



# CS 1520 / Fall 2005

## Programming Languages for Web Applications

---

06 – Perl: Functions and I/O

Alexandros Labrinidis  
University of Pittsburgh



## How to run Perl programs (revised)

---

- Use text editor (say pico) to write and save program, say `hello.pl`

- **Option 1:**

```
perl -w hello.pl
```

- **Next time:**

```
perl -w hello.pl
```

- **Option 2:**

- Make sure first line of `hello.pl` is the following  
`#!/bin/perl -w`

```
chmod u+x hello.pl
```

```
./hello.pl
```

- **Next time:**

```
./hello.pl
```



## Variables: Scalar/Arrays/Hashes

- **Scalar:**
  - \$a
  - \$asdfldshjfgakhjsdf
  - No type defined
- **Arrays:**
  - @b
  - \$b[1] = second element of array b
  - No size defined
- **Hashes:**
  - %c
  - \$c{"alex"} = element of hash c whose key is "alex"



## Hash Functions

- **keys(%hashname)**
  - return the list of current keys
  - \$fred{"a"} = "b";
  - \$fred{"c"} = "d";
  - \$fred(15) = 143;
  - @list = keys(%fred);     # @list = ("a", "c", 15);
  - \$n = keys(%fred);        # \$n = 3;
  
  - foreach \$k (keys (%fred)) {  
    print "we have \$fred{\$k} at key \$k\n";  
}
- **values(%hashname)**



## Hash functions (cont)

- **each (%hashname)**

- Iterate over all elements of hash hashname

- ```
$lastname{"Alex"} = "Labrinidis";  
$lastname{"Panos"} = "Chrysanthis";  
$lastname{"John"} = "Ramirez";
```

```
while ( ($first, $last) = each (%lastname) ) {  
    print "The last name of $first is $last\n";  
}
```

- **delete**

- ```
delete $lastname{"Panos"};
```



## User-defined functions

- ```
sub myfunction {  
    statement_1;  
    statement_2;  
    ...  
}
```

- ```
sub say_hello {  
    print "Hello world\n";  
}
```

- ```
sub say_what {  
    print "Hello $what\n";    # $what is global var  
}
```



## Invoking user-defined functions

- Very simple:
  - `say_hello();`
- Can also be part of an expression:
  - `$a = $b + say_hello();`
- **Q:** How to return values?
- **A:** using the return command
- Example: `sum.pl`



## How to pass arguments?

- `@_` is a special array that holds all arguments to current function
- Two standard ways to use it:
  - `@_[0]` is the first argument, `@_[1]` is the second argument, ...
  - `($a, $b) = @_;` #assigns first arg to \$a and second arg to \$b
- Examples:
  - `addtwo.pl`



## Lexical Scope

- Global variables
  - Variables outside the function are accessible within the function
  - **Scope:** global
  
- Private variables in functions
  - `my ($sum);`
  - **Scope:** only valid while inside the current statement block (i.e., function)
  
- Semi-private variables in functions
  - `local ($sum)`
  - **Scope:** valid until function terminates (i.e., even if another function was called from within)