?>
```

# PHP

- Strings
  - Interpolation in Strings
  - Concatenation with `.`
  - `echo`

In [1]:

```
%%script php
<?php
$a = 10;
echo "This is interpolated: $a";
echo 'This isnt $a';
```

```
This is interpolated: 10This isnt $a
```

# Arrays

- Two Types
  - Associative
  - Indexed
- Declaring
- `count` function
- Adding new elements
- Sorting

In [ ]:

```
%%script php
<?php
$indexed = array();
$indexed[] = 1;
$indexed[] = 2;
echo count($indexed);
```

In [ ]:

```
%%script php
<?php
$indexed = [1,2];
print_r($indexed);
```

In [ ]:

```
%%script php
<?php
$assoc = ['lunes'=>'monday','martes' => 'tuesday','miercoles' => 'wednesday'];
$assoc['jueves'] = 'thursday';
asort($assoc);
print_r($assoc);
ksort($assoc);
print_r($assoc);
sort($assoc);
print_r($assoc);
```

# Functions in PHP

- Declaring (`function`) keyword
- Scope
  - `global`
- Returning from Functions
- Pass-by-value is default

In [ ]:

```
%%script php
<?php
$a = 10;
function scope_tester(){
    $a .= "Hello";
}
scope_tester();
echo $a;
```

In [ ]:

```
%%script php
<?php
$a = 10;
function scope_tester(){
    global $a;
    $a .= "Hello";
}
scope_tester();
echo $a;
```

# Files

- Reading and Writing using file handles
- Reading and writing using specialized fucntions
- Opening network locations like files
- Uploading files with `$_FILES`

In [ ]:

```
%%script php
<?php
$fp = fopen("usm.csv",'r');
while($line=fgets($fp)){
    echo $line;
}
```

In [ ]:

```
%%script php
<?php
$stuff = file_get_contents("usm.csv");
echo $stuff;
```

In [ ]:

```
%%script php
<?php
$stuff = file("usm.csv");
print_r($stuff);
```

# Superglobals

- What is a subperglobal
- How to use `$_GET`, `$_POST`, `$_SESSION`
- What type of things are in `$_SERVER`

# PHP Objects

- How to Define
- How to call methods –> syntax

In [ ]:

```
%%script php
<?php
class Money{
    private $dollars;
    private $cents;

    function __construct($d = 0, $c =0){
        $this->dollars = $d;
        $this->cents = $c;
    }

    function __toString()
    {
        return '$' . $this->dollars . "." . $this->cents;
    }

    function getDollars(){
        return $this->dollars;
    }
}

$currency =  new Money(10,10);
echo $currency;
echo "\n";
echo $currency->getDollars();
```

# Regular Expressions

- All start with `preg_`
- How matches are stored
- How to replace
  - What type of data can be used with the replace function

In [ ]:

```
%%script php
<?php
preg_match("/([\w\.]+)@\w+\.\w\w\w/","My email address is bryan.wilkinson@umbc.ed
u",$matches);
print_r($matches);
```

In [ ]:

```
%%script php
<?php
echo preg_replace('/\*\*(.+?)\*\*/','<strong>$1</strong>',"Outside **Make this Str
ong**");
```

## Miscellaneous

- Be able to compare and contrast languages at a high level
  - What language might you choose for which task?
- Julia
  - What it is used for?
  - What is unique about it?
- Ruby
  - What is it used for?
  - What are blocks?