Shell Scripting:

```
#!/bin/sh

## Variables, Strings, Calculation, Command##
var1=123
var2=345
# varstring='456$var'  varstringstring="1 2 3"
# echo $var
# echo $varstring
# echo $varstringstring
# echo $(($var1+$var2))
# ret=$(ls -al)
# echo "The result is"
# echo $ret

# if [ $var2 == 123 ]
# then
#   echo equal
# elif [ $var2 == 345 ]
# then
#   echo "equals to second"
# else
#   echo "not equal"
# fi

# if [ -d "/var/log/wifi.log" ]
# then
#   echo "file exist"
# else
#   echo "not exist"
# fi

### While loop ###
# i=0
# output=""
# while [ $i -lt 3 ]
# do
#   i=$(($i+1))
#   output="$output $i"
# done
# echo $output

### For loop ###
# l=`ls /`
# for i in $l
# do
#   if [ ! -d "/$i" ]
#   then
#     echo "regular file"
#   else
#     echo "dirctory"
#   fi
# done

### Functions ###
# function function_name {
#   a="test"
#   echo $1 $2
#   echo $a
#   b="return"
# }
# function_name arg1_func arg2_func
# echo $b
# echo $# # #of arguments
# echo $*
# echo $$

### Case ###
# varcase=1
# case $varcase in
#   1)
#     echo "Case 1"
#     ;;
#   2)
#     echo "Case 2"
#     ;;
#   *)
#     echo "default"
#     ;;
# esac

### More Arrays ###
# arr=(value1 value2  .... valueN)
# echo ${arr[0]}
# arr[0]=valuenew
# echo ${arr[0]}

### Extras ###
# for i in {1..10}
# do
#   echo $i
# done

# teststr=12345
# teststr2=123456
# echo ${#teststr2}
```

Python Script:

```
# print("new string 1", end="")
# print("new string 2")

# import os
# os.func_name()

a = "this is a string"
# print(type(a))
# print(a[-1])
# print(a[len(a)-1])
# print(len(a))

# print(a[1:-1])

ret = a.find("string")
# print(ret)
# print(a.upper())

# test_list = ["element 1", "element b", 1.0,
5, [1,2]]
# # print(test_list[-1][0])
# print(test_list[1:3])
# print(len(test_list))
# test_list[0] = "element A"
# print(test_list)
# test_list.append("new through append")
# test_list.append("element A")
# test_list.remove('element A')
# print(test_list)
# test_list.pop(0)
# print(test_list)
# test_list.pop(-1)
# print(test_list)

# s = "one,two,three,4"
# print(s.split(','))

# test_tuple = (1,3.0,"test")
# test_tuple[2] = "new value"

# a = (1,2)
# test_dict = {"course name": "35L", "number":
36, "instructor": "prof. eggert", a: "234" }
# print(test_dict[a])
# test_dict['new key'] = "new value'"
# print(test_dict)
# test_dict['new key'] = [test_dict['new
key'], "new value 2"]
# print(test_dict)
# print(test_dict.keys())
# print(test_dict.values())
# print(test_dict.items())
# d = {"1" : "100", "2": "200"}
# if "1" in d:
#   print("1 in d")
# # elif 2 in l:
# #   print("2 in l")
# else:
#   print("not in d")

# a = 0
# while a < 10:
#   print(a)
#   a += 2

l = [1, 2, 3, 4]
d = {"1" : "100", "2": "200"}
# for i in l:
#   print(i + 2)
# for i in d.keys():
#   print(i, d[i])

# a = 3
# b = 10
# for i in range(a, b, 2):
#   if i == 3:
#     continue
#   if i == 9:
#     break
#   print(i)



def func_name(arg1, arg2):
  if arg1 < 0:
    return 0
  tmp = arg1 + arg2
  return tmp

# print(func_name(1,2))

# class className:
#   __var2 = 5
#   def __init__(self, v):
#     self.var1 = v
#   def function_name(self, arg1, arg2):
#     if arg1 < 0:
#       return 0
#     tmp = arg1 + arg2 + self.var1
#     return tmp

# t = className(10)
# ret = t.function_name(1,2)
# print(ret)

# print(t._className__var2)

try:
  func_name(1,2,3)
except TypeError:
  print("error")
```

