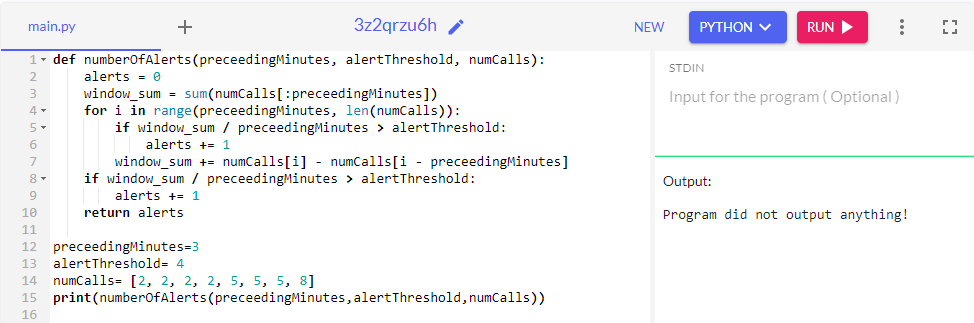
**Coding Round Question Set**

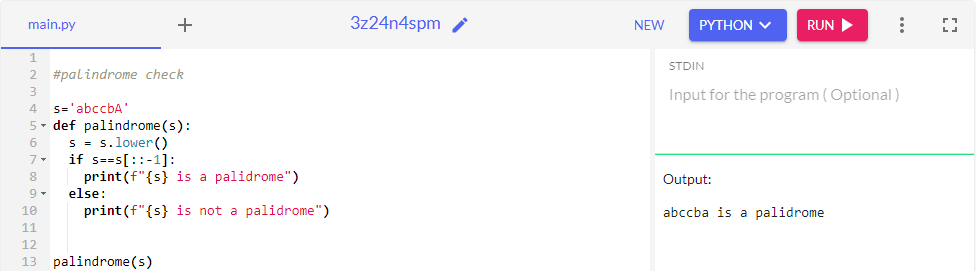
1. **Bracket Validator**

****

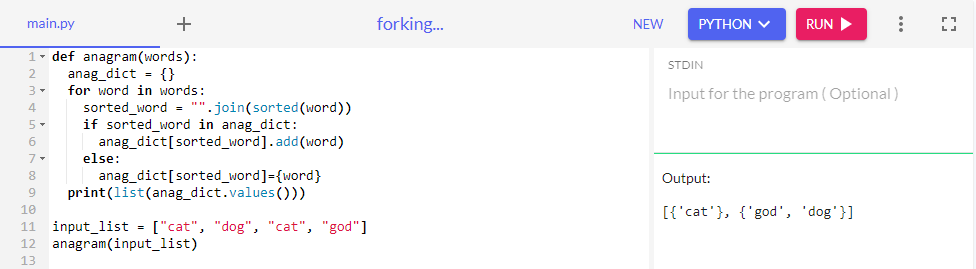
1. **Using Functions (Question is incomplete.)**



