

# 

[Conditionals and Loops](#_4vieutzdogva) **1**

[Logic and Functions](#_idmqy9aew36e) **5**

[Strings](#_glzuanwi0gp9) **10**

# 

# Conditionals and Loops

1. Write a program using a loop, to get the following output. The output should be formatted the same way as below.

\* \* \* \* \* \*   
 \* \* \* \* \*   
 \* \* \* \*   
 \* \* \*   
 \* \*   
 \*

1. Write a Python program to construct the following pattern, using a nested for loop.

\*  
 \* \*  
 \* \* \*  
 \* \* \* \*  
 \* \* \* \* \*  
 \* \* \* \*  
 \* \* \*  
 \* \*  
 \*

1. Write a Python program that accepts a word from the user and reverse it.
2. Write a Python program which takes two digits m (row) and n (column) as input and generates a two-dimensional array. The element value in the i-th row and j-th column of the array should be i\*j.
3. Write a Python program that accepts a string and calculate the number of digits and letters.
4. Write a Python program to check the validity of password input by users.

Validation :

* At least 1 letter between [a-z] and 1 letter between [A-Z].
* At least 1 number between [0-9].
* At least 1 character from [$#@].
* Minimum length 6 characters.
* Maximum length 16 characters.

1. Write a Python program to find numbers between 100 and 400 (both included) where each digit of a number is an even number. The numbers obtained should be printed in a comma-separated sequence.
2. Write a Python program to display the sign of the Chinese Zodiac for a given year in which you were born.
3. Write a Python program to check if a triangle is equilateral, isosceles or scalene.
   1. An equilateral triangle is a triangle in which all three sides are equal.
   2. A scalene triangle is a triangle that has three unequal sides.
   3. An isosceles triangle is a triangle with (at least) two equal sides.
4. What are m and n after the following code is executed?

|  |
| --- |
| n = 123456789  m = 0  while n != 0:  m = (10 \* m) + (n % 10)  n //= 10 |

1. a, b, c = 0, 0, 0 . Write a python program to print all permutations using those three variables

Output : 000 , 001 ,002, 003, 004, 005 ,006, 007, 008, 009, 010, 011 …… 999

1. Write a program that asks the user to enter 2 numbers: first number is integer and second number is number of numbers that will be included in the multiplication table. Program should output a multiplication table.

Eg:

X = 15

no\_numbers=10

Output:

15 X 1 = 15

15 X 2 = 30

15 X 3 = 45

15 X 4 = 60

15 X 5 = 75

15 X 6 = 90

15 X 7 = 105

15 X 8 = 120

15 X 9 = 135

15 X 10 = 150

1. Write a program that will continuously ask users to enter numbers. Once user types “done”, the program should stop asking for more numbers and display the sum of all the numbers that user entered.
2. Write a program that asks the user to enter one number and then display the factorial of that number.
3. Write a program that asks the user to enter one number and then display the fibonacci sequence for that number.
4. Write a program that receives 10 integers and then display their sum and average value on the screen.
5. Write a program that asks the user to enter one number and then display the sum of all even/odd numbers from 0 to the number that user entered.
6. Write a program that asks the user to enter a number of rows and then the program should output Floyd’s triangle.

Eg.

Number of rows = 5

Output:

1

01

101

0101

10101

1. Write a program that asks the user to enter a number and then print all perfect numbers from 0 all the way to the entered number (Perfect number is a positive number whose sum of all positive divisors excluding that number is equal to that number. For example 6 is the perfect number since divisors of 6 are 1, 2 and 3. Sum of its divisor is 1 + 2+ 3 = 6)
2. Write a program that finds the number and sum of all numbers between x and y which are divisible by z.
3. Write a program to Check Whether a Number can be Express as Sum of Two Prime Numbers.
4. Write a program that checks whether a triangle is right-angled triangle ( Triangle is right-angled if we can apply Pythagorean theorem)
5. Lilly is N years old. For each birthday she receives a present. For each odd birthday (1, 3, 5, …, n) she receives toys, and for each even birthday (2, 4, 6, …, n) she receives money. For her second birthday she received 10.00 USD, and the amount is increased by 10.00 USD for each following birthday (2 -> 10, 4 -> 20, 6 -> 30 etc.). Over the years Lilly has secretly saved her money. Lilly's brother, in the years when she received money, took 1.00 USD from each of the amounts. Lilly has sold the toys, received over the years, each one for P USD and added the sum to the amount of saved money. With the money she wanted to buy a washing machine for X USD. Write a program that calculates how much money she has saved and if it is enough to buy a washing machine. ( resource: <https://csharp-book.softuni.org/Content/Chapter-5-2-loops-exam-problems/exam-problems/smart-lily/smart-lily.html> )
6. What is the output of this program?