C Program:

```
#include <stdio.h>
#include <stdlib.h>

// Macro
#define BUFFER_SIZE 1024
#define min(X, Y) ((X) < (Y) ? (X) : (Y))

void f ( int ** a , int ** b);
int f1 (int a);
int f2 (int a);

int main(int argc, char *argv[]) {
  int a = 10;
  int b = 15;

  // Pointer to pointer
  int * a_p = &a;
  int * b_p = &b;

  printf("before f: a_p points to %d\n",
*a_p);
  printf("%p\n", (void *) a_p);

  // Pass pointer to pointer to function
  f(&a_p, &b_p);
  printf("%p\n", (char *) a_p);

  printf("after f: a_p points to %d\n", *a_p);
  // printf ("Number of arguments %d, the
first argument is %s\n", argc, argv[1]);
  int arr1[10];
  // printf("%lu \n", sizeof(arr1));
  // Put int i outside the loop for some older
version of C
  // int i;
  // for ( i =0; i<5; i+=1){printf("%d", i);}

  // Function Pointer
  int (*fn_ptr)(int);
  fn_ptr = f2;
  printf("Return is: %d \n",(*fn_ptr)(1));

  char * p_c;
  printf("pointer: %p\n", p_c);
  p_c = malloc(1);
  printf("pointer: %p\n", p_c);
  *p_c = 'a';
  // Don't try this!! You only allocate 1
bytes but trying to access 2 bytes
  // This might work on your computer, but
actually undefined behavior
  *(p_c+1) = 'b';
  printf("charactor is %c \n", *p_c);
  free(p_c);
  // Don't try this!! You free the memory but
try to access again
  // This might work on your computer, but
actually undefined behavior
  printf("charactor is %c \n", *p_c);
  return 0;
}

void f ( int ** a_ptr , int ** b_ptr) {
  printf("%p\n", *a_ptr);
  // printf("%p\n", a_ptr);
  *a_ptr = *b_ptr;
}

int f1 (int a) {return a+1;}

int f2 (int a) {return a-1;}
```

System Calls:

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>

int main() {
  int fd = open("file.txt", O_RDONLY);
  if (fd < 0) {
    // exit(1);
    perror("open error! ");
  }

  char buf[40];
  int ret = read(fd, buf, 8);
  if (ret < 0) {perror("read error!");}
  // printf("%s\n", buf);
  write(1, buf, 8);
  write(1,"\n",1);
  read(fd, buf, 8);
  write(1, buf, 8);
  write(1, "\n",1);

  lseek(fd, 0, SEEK_SET);
  read(fd, buf, 8);
  write(1, buf, 8);
  write(1, "\n",1);

  lseek(fd, -2, SEEK_CUR);
  read(fd, buf, 8);
  write(1, buf, 8);
  write(1, "\n",1);

  struct stat st;
  fstat(fd, &st);
  printf("The size of this file is %d\n",
st.st_size);

  close(fd);
  return 0;
}
```

```
Makefile:
CC=gcc
CFLAGS=-ldl -Wl,-rpath=.

all: main-load libmylib-d.so
main-load: main-load.c
	$(CC) $^ -o $@ $(CFLAGS)
libmylib-d.so: mylib.h mylib.c
	$(CC) -fPIC -c mylib.c -o mylib.o
	$(CC) -shared mylib.o -o libmylib-d.so
clean:
	rm *.o libmylib-d.so main-load

.PHONY: clean all
```

Dynamic Loading:

```
#include "mylib.h"
#include <dlfcn.h>
#include <stdio.h>

int main() {
  void * handle;
  void (*g)();
  handle = dlopen("./libmylib-d.so",
RTLD_LAZY);
  if (dlerror() != NULL) {
    printf ("error! %s \n", dlerror());
  }
  g = dlsym(handle, "f");
  (*g)();
  dlclose(handle);
  // check with dlerror()
  return 0;
}
```