Python Language Exercises

Draft 23 January 2013

Functions

- Write a function named echo, that receives 1 parameter, and returns that value.

- Write a function called main_function, that receives no parameters and returns no values. This function calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named swap, that receives 2 parameters, and returns the two parameters in reverse order.

- Update the function called main_function, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

Arithmetic Expressions

- Write a function named reverse, that receives 1 numeric parameter, and returns the negative of the numeric value.

- Write a function called main_arithmetic, that receives no parameters and returns no values. This function calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named plus, that receives 2 numeric parameters, and returns the sum of the two parameters.

- Update the function called main_arithmetic, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named diff, that receives 2 numeric parameters, and returns the result of subtracting the second from the first.

- Update the function called main_arithmetic, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named abs_diff, that receives 2 numeric parameters, and returns the numeric difference between the two numbers.

- Update the function called main_arithmetic, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named divide, that receives 2 integer parameters, and returns the result of dividing the first by the second. The result should be integer.

- Update the function called main_arithmetic, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named remainder, that receives 2 integer parameters, and returns the remainder when the first is divided by the second.

- Update the function called main_arithmetic, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named power, that receives 2 parameters, and returns the result of raising the first number to the power of the second.

- Update the function called main_arithmetic, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called calculate, that receives 5 parameters (a-e), and returns a + b divided by d - e, all multiplied by c.

- Update the function called main_arithmetic, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named ratio, that receives 2 float numeric parameters and returns ratio of the biggest to the smallest.

- Update the function called main_arithmetic, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named pythagoras, that receives 2 parameters, and returns the square root of the sum of the squares of the 2 numbers.

- Update the function called main_arithmetic, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

Boolean Expressions (logic)

- Write a function named reverse, that receives 1 boolean parameter, and returns the opposite truth value.

- Write a function called main_boolean, that receives no parameters and returns no values. This function calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named band, that receives 2 boolean parameters, and returns true if both parameters are true. Otherwise, it returns false.

- Update the function called main_boolean, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named bor, that receives 2 boolean parameters, and returns true if at least one parameter is true. Otherwise, it returns false.

- Update the function called main_boolean, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named xsame, that receives 2 boolean parameters, and returns true if both parameters are true or both are false. Otherwise, it returns false.

- Update the function called main_boolean, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named xor, that receives 2 boolean parameters, and returns true if one parameter is true and the other is false. Otherwise, it returns false.

- Update the function called main_boolean, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

Boolean Expressions (numbers)

- Write a function named positive, that receives 1 numeric parameter, and returns true if the number is greater than 0. Otherwise, it returns false.

- Write a function called main_boolean_numbers, that receives no parameters and returns no values. This function calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named bigger, that receives 2 numeric parameters, and returns true if the first parameter is larger. Otherwise, it returns false.

- Update the function called main_boolean_numbers, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named no_diff, that receives 2 numeric parameters, and returns true if the values are the same. Otherwise, it returns false.

- Update the function called main_boolean_numbers, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named not_same, that receives 2 numeric parameters, and returns true if the values are not the same. Otherwise, it returns false.

- Update the function called main_boolean_numbers, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named less_than, that receives 2 numeric parameters, and returns true if the first value is less than the second value. Otherwise, it returns false.

- Update the function called main_boolean_numbers, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named at_least_13, that receives 1 numeric parameter, and returns true if the parameter is 13 or higher. Otherwise, it returns false.

- Update the function called main_boolean_numbers, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named at_most_13, that receives 1 numeric parameter, and returns true if the parameter is 13 or lower. Otherwise, it returns false.

- Update the function called main_boolean_numbers, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

Conditionals

- Write a function named biggest, that receives 2 numeric parameters and returns the value of the largest.

- Write a function called main_conditional, that receives no parameters and returns no values. This function calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named smallest, that receives 2 numeric parameters and returns the value of the smallest.

- Update the function called main_conditional, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named letter_grade, that takes a single numerical parameter that is a percentage of points and returns a string that contains the equivalent letter grade. 60 percent is a D, 70 percent is a C, 80 percent is a B and 90 percent is an A.

- Update the function called main_conditional, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named food_tax, that receives as parameters subtotal (a number), and grocery (a boolean). The function returns a number that represents the tax on subtotal.

  The city has a tax on grocery food at 3 percent. The tax on prepared food (non-grocery) is 7.25 percent.

