**Logo

Description automatically generated**

**San Francisco Bay University**

**Python Programming**

**Homework Assignment #1**

**Due day: 2/11/2024**

**Instruction:**

1. **Push the source code to Github platform.**
2. **Please follow the code style rule like programs on handout.**
3. **Overdue homework submission could not be accepted.**

**4. Takes academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**

1. Build up a function to implement the following operation.

*def if\_function(condition, true\_result, false\_result):*

*"""Return true\_result if condition is a true value, and*

*false\_result otherwise.*

*>>> if\_function(True, 2, 3)*

*2*

*>>> if\_function(False, 2, 3)*

*3*

*>>> if\_function(3==2, 3+2, 3-2)*

*1*

*>>> if\_function(3>2, 3+2, 3-2)*

*5*

*"""*

1. Create a function to add odd numbers less equal than numeric input parameter together as follows

*"""*

*>>> sum\_odd(6) #1+3+5*

*9*

*>>>sum\_odd(7) #1+3+5+7*

*16*

*"""*

1. Define a function for *4* inputs *a, b, c, d*, and return sum of square of two smallest number from *a, b, c* and *d*, such as

*"""*

*>>> foo(1, 2, 3, 4)*

*>>> 5 #*

*>>> foo(-3, 1, 5, 6)*

*>>> 10 #*

*"""*

1. Write a function named *"df"* that takes three integers *x, y,* and *z*. It returns whether subtracting one of these numbers from another gives the third.

*>>> df(5, 3, 2) # 5 - 3 is 2*

*True*

*>>> df(2, 3, 5) # 5 - 3 is 2*

*True*

*>>> df(2, 5, 3) # 5 - 3 is 2*

*True*

*>>> df(-2, 3, 5) # 3 - 5 is -2*

*True*

*>>> df(-5, -3, -2) # -5 - -2 is -3*

*True*

*>>> df(-2, 3, -5) # -2 - 3 is -5*

*True*

*>>> df(2, 3, -5)*

*False*

*>>> df(10, 6, 4)*

*True*

*>>> df(10, 6, 3)*

*False*

1. Create a function that takes an integer *m* greater than *1* and returns the largest integer smaller than *m* that evenly divides *m*.

*def lrgst\_factor(m):*

*"""Return the largest factor of n that is smaller than n.*

*>>> lrgst\_factor (15) # factors are 1, 3, 5*

*5*

*>>> lrgst\_factor (80) # factors are 1, 2, 4, 5, 8, 10, 16, 20, 40*

*40*

*"""*

1. Define a function which takes in a number *n* and determines whether the number is a perfect number. A perfect number is equal to the sum of its factors. For instance, *6* is a perfect number since *6 = 1 + 2 + 3*.

*def pfct\_num(m):*

*"""*

*Returns True or False indicating whether "n" is a perfect*

*number. A number is a perfect number when the sum of all its*

*factors equal the number itself.*

*>>> pfct\_num (6)*

*True*

*>>> pfct\_num (8)*

*False*

*>>> pfct\_num (28)*

*True*

*"""*

1. Implement a function to check if the number of bits from two positive input parameters is the same or not.

*def same\_ord(a, b):*

*"""Return whether positive integers a and b have the same number of digits.*

*>>> same\_ord(50, 70) # 2 bits of a and b*

*True*

*>>> same\_ord(50, 100) # a is 2 bits; b is 3 bits*

*False*

*>>> same\_ord(1000, 100000) # a is 4 bits; b is 6 bits*

*False*

*"""*

1. Write a function that takes in a number and determines if the digits contain two adjacent 5s.

*def double\_5(n):*

*"""Return true if n has two fives in a row.*

*>>> double\_5 (5)*

*False*

*>>> double\_5 (55)*

*True*

*>>> double\_5 (550055)*

*True*

*>>> double\_5 (12345)*

*False*

*>>> double\_5 (50505050)*

*False*

*"""*

1. Design a function that returns the number of unique digits in a positive integer.

*def uniq\_digits(x):*

*"""Return the number of unique digits in positive integer n*

*>>> uniq\_digits (8675309) # All are unique*

*7*

*>>> uniq\_digits (1313131) # 1 and 3*

*2*

*>>> uniq\_digits (13173131) # 1, 3, and 7*

*3*

*>>> uniq\_digits (10000) # 0 and 1*

*2*

*>>> uniq\_digits (101) # 0 and 1*

*2*

*>>> uniq\_digits (10) # 0 and 1*

*2*

*"""*

1. Write a def function "*amc*" with a positive integer "*n*" input parameter. It returns the smallest amicable number greater than "*n*". Two different numbers are both amicable if the sum of the proper divisors of each is equal to the other. Any number that's part of such a pair is an amicable number.

\*Hint: You may want to create a separate function to sum proper divisors.

*def amc(n):*

*"""*

*Return the smallest amicable number greater than positive integer n.*

*Every amicable number x has a buddy y different from x, such that*

*the sum of the proper divisors of x equals y, and*

*the sum of the proper divisors of y equals x.*

*For example, 220 and 284 are both amicable because*

*1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 is 284, and*

*1 + 2 + 4 + 71 + 142 is 220*

*>>> amc(5)*

*220*

*>>> amc(220)*

*284*

*>>> amc(284)*

*1184*

*>>> r = amc(5000)*

*>>> r*

*5020*

*"""*