# CIS241

## System-Level Programming and Utilities

### bash Functions

**Erik Fredericks, frederer@gvsu.edu**

**Fall 2025**

> **Based on material provided by Erin Carrier, Austin Ferguson, and Katherine Bowers**

# Declaring and calling functions

```
function my_function() {
  commands
}

# To call:
my_function
```

# Function parameters

**Just like normal command line arguments!**

```
function my_function() {
    echo "You passed: $1"
}

# To call:
my_function foo
```

# A TEST

```
function repeat() {
   range=$(seq 0 $2)
   for i in $range; do
      echo "$1"
   done
}

repeat "hello" 5
repeat "!!!" 10
repeat $1 $2
```

**What does this do?**

# Scope

**You have to mark variables as local (and is good practice)**

```
function my_function() {
    local x
    x=5
    echo "Inside function: x = ${x}"
}

x=1
echo "Before function: x = ${x}"
my_function
echo "After function: x = ${x}"
```

# Return values

**Return value in `$?`**

```
function functionName()  {
   # Do stuff
   return 10
}
functionName
echo $?
```

# Return values

```
function functionName()  {
   # Do stuff
   echo "Hi there!"
   return 10
}
val=$( functionName )
echo $val
```

**What happens?**

# Trapping signals

**Remember the whole `kill` command? You can handle it!**

**Common signals:**

**`Ctrl+C` → `SIGINT` (2) → Interrupt process**

**`Ctrl+Z` → `SIGSTP` (18) → Stop/pause process**

**`kill -9` → `SIGKILL (9)` → Cannot be trapped!**

```
# Trap a signal
trap "<command to run>" <signal to trap>
trap "echo THIS IS A TRAP" SIGINT
```

# Why relevant here?

## Make your programs die gracefully!

```
function endScript {
   # write out important data
}
trap endScript EXIT
```