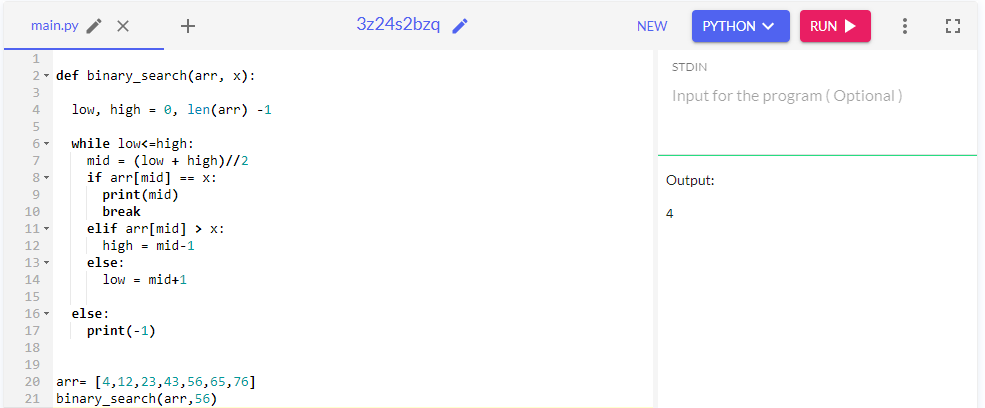
1. **Palindrome Algorithm**

****

1. **Anagram Algo and variations**

****

1. **Binary Search Algorithm Implementation**

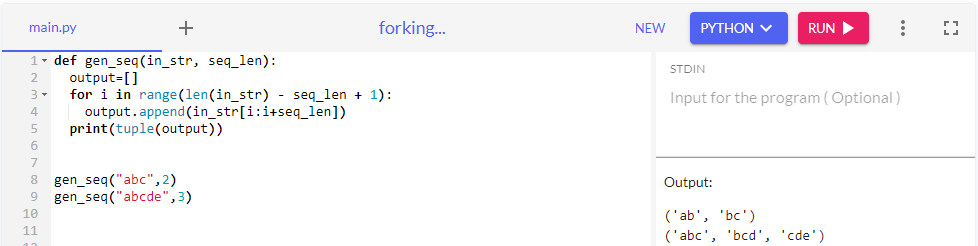
****

1. **Strings 1**

Write a java program to generate the sequences based on the length passed as an input, from the given input string.

For example, if the input string is “abc”, and input length is 2, then the output must be {ab,bc} only (only the consecutive sequences allowed)

If the input is “abcde” and length is 3, then output must be {abc, bcd, cde}

****

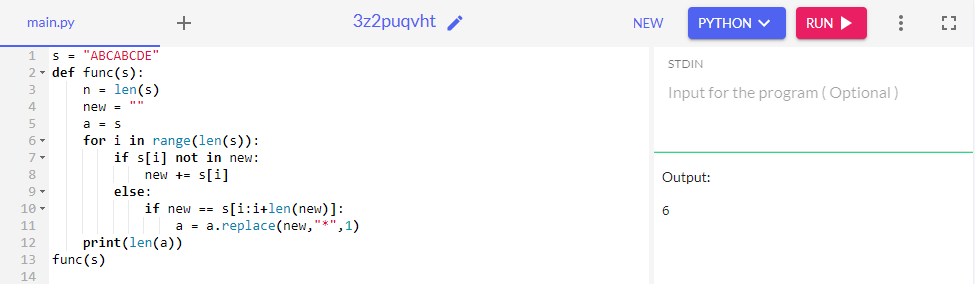
1. **Strings 2**

A magic potion can be made out of the formula when added in a particular order. The formula used can be from {A,B,C,D,E,F,G,H} and in any order

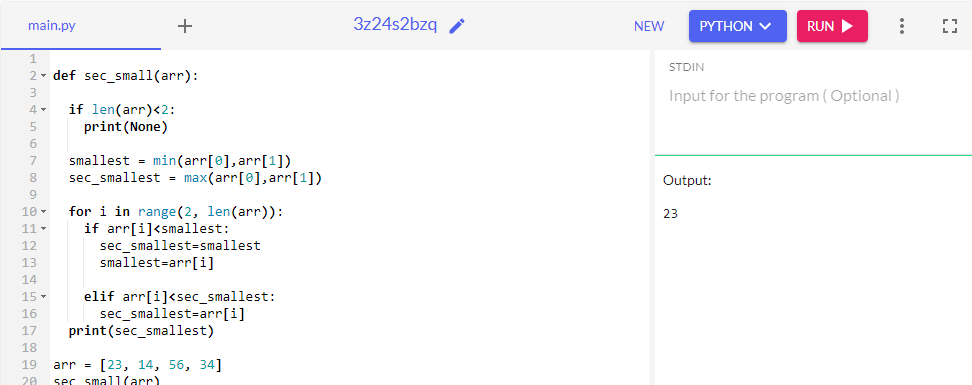
For example, if the formula is made with the following sequence ABCDABCE , then the number of steps to achieve that formula would be 8.

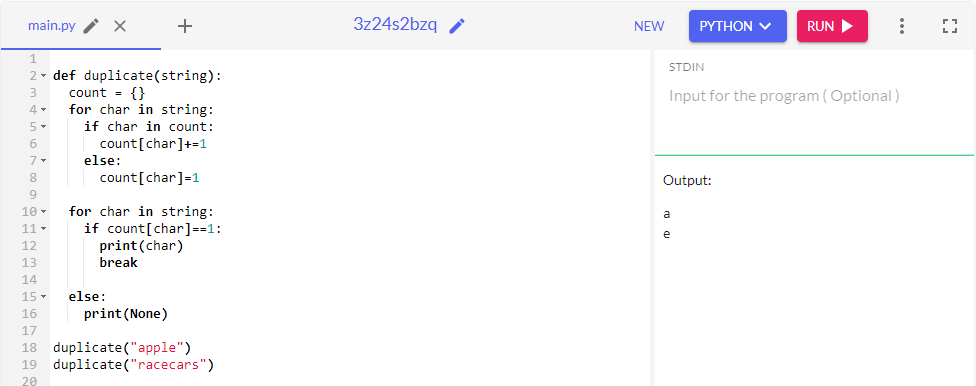
If the input is ABCABCE, then we can place \* for the repeating sequence, ABC\*E and the number of steps to achieve the formula would be 5.

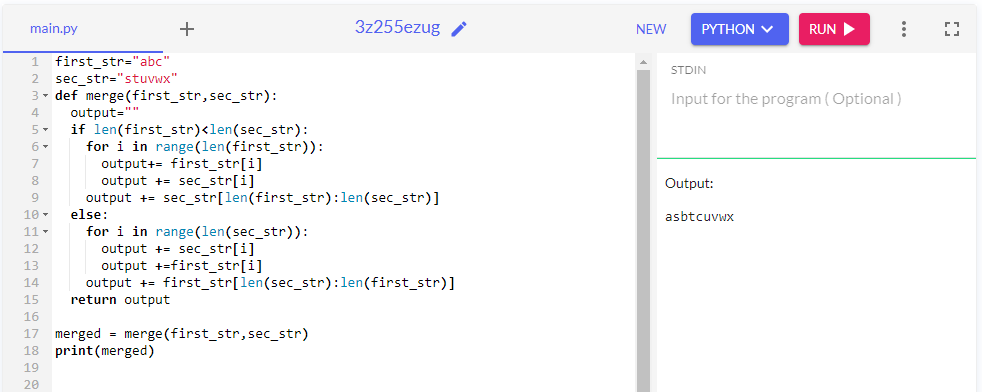
Write a program for achieving the same

****

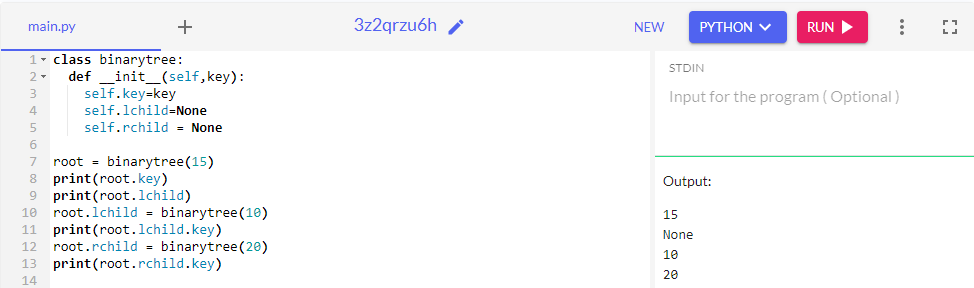
1. **Second Smallest element in an array**

****

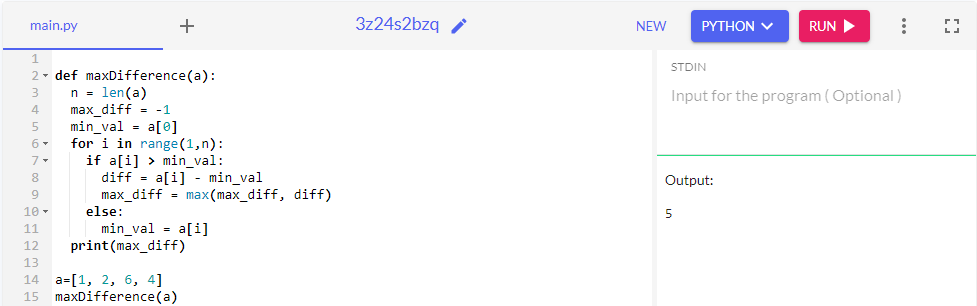
1. **First Non–Duplicate character in string**
2. **String Manipulation**



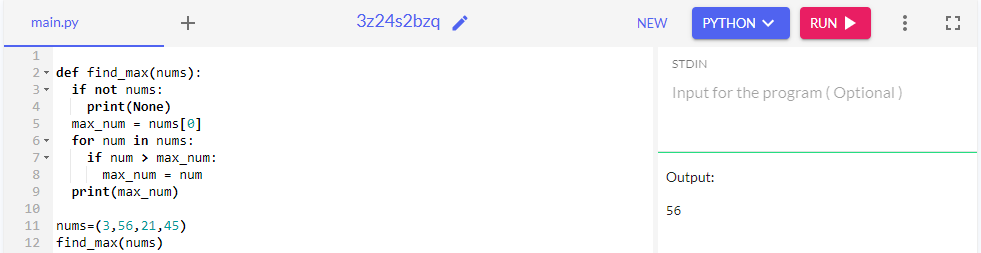
1. **Binary Tree Searching Algorithm and its implementation**



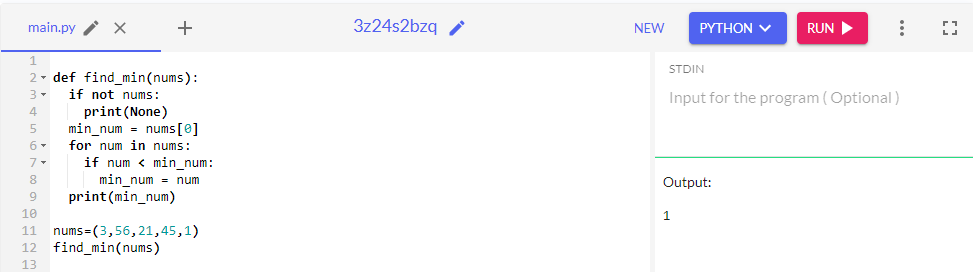
1. **Max Difference**



1. **Min-Max**

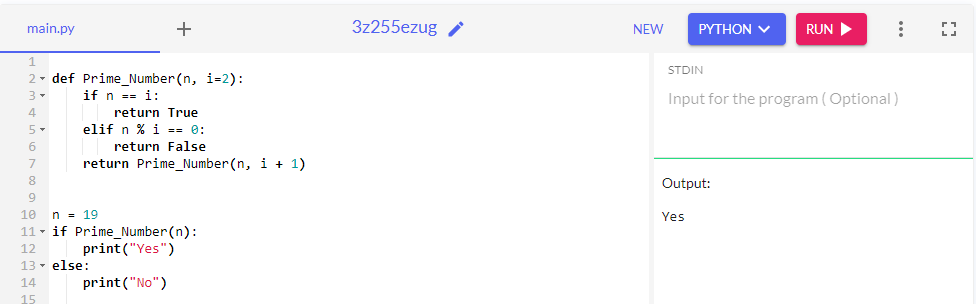


The time complexity is O(n) where n is the length of the list



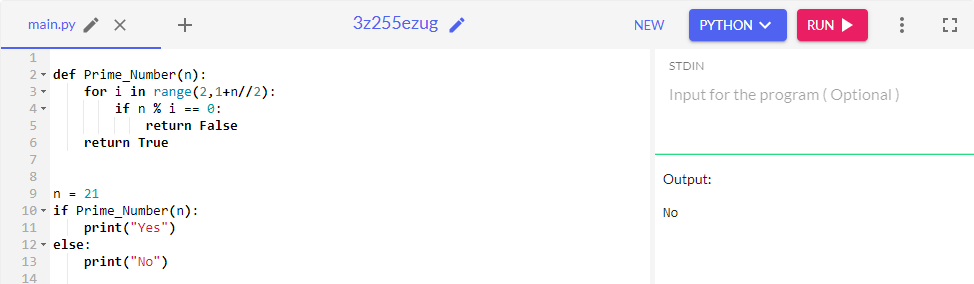
The time complexity is O(n) where n is the length of the list

1. **Prime Number**
2. **With Recurssion**



The time complexity is O(sqrt(n)).

1. **Without Recurssion**

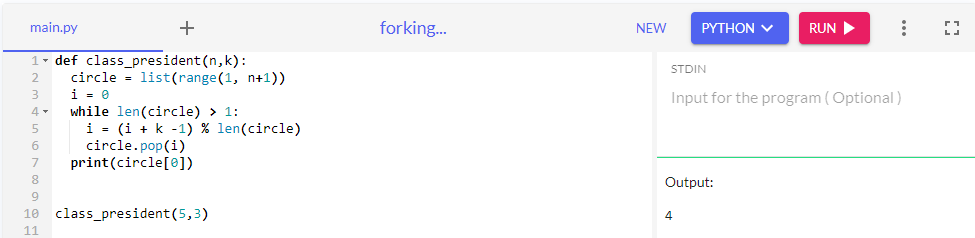


The time complexity is O(sqrt(n)).

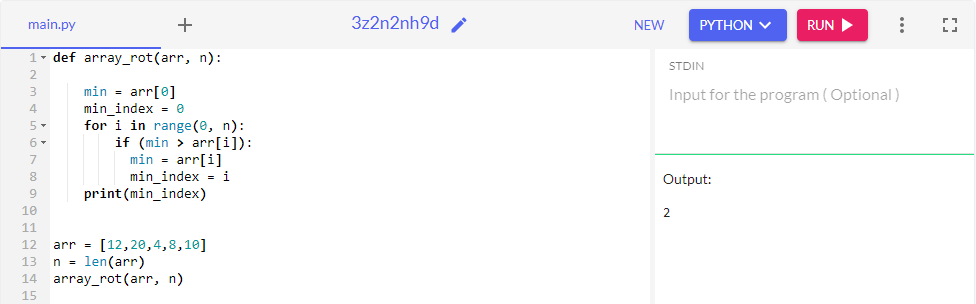
1. **Singly and doubly linked list implementation**

|  |
| --- |
| **Singly** |
|  |
|  |  |
|  | class Node: |
|  | def \_\_init\_\_(self, data=None): |
|  | self.data = data |
|  | self.next = None |
|  |  |
|  | class LinkedList: |
|  | def \_\_init\_\_(self): |
|  | self.head = None |
|  |  |
|  | def search(self, target): |
|  | current = self.head |
|  | while current != None: |
|  | if current.data == target: |
|  | return current |
|  | current = current.next |
|  | return None |
|  |  |
|  | def insert(self, data): |
|  | new\_node = Node(data) |
|  | new\_node.next = self.head |
|  | self.head = new\_node |
|  |  |
|  | def delete(self, target): |
|  | current = self.head |
|  | previous = None |
|  | while current != None: |
|  | if current.data == target: |
|  | if previous == None: |
|  | self.head = current.next |
|  | else: |
|  | previous.next = current.next |
|  | return |
|  | previous = current |
|  | current = current.next |
|  |  |
|  | douby : |
|  |  |
|  | class Node: |
|  | def \_\_init\_\_(self, data=None): |
|  | self.data = data |
|  | self.next = None |
|  | self.prev = None |
|  |  |
|  | class DoublyLinkedList: |
|  | def \_\_init\_\_(self): |
|  | self.head = None |
|  |  |
|  | def search(self, target): |
|  | current = self.head |
|  | while current != None: |
|  | if current.data == target: |
|  | return current |
|  | current = current.next |
|  | return None |
|  |  |
|  | def insert(self, data): |
|  | new\_node = Node(data) |
|  | if self.head == None: |
|  | self.head = new\_node |
|  | else: |
|  | new\_node.next = self.head |
|  | self.head.prev = new\_node |
|  | self.head = new\_node |
|  |  |
|  | def delete(self, target): |
|  | current = self.head |
|  | while current != None: |
|  | if current.data == target: |
|  | if current.prev == None: |
|  | self.head = current.next |
|  | if current.next != None: |
|  | current.next.prev = None |
|  | else: |
|  | current.prev.next = current.next |
|  | if current.next != None: |
|  | current.next.prev = current.prev |
|  | return |
|  | current = current.next |
|  |  |
|  |  |
|  | time compexisties : |
|  |  |
|  | Singly linked list: |
|  |  |
|  | Search: O(n) (where n is the length of the list) |
|  | Insert: O(1) |
|  | Delete: O(n) (where n is the length of the list) |
|  | Doubly linked list: |
|  |  |
|  | Search: O(n) (where n is the length of the list) |
|  | Insert: O(1) |
|  | Delete: O(n) (where n is the length of the list) |
|  |  |

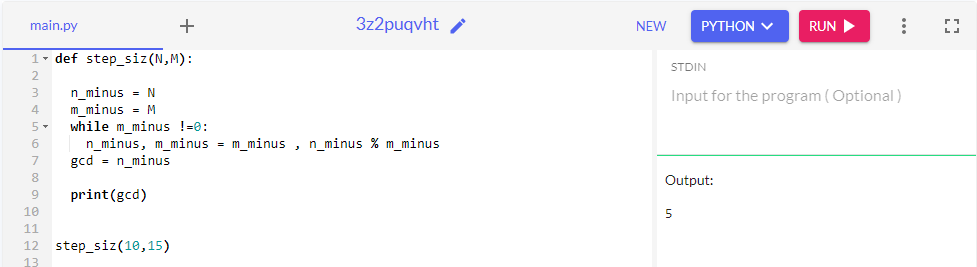
1. **Class President (same solution for Q-31)**



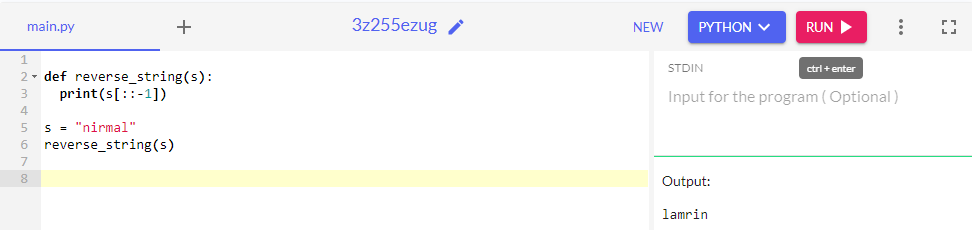
1. **Array Rotation**

****