For x in range(4):

For y in range (5):

print(“Hello World”)

print(“Hello student”)

print(“Hello everyone”)

1. Write a program that will ask users to enter numbers repeatedly. Output should show the product of all even numbers that the user enters and sum of all odd numbers that user enters. Once the user enters 0, the program should stop asking the user for more inputs.
2. Write a program that will simulate “Hot and Cold” games. Target number is generated randomly when program and program should repeatedly ask the user to enter his guess number. If his number is larger or greater than that number + 3, the program should output “Warm” in all other cases it should output “Cold”. Program should display “Congratulations, your guess was right” if the user enters the correct number.

# Logic and Functions

1. Write a program with a loop that repeatedly asks the user to enter a word. The user should enter nothing (press Enter without typing anything) to signal the end of the loop. Once the loop ends, the program should display the average length of the words entered, rounded to the nearest whole number.
2. How many times is the letter o printed by the following statements?
3. s = "long long story"

for i in range(len(s)):

if i % 3 == 0:

print(s[i])

1. t = " bonobo"

s = t[:-3] + t[1:3]+t[:5]

for i in range(len(s)):

if i % 2 == 0:

print(s[i])

1. How many times is the letter o printed by the following statements?

t = "landinafrica"

s = t[:-2] + t[1:]

for i in range(len(s)):

if i % 3 == 0:

print(s[i])

1. When squirrels get together for a party, they like to have cigars. A squirrel party is successful when the number of cigars is between 40 and 60, inclusive. Unless it is the weekend, in which case there is no upper bound on the number of cigars. Write a function to return True if the party with the given values is successful, or False otherwise.

cigar\_party(30, False) → False

cigar\_party(50, False) → True

cigar\_party(70, True) → True

1. The squirrels in Palo Alto spend most of the day playing. In particular, they play if the temperature is between 60 and 90 (inclusive). Unless it is summer, then the upper limit is 100 instead of 90. Given an int temperature and a boolean is\_summer, write a function to return True if the squirrels play and False otherwise.  
   squirrel\_play(70, False) → True  
   squirrel\_play(95, False) → False  
   squirrel\_play(95, True) → True
2. You are driving a little too fast, and a police officer stops you. Write code to compute the result, encoded as an int value: 0=no ticket, 1=small ticket, 2=big ticket. If the speed is 60 or less, the result is 0. If speed is between 61 and 80 inclusive, the result is 1. If speed is 81 or more, the result is 2. Unless it is your birthday -- on that day, your speed can be 5 higher in all cases. Write a function.   
   caught\_speeding(60, False) → 0  
   caught\_speeding(65, False) → 1  
   caught\_speeding(65, True) → 0
3. Given 2 ints, a and b, return their sum. However, sums in the range 10..19 inclusive, are forbidden, so in that case just return 20. Write a function.   
   sorta\_sum(3, 4) → 7  
   sorta\_sum(9, 4) → 20  
   sorta\_sum(10, 11) → 21
4. Given a day of the week encoded as 0=Sun, 1=Mon, 2=Tue, ...6=Sat, and a boolean indicating if we are on vacation, return a string of the form "7:00" indicating when the alarm clock should ring. Weekdays, the alarm should be "7:00" and on the weekend it should be "10:00". Unless we are on vacation -- then on weekdays it should be "10:00" and weekends it should be "off". Write a function.  
   alarm\_clock(1, False) → '7:00’  
   alarm\_clock(5, False) → '7:00'  
   alarm\_clock(0, False) → '10:00'
5. The number 6 is a truly great number. Given two int values, a and b, write a function to return True if either one is 6. Or if their sum or difference is 6. Note: the function abs(num) computes the absolute value of a number.  
   love6(6, 4) → True  
   love6(4, 5) → False  
   love6(1, 5) → True
6. Given a number n, return True if n is in the range 1..10, inclusive. Unless outside\_mode is True, in which case return True if the number is less or equal to 1, or greater or equal to 10. Write a function.  
   in1to10(5, False) → True  
   in1to10(11, False) → False  
   in1to10(11, True) → True
7. Given a non-negative number "num", return True if num is within 2 of a multiple of 10. Note: (a % b) is the remainder of dividing a by b, so (7 % 5) is 2. Write a function  
   near\_ten(12) → True  
   near\_ten(17) → False  
   near\_ten(19) → True
8. We want to make a row of bricks that is an inch long. We have a number of small bricks (1 inch each) and big bricks (5 inches each). Write a function to return True if it is possible to make the goal by choosing from the given bricks.  
   make\_bricks(3, 1, 8) → True  
   make\_bricks(3, 1, 9) → False  
   make\_bricks(3, 2, 10) → True
9. Given 3 int values, a b c, write a function to return their sum. However, if one of the values is 13 then it does not count towards the sum and values to its right do not count. So for example, if b is 13, then both b and c do not count.  
   lucky\_sum(1, 2, 3) → 6  
   lucky\_sum(1, 2, 13) → 3  
   lucky\_sum(1, 13, 3) → 1
10. Write a function that for given three integers, a b c, return true if it is possible to add two of them to get the third.  
    twoAsOne(1, 2, 3) → True  
    twoAsOne(3, 1, 2) → True  
    twoAsOne(3, 2, 2) → False
11. For this problem, we'll round an int value up to the next multiple of 10 if its rightmost digit is 5 or more, so 15 rounds up to 20. Alternately, round down to the previous multiple of 10 if its rightmost digit is less than 5, so 12 rounds down to 10. Write a function that for given 3 ints, a b c, returns the sum of their rounded values.

round\_sum(16, 17, 18) → 60

round\_sum(12, 13, 14) → 30

round\_sum(6, 4, 4) → 10

1. Write a function that will return the number of even digits smaller than 7 in an integer.
2. Write a function blckjack that for 2 int values greater than 0, returns whichever value is nearest to 21 without going over. Return 0 if they both go over.

blackjack(19, 21) → 21

blackjack(21, 19) → 21

blackjack(19, 22) → 19

1. Write a function that for given 3 int values, a b c, returns their sum. However, if one of the values is the same as another of the values, it does not count towards the sum.

loneSum(1, 2, 3) → 6

loneSum(3, 2, 3) → 2

loneSum(3, 3, 3) → 0

1. Write a function that for given three integers, a b c, return true if it is possible to add two of them to get the third.

twoAsOne(1, 2, 3) → True

twoAsOne(3, 1, 2) → True

twoAsOne(3, 2, 2) → False

1. Write a function that takes a year as a parameter and returns True if the year is a leap year, False otherwise. 3 criteria must be taken into account to identify leap years:

* the year is evenly divisible by 4;
* if the year can be evenly divided by 100, it is not a leap year, unless
* the year is also evenly divisible by 400. Then it is a leap year.

1. We want to make a package of goal kilos of chocolate. We have small bars (1 kilo each) and big bars (5 kilos each). Return the number of small bars to use, assuming we always use big bars before small bars. Write a function to return -1 if it can't be done.  
   make\_chocolate(4, 1, 9) → 4  
   make\_chocolate(4, 1, 10) → -1  
   make\_chocolate(4, 1, 7) → 2
2. Write a function which will find all such numbers which are divisible by 7 but are not a multiple of 5, between numbers a and b (both included).
3. Write a function to compute the sum of first n given prime numbers.
4. A happy number is defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers, while those that do not end in 1 are unhappy numbers. Write a function to check whether a number is "happy" or not.  
   is\_happy(7) -> True

is\_happy(932) -> True

is\_happy(6) -> False

1. Write a function to count the number of prime numbers less than a given non-negative number.  
   number\_of\_primes(10) -> 4

number\_of\_primes(100) -> 25

1. Write a function to check whether a given integer is a palindrome or not. Note: An integer is a palindrome when it reads the same backward as forward. Negative numbers are not palindromic.  
   is\_palindrome(100) -> False

is\_palindrome(252) -> True

is\_palindrome(-838) -> False

1. Write a function to find the largest and smallest digit of a given number.  
   largest\_smallest\_digit(9387422) -> Largest: 9, Smallest: 2
2. Write a function to find out if the given number is abundant. Note: In number theory, an abundant number or excessive number is a number for which the sum of its proper divisors is greater than the number itself. The integer 12 is the first abundant number. Its proper divisors are 1, 2, 3, 4 and 6 for a total of 16.  
   is\_abundant(12) -> True

is\_abundant(13) -> False

1. Write a function to sum all amicable numbers from 1 to specified numbers. Note: Amicable numbers are two different numbers so related that the sum of the proper divisors of each is equal to the other number. (A proper divisor of a number is a positive factor of that number other than the number itself. For example, the proper divisors of 6 are 1, 2, and 3.)

amicable\_numbers\_sum(9999) -> 31626

amicable\_numbers\_sum(999) -> 504

amicable\_numbers\_sum(99) -> 0

1. Write a function to find the next smallest palindrome of a specified number.  
   next\_smallest\_palindrome(99) -> 101

next\_smallest\_palindrome(1221) -> 1331

1. Write a function to check whether a given number is a Disarium number. A Disarium number is a number defined by the following process: Sum of its digits powered with their respective position is equal to the original number. For example 175 is a Disarium number: as 11+72+53 = 135
2. Write a function to check if a given number is a Harshad number or not. Return True if the number is Harshad otherwise False. In recreational mathematics, a Harshad number in a given number base, is an integer that is divisible by the sum of its digits when written in that base.

Example: The number 18 is a Harshad number in base 10, because the sum of the digits 1 and 8 is 9 (1 + 8 = 9), and 18 is divisible by 9.

The number 19 is not a harshad number in base 10, because the sum of the digits 1 and 9 is 10 (1 + 9 = 10), and 19 is not divisible by 10.  
is\_harsad(666) -> True

is\_harsad(11) -> False

# Strings

1. Write a function that for two given strings prints all letters from the first string that appear at least twice in the second string.
2. Write a program that accepts a sentence as input and converts each word to “Pig Latin.” In one version, to convert a word to Pig Latin, you remove the first letter and place that letter at the end of the word. Then, you append the string “ay” to the word. Here is an example:

English: I SLEPT MOST OF THE NIGHT

Pig Latin: IAY LEPTSAY OSTMAY FOAY HETAY IGHTNAY

1. Write a function that for a given string counts the number of words ending in 'y' or 'z' -- so the 'y' in "heavy" and the 'z' in "fez" count, but not the 'y' in "yellow" (not case sensitive). We'll say that a y or z is at the end of a word if there is not an alphabetic letter immediately following it.

countYZ("gray day thiz Monday") → 4

countYZ("gray day this Tuesday") → 3

countYZ("day fyyyz") → 2

1. We'll say that a "triple" in a string is a char appearing three times in a row. Write a function that returns the number of triples in the given string. The triples may overlap.

countTriple("abcXXXabc") → 1

countTriple("xxxabyyyycd") → 3

countTriple("a") → 0

1. What is printed by the following statements?

animal = "umzwilili"

print(animal[3:7])

print(animal[1:3]+animal[2:])

print(animal[-3:-1]\*2)

print(animal[4:99]+animal[1:-2])

print(’a’ in animal)

1. Write a function called removedup that takes a string and creates a new string by only adding those characters that are not already present. In other words, there will never be a duplicate letter added to the new string.
2. Write a function that removes all occurrences of a given letter from a string. If a given letter does not appear in the string, then remove the first letter of the string.
3. What is printed by the following statements?

animal = "alpaca"

print(animal[2:4])

print(animal[:4]+animal[2:])

print(animal[3:-1]\*3)

print(animal[7:99]+animal[1:2])

print(’t’ in animal)

1. What is printed by the following statements?

aflist = ["ba", ["ab", "bc", "ad"],"bd"]

print(aflist[-2]+ ["ad", "ad"])

print(aflist[1:] \* 2)

print([aflist[-1]+ ["af", "ag"]] \* 2)

1. Write a function that removes the last occurrence of a given letter from a given string. If a given letter does not appear in the string, then remove the last and the first letter of the string.
2. Given two strings, a and b, write a function to return the result of putting them together in the order abba, e.g. "Hi" and "Bye" returns "HiByeByeHi".  
   make\_abba('Hi', 'Bye') → 'HiByeByeHi'  
   make\_abba('Yo', 'Alice') → 'YoAliceAliceYo'  
   make\_abba('What', 'Up') → 'WhatUpUpWhat'
3. Given an "out" string length 4, such as "<<>>", and a word, write a function to return a new string where the word is in the middle of the out string, e.g. "<<word>>".

make\_out\_word('<<>>', 'Yay') → '<<Yay>>'

make\_out\_word('<<>>', 'WooHoo') → '<<WooHoo>>'

make\_out\_word('[[]]', 'word') → '[[word]]'

1. Given a string, write a function to return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.

extra\_end('Hello') → 'lololo'

extra\_end('ab') → 'ababab'

extra\_end('Hi') → 'HiHiHi'

1. Given a string, write a function to return the string made of its first two chars, so the String "Hello" yields "He". If the string is shorter than length 2, return whatever there is, so "X" yields "X", and the empty string "" yields the empty string "".

first\_two('Hello') → 'He'

first\_two('abcdefg') → 'ab'

first\_two('ab') → 'ab'

1. Given a string of even length, write a function to return the first half. So the string "WooHoo" yields "Woo".

first\_half('WooHoo') → 'Woo'

first\_half('HelloThere') → 'Hello'

first\_half('abcdef') → 'abc'

1. Given a string, write a function to return a version without the first and last char, so "Hello" yields "ell". The string length will be at least 2.

without\_end('Hello') → 'ell'

without\_end('java') → 'av'

without\_end('coding') → 'odin'

1. Given 2 strings, a and b, write a function to return a string of the form short+long+short, with the shorter string on the outside and the longer string on the inside. The strings will not be the same length, but they may be empty (length 0).

combo\_string('Hello', 'hi') → 'hiHellohi'

combo\_string('hi', 'Hello') → 'hiHellohi'

combo\_string('aaa', 'b') → 'baaab'

1. Given 2 strings, write a function to return their concatenation, except omit the first char of each. The strings will be at least length 1.

non\_start('Hello', 'There') → 'ellohere'

non\_start('java', 'code') → 'avaode'

non\_start('shotl', 'java') → 'hotlava'

1. Given a string, write a function to return a "rotated left 2" version where the first 2 chars are moved to the end. The string length will be at least 2.

left2('Hello') → 'lloHe'

left2('java') → 'vaja'

left2('Hi') → 'Hi'

1. Given a string, write a function to return a string where for every char in the original, there are two chars.

double\_char('The') → 'TThhee'

double\_char('AAbb') → 'AAAAbbbb'

double\_char('Hi-There') → 'HHii--TThheerree'

1. Write a function to return the number of times that the string "hi" appears anywhere in the given string.  
   count\_hi('abc hi ho') → 1  
   count\_hi('ABChi hi') → 2  
   count\_hi('hihi') → 2
2. Write a function to return True if the string "cat" and "dog" appear the same number of times in the given string.  
   cat\_dog('catdog') → True

cat\_dog('catcat') → False

cat\_dog('1cat1cadodog') → True

1. Write a function to return the number of times that the string "code" appears anywhere in the given string, except we'll accept any letter for the 'd', so "cope" and "cooe" count.  
   count\_code('aaacodebbb') → 1

count\_code('codexxcode') → 2

count\_code('cozexxcope') → 2

1. Given two strings, write a function to return True if either of the strings appears at the very end of the other string, ignoring upper/lower case differences (in other words, the computation should not be "case sensitive"). Note: s.lower() returns the lowercase version of a string.

end\_other('Hiabc', 'abc') → True

end\_other('AbC', 'HiaBc') → True

end\_other('abc', 'abXabc') → True

1. Write a function to return True if the given string contains an appearance of "xyz" where the xyz is not directly preceded by a period (.). So "xxyz" counts but "x.xyz" does not.

xyz\_there('abcxyz') → True

xyz\_there('abc.xyz') → False

xyz\_there('xyz.abc') → True

1. Write a function to get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself.

Sample String : 'restart'

Expected Result : 'resta$t'

1. Write a function to get a single string from two given strings, separated by a space and swap the first two characters of each string.

Sample String : 'abc', 'xyz'

Expected Result : 'xyc abz'

1. Write a function to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing' then add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged.

Sample String : 'abc'

Expected Result : 'abcing'

Sample String : 'string'

Expected Result : 'stringly'

1. Write a function to find the first appearance of the substring 'not' and 'poor' from a given string, if 'not' follows the 'poor', replace the whole 'not'...'poor' substring with 'good'. Return the resulting string.

Sample String : 'The lyrics is not that poor!'

'The lyrics is poor!'

Expected Result : 'The lyrics is good!'

'The lyrics is poor!'

1. Write a function to find the first non-repeating character in a given string.
2. Write a function to find the first repeated character in a given string
3. Write a function to remove spaces from a given string.
4. Write a function to find maximum length of consecutive 0’s in   
   given binary string.  
   max\_consecutive\_0('111000010000110) -> 4  
   max\_consecutive\_0('111000111') -> 3
5. Write a function to replace each character of a word of length five and more with hash character (#).  
   replace\_words('Count the lowercase letters in the said list of words') -> ##### the ######### ####### in the said list of ######
6. Write a function to compute the sum of digits of a given string.

digit\_sum(‘123’) -> 6

1. Write a function to lowercase first n characters in a string.  
   lower\_n\_letters('W3RESOURCE.COM', 4) -> w3reSOURCE.COM