# PYTHON PROGRAMMING

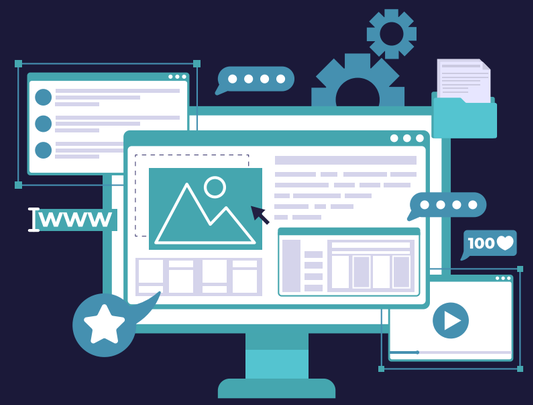
PICT IEEE STUDENT BRANCH

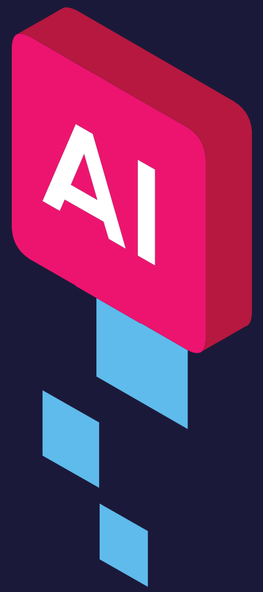
# Contents

# Python - History and Uses

Developed by Guido Van Rossum in 1991
Extensively used in:

- Backend Web Development
- Automation
- Machine learning
- Data Science and Data Analysis



# Data types and casting in Python

| | |
|---|---|
| **Text Type:** | **str** |
| **Numeric Types:** | **int, float, complex** |
| **Sequence Types:** | **list, tuple, range** |
| **Mapping Type:** | **dict** |
| **Set Types:** | **set, frozenset** |
| **Boolean Type:** | **bool** |
| **Binary Types:** | **bytes, bytearray, memoryview** |
| **None Type:** | **NoneType** |

# Problem 1

Help Jahan complete the following tasks on the given data.

Task 1: Convert all the int and float datatype to string and add them in such a way that it depicts today's date.

Task 2: Convert all the string data having decimals to int and add them.

Task 3: Convert all the string data without decimals to float and add them.

Print all the three summations with their data type.

"3.8", 6.202, "4", 1.0, "5", "7.2", 2, "9.087", "1"

# Problem 2

Accept a positive integer as input and print True if it is a perfect square and False otherwise. For example, if the input is 25 , then you must print True . If the input is 15 , then you must print False .

# Strings and string formatting in Python

# Problem 3

Accept a word as input. Assume that all characters in the word are in lower case. Print True if the first and last letter of the word are the same, and False otherwise.

Note -- Use string slicing to solve this problem

# Problem 4

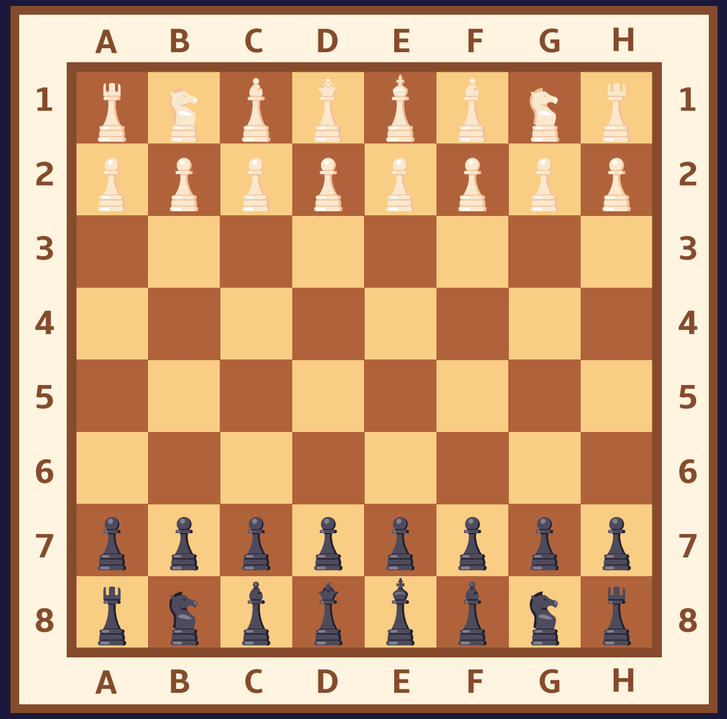
Write a python code to reverse a String without using a for loop or a while loop.

# Python Conditional Statements

# Problem 5

A three digit number is called a sandwich number if the difference between its first and last digit is equal to its middle digit. Accept a three digit number as input and print sandwich if the number is a sandwich number. Print plain if the number is not a sandwich number. For example, 415 and 112 are sandwich numbers.

# Problem 6




Accept two positions as input: start and end .
Determine if a bishop at start can move to end.

# Loop Control in Python

# Problem 7

Write a program using for and while loop to print the sum of all odd numbers from 1 to 100.