- Update the function called main_conditional, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named same, that receives 3 parameters and returns true if they are all the same, otherwise it returns false.

- Update the function called main_conditional, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named big3, that receives 3 parameters and returns the largest value.

- Update the function called main_conditional, so that it calls each of the functions listed above with appropriate values, and receives the return

value(s).

- Write a function named small_sum, that receives 3 parameters and returns the sum of the two smallest values.

- Update the function called main_conditional, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named meat_loaf, that receives 3 boolean parameters and returns true if exactly two of the inputs are true.

- Update the function called main_conditional, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named big3reorder, that receives 3 parameters and returns the three values, largest first, and smallest last.

- Update the function called main_conditional, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named positive_multiple, that receives 2 numeric parameters and returns the result of multiplying the two numbers, if both are positive, or both are negative. Otherwise, it returns the negative of the result of multiplying both numbers.

- Update the function called main_conditional, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

Counted Loops

- Write a function named total, that receives 1 numeric parameter, and returns the sum of all numbers from 0 to 1 less than the parameter.

- Write a function called main_counted_loop, that receives no parameters and returns no values. This function calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named total_slice, that receives 2 numeric parameters, and returns the sum of all numbers from the first parameter to 1 less than the second parameter.

- Update the function called main_counted_loop, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named total_slice2, that receives 2 numeric parameters, and returns the sum of all numbers from the smaller parameter to 1 less than the larger parameter.

- Update the function called main_counted_loop, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named total_odds, that receives 2 numeric parameters, and returns the sum of all odd numbers from the first parameter to 1 less than the second parameter.

- Update the function called main_counted_loop, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named total_evens, that receives 2 numeric parameters, and returns the sum of all even numbers from the first parameter to 1 less than the second parameter.

- Update the function called main_counted_loop, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named total_7s, that receives 2 numeric parameters, and returns the sum of all numbers from the first parameter to 1 less than the second parameter, that are multiples of 7.

- Update the function called main_counted_loop, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named total_non7s, that receives 2 numeric parameters, and returns the sum of all numbers from the first parameter to 1 less than the second parameter, that are not multiples of 7.

- Update the function called main_counted_loop, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function named squares, that receives 1 numeric parameter, and returns the sum of the squares of all numbers up to, but not including the parameter.

- Update the function called main_counted_loop, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

Strings

- Write a function called hello, that receives no parameters, and returns a string with the word "hello" in it.

- Write a function called main_string, that receives no parameters and returns no values. This function calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called nothing, that receives no parameters, and returns an empty string.

- Update the function called main_string, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called second_letter, that receives one string parameter. The function returns the character located the second position (index 1) in the string. The string will always have at least 2 characters.

- Update the function called main_string, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called one_letter, that receives two parameters, a string and a number. The function returns the character located at that index in the string. The number will always be in the legal index range.

- Update the function called main_string, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called concatenate, that receives two string parameters, and returns a string that has the first string followed by the second string.

- Update the function called main_string, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called beauty, that receives one string parameter, and returns a string that has the string followed by the word "beauty".

- Update the function called main_string, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called slice_of_life, that receives one string parameter,

and returns a 3 character string starting at the character with index 2. The input string will always be long enough.

- Update the function called main_string, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called slice_of_heaven, that receives 2 parameters, a string and a number. The function returns a 4 character string starting at the character with index specified by the second parameter. The input string will always be long enough.

- Update the function called main_string, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called slice_of_perfection, that receives 3 parameters, a string and two numbers. The function returns a character string starting at the character with index specified by the second parameter, of length specified by the third parameter. The input string will always be long enough.

- Update the function called main_string, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called length, that receives 1 string parameter, and returns the number of characters in the string.

- Update the function called main_string, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

Lists

- Write a function called short_list, that receives no parameters, and returns a list with numbers 1, 2, and 3 in it, in that order.

- Write a function called main_list, that receives no parameters and returns no values. This function calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called hollow, that receives no parameters, and returns an empty list.

- Update the function called main_list, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called third_value, that receives one list parameter. The function returns the item located the third position (index 2) in the list. The list will always have at least 3 items.

- Update the function called main_list, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called one_value, that receives two parameters, a list and a number. The function returns the item located at that index in the list. The number will always be in the legal index range.

- Update the function called main_list, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called add_lists, that receives two list parameters, and returns a list that has all of the elements of the first list followed by all of the elements of the second list.

- Update the function called main_list, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called pie, that receives one list parameter. This function returns the list with one item added onto the end of it. The item should be the number 314.

- Update the function called main_list, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called grow_one, that receives two parameters, one is a list and the other is a single item. This function returns the list with the item added onto the end of it.

- Update the function called main_list, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called sub_list, that receives one list parameter. This function returns a list with 4 items in it that start with the second (index 1) item in the input list. The input list will always be long enough.

- Update the function called main_list, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called sub_list2, that receives one list parameter and one numeric parameter. This function returns a list with 3 items in it that starts item indexed by the numeric parameter. The input list will always be long enough.

- Update the function called main_list, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called sub_list3, that receives one list parameter and two numeric parameters. This function returns a list with the number of items specified by the second numeric parameter, that starts item indexed by the first numeric parameter. The input list will always be long enough.

- Update the function called main_list, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called list_length, that receives one list parameter. This function returns the number of items in the list.

- Update the function called main_list, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

Sequence Loops

- Write a function called list_total, that receives a list of integers as a parameter, and returns the total of all numbers in the list.

- Write a function called main_sequence_loop, that receives no parameters and returns no values. This function calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called list_total2, that receives a list of integers as a parameter, and returns the total of all even numbers in the list.

- Update the function called main_sequence_loop, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called list_total3, that receives a list of integers as a parameter, and returns the total of all numbers in the list indexed by an odd number.

- Update the function called main_sequence_loop, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called string_lower_count, that receives a string as a parameter, and returns the number of lower case letters in the string.

- Update the function called main_sequence_loop, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called string_digit_count, that receives a string as a parameter, and returns the number of digits (0-9) in the string.

- Update the function called main_sequence_loop, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

- Write a function called string_word_count, that receives a string as a parameter, and returns the number of times that a space is followed by a letter (upper or lower case).

- Update the function called main_sequence_loop, so that it calls each of the functions listed above with appropriate values, and receives the return value(s).

```
#
# Functions
#

# Defining a function:
def function_name(parameters):
    function_body
    return values
# function_name is the programmer's choice
# parameters is 0 or more variables, separated by commas
# function_body is 0 or more python statements
# values is 0 or more variables and values to return, separated by commas
# function_body and return are indented to show they are inside of the
#     function definition (usually 4 spaces, but always, all the same)

# Calling a function:
values = function_name(parameters)
# function_name is the name of a defined function
# parameters is 0 or more variables and values, separated by commas,
#     matching the function definition
# values is 0 or more variables to receive returned values, separated by commas

# Example functions
def example1():
    return 3

def example2(x):
    y = x * 3
    return y

def example3(x, y):
    z = x + y
    return z

def example4(x, y):
    z = x + y
    w = x − y
    return z, w

# Example function calls
def main_example():
    a = example1()
    b = example2(a)
    c = example3(7, b)
    d, e = example4(9, 13)
    return

# Example main function call
main_example()
```

```
#
# Arithmetic
#

# Python arithmetic allows the following operators:
# +  : addition of two values
# −  : difference of two values, negation of one value
# *  : product of two values
# /  : division of two values
# %  : remainder of division of two values
# ** : power of two values
# () : grouping of operations to override default precedence (PEMDAS)

# The generic form of binary (two value) operators is this:
# a op b
# where a and b are numeric values, and op is one of the operator symbols above.
# A numeric value may be provided by a literal number, or by a variable that
# currently has a numeric value.


# Example arithmetic
def example1():
    a = 3 + 2
    return a

def example2():
    a = 3 ** 2
    return a

def example3(x):
    y = x % 3
    return y

def example4(x):
    y = −x
    return y

def example5(x, y):
    z = x / y
    return z

def example6(x, y):
    z = 3 * (x − y)/(x + y)
    return z

# Example function calls
def main_example():
    a = example1()
    b = example2()
    c = example3(b)
    d = example4(c)
    e = example5(7, b)
    f = example6(b, 19)
    return

# Example main function call
main_example()
```

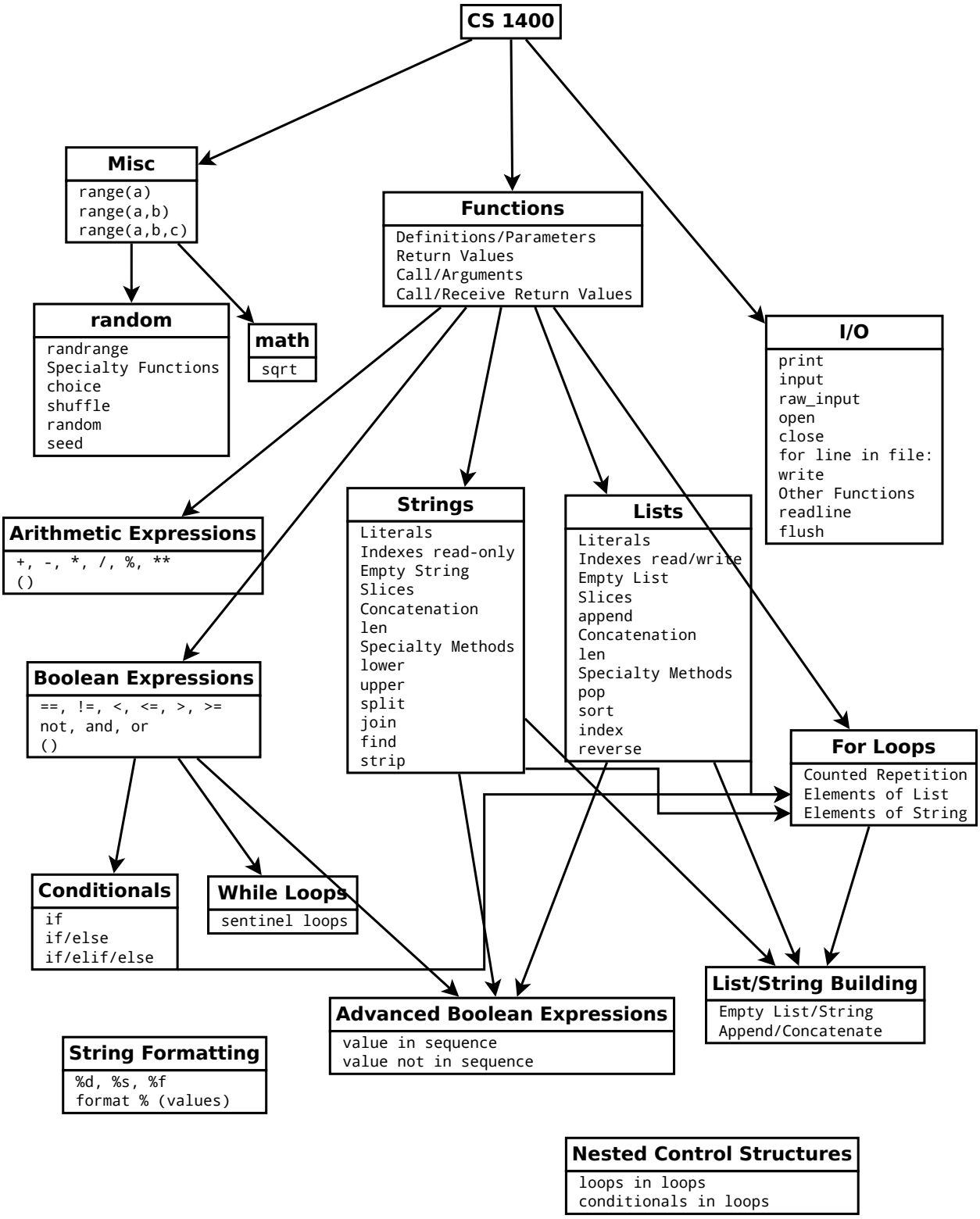
```
#
# Boolean Expressions (Logic)
#

# Python logic allows the following operations:
# True  : always true
# False : never true
# and   : true if two values are both true
# or    : true if at least one of two values is true
# not   : true if one value is false

# ==  : true if two values are the same
# !=  : true if two values are not the same
# >   : true if the first value is larger than the second value
# >=  : true if the first value is larger than or the same as the second value
# <   : true if the first value is smaller than the second value
# <=  : true if the first value is smaller than or the same as the second value

# () : grouping of operations to override default precedence


# Example boolean expressions
def example1():
    return 3 == 3

def example2():
    return 3 != 3

def example3():
    return 4 < 3

def example4(x):
    return 4 > x

def example5(x, y):
    return x == y

# a and b are boolean
def example6(a, b):
    return a and b

def example7(a, b):
    return a or b

def example8(a):
    return not a

# x, y and z are numbers
def example9(x, y, z):
    return (x < y) and (y == z or y > z)


# Example function calls
def main_example():
    a = example1()
    b = example2()
    c = example3()
    d = example4(6)
    e = example5(7, 9)
    f = example6(a, c)
    g = example7(d, c)
    h = example8(g)
    i = example9(5, 6, 7)
```

```
    return

# Example main function call
main_example()
```

```
#
# Conditionals
#

# Using conditionals
if test:
    if_body

if test:
    if_body
else:
    else_body

if test1:
    if_body
elif test2:
    elif_body
else:
    else_body

# test, test1 and test2 are boolean expressions
# if_body, else_body, and elif_body are 1 or more python statements
# The if statement must always come first.
# There can be 0 or more elif statements.
# The else statement must always come last, if it is present.
# if_body, elif_body, and else_body are indented to show they are inside of the
#     structure (usually 4 spaces, but always, all the same)

# Example conditionals
def example1(a, b):
    x = a - b
    if a < b:
        x = b - a
    return x

def example2(a, b):
    if a > b:
        x = a - b
    else:
        x = b - a
    return x

def example3(a):
    if a < 10:
        x = 1
    elif x < 20:
        x = 2
    elif x < 30:
        x = 3
    else:
        x = 4
    return x


# Example function calls
def main_example():
    a = example1(3, 4)
    b = example2(a, 5)
    c = example3(b)
    return

# Example main function call
main_example()
```

```
#
# Counting Loops (for loops)
#

# Using for loops
for variable in sequence:
    for_body

# sequence is the list of values to be used.
# variable is used to store each value, in order.
# for_body is 1 or more python statements.
# for_body is indented to show it is inside of the
#     loop (usually 4 spaces, but always, all the same)

# Example loops
def example1():
    count = 0
    for i in range(10):
        count = count + 1
    return count

def example2(a):
    count = 0
    for i in range(a):
        count = count + 1
    return count

def example3(a):
    total = 0
    for i in range(a):
        total = total + i
    return total

def example4():
    total = 0
    for i in range(3, 7):
        total = total + i
    return total

# Example function calls
def main_example():
    a = example1()
    b = example2(5)
    c = example3(b)
    d = example4()
    return

# Example main function call
main_example()
```

```
#
# Sentinel Loops (while loops)
#

# Using while loops
while test:
    while_body

# test is a boolean expressions
# while_body is 1 or more python statements
# while_body is indented to show it is inside of the
#     loop (usually 4 spaces, but always, all the same)

# Example loops
def example1():
    count = 0
    while count < 10:
        count = count + 1
    return count

def example2(a):
    count = 0
    while count < a:
        count = count + 1
    return count

def example3(a):
    total = 0
    i = 0
    while total < a:
        total = total + i
        i = i + 1
    return i


# Example function calls
def main_example():
    a = example1()
    b = example2(5)
    c = example3(b)
    return

# Example main function call
main_example()
```


CS 1400
Misc
range(a)
range(a,b)
range(a,b,c)
Functions
Definitions/Parameters
Return Values
Call/Arguments
Call/Receive Return Values
I/O
print
input
raw_input
open
close
for line in file:
write
Other Functions
readline
flush
random
randrange
Specialty Functions
choice
shuffle
random
seed
math
sqrt
Arithmetic Expressions
+, -, *, /, %, **
()
Strings
Literals
Indexes read-only
Empty String
Slices
Concatenation
len
Specialty Methods
lower
upper
split
join
find
strip
Lists
Literals
Indexes read/write
Empty List
Slices
append
Concatenation
len
Specialty Methods
pop
sort
index
reverse
Boolean Expressions
==, !=, <, <=, >, >=
not, and, or
()
For Loops
Counted Repetition
Elements of List
Elements of String
Conditionals
if
if/else
if/elif/else
While Loops
sentinel loops
List/String Building
Empty List/String
Append/Concatenate
Advanced Boolean Expressions
value in sequence
value not in sequence
String Formatting
%d, %s, %f
format % (values)
Nested Control Structures
loops in loops
conditionals in loops