# Problem 8

Write a code to accept the name of a person as input and print the initials as output. Assume that the name will be of this form: <first name> <last name> . Also assume that the first name and last name will be a single word, and there will be exactly one space between the two names. For example, if the input is Hardik Pandya, the output should be HP.

# Problem 9

Write a program to find the highest common factor (HCF) of two numbers.

# Problem 10

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 2 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| 3 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 | 30 |
| 4 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 |
| 5 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| 6 | 6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 | 60 |
| 7 | 7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 | 70 |
| 8 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 |
| 9 | 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 | 90 |
| 10 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |

Write a program to print the following table.
The numbers are separated by the tab character.

# Iterable Datatypes

1) Lists
2) Tuples
3) Sets
4) Dictionaries

1) Lists:
List items are ordered, changeable, and allow duplicate values.
E.g. mylist = ["apple", "banana", "cherry"]

2) Tuples
Tuple items are ordered, unchangeable, and allow duplicate values.
E.g. mytuple = ("apple", "banana", "cherry")

3) Sets

Set items are unordered, unchangeable and duplicate entries are not allowed.

E.g. myset = {"apple", "banana", "cherry"}

4) Dictionaries

Dictionary items are ordered, changeable, and does not allow duplicates.

Dictionary items are presented in key:value pairs

E.g. mydict = { "brand": "Ford", "model": "Mustang", "year": 1964}

# Basic operations and built-in methods of different iterable data types.

# 1. List

| | |
|---|---|
| **append()** | **Adds an element at the end of the list** |
| **clear()** | **Removes all the elements from the list** |
| **copy()** | **Returns a copy of the list** |
| **count()** | **Returns the number of elements with the specified value** |
| **extend()** | **Add the elements of a list (or any iterable), to the end of the current list** |
| **index()** | **Returns the index of the first element with the specified value** |
| **insert()** | **Adds an element at the specified position** |
| **pop()** | **Removes the element at the specified position** |
| **remove()** | **Removes the item with the specified value** |
| **reverse()** | **Reverses the order of the list** |
| **sort()** | **Sorts the list** |

# 2. Tuple

| | |
|---|---|
| **count()** | **Returns the number of times a specified value occurs in a tuple** |
| **index()** | **Searches the tuple for a specified value and returns the position of where it was found** |

# 3. Set

| | |
|---|---|
| **add()** | **Adds an element to the set** |
| **clear()** | **Removes all the elements from the set** |
| **copy()** | **Returns a copy of the set** |
| **difference()** | **Returns a set containing the difference between two or more sets** |
| **discard()** | **Remove the specified item** |
| **intersection()** | **Returns a set, that is the intersection of two other sets** |
| **pop()** | **Removes an element from the set** |
| **remove()** | **Removes the specified element** |
| **union()** | **Return a set containing the union of sets** |
| **update()** | **Update the set with the union of this set and others** |

# 4. Dictionary

| | |
|---|---|
| **clear()** | **Removes all the elements from the dictionary** |
| **copy()** | **Returns a copy of the dictionary** |
| **get()** | **Returns the value of the specified key** |
| **items()** | **Returns a list containing a tuple for each key value pair** |
| **keys()** | **Returns a list containing the dictionary's keys** |
| **pop()** | **Removes the element with the specified key** |
| **popitem()** | **Removes the last inserted key-value pair** |
| **update()** | **Updates the dictionary with the specified key-value pairs** |
| **values()** | **Returns a list of all the values in the dictionary** |

# Problem 11

A bot starts from the origin. It is allowed to make the following moves:
UP, DOWN, RIGHT, LEFT.
Each move has a magnitude of 1 unit. Given the moves made by the bot calculate the total distance travelled by the bot. The distance mentioned here is Euclidean distance.

UP
DOWN
LEFT
LEFT
RIGHT
UP

# Problem 12

Perfom various operations on the given list to convert it to the expected list given.

Given list: [3, 9, 6, 12]

Expected lists:

[12, 10, 9, 6]

[3, 6, 9, 12]

[4, 3, 2, 1]

# Problem 13

Perfom various operations on the given set to convert it to the expected set given.

Given set: {3, 9, 6, 12}

Expected set:
{3, 4, 5, 6}

# Problem 14

Simulating Nim

Simulate the following game. There are marbles in a pile to begin with. Two players take turns to play the game. Each player can remove at most four marbles from the pile, but has to remove at least one marble. The player who removes the last marble is the loser. One of the players will be the computer. The other player will be a human. For every turn, you must accept an input from the user which will be the number of marbles that the user has to remove from the pile. If the user makes a wrong move — more than four marbles or less than one marble — flash a warning message and keep accepting input until a valid input is received. For simulating the computer's moves, use the random library. This game has a winning strategy for the person who makes the first move. Can you figure it out?

# Problem 15

Two vectors are parallel to each other if one can be expressed as a scalar multiple of the other.
For example [1, 2, 3] and [4, 8, 12] are parallel because
[4, 8, 12] is 4. [1, 2, 3]. The expression
4 . [1, 2, 3] stands for the vector obtained by multiplying each element of [1, 2, 3] by 4 . In general,
two vectors u and v are parallel, if we there is a non-zero scalar c such that:
v = c . u
Accept two vectors as input from the user and determine if they are parallel to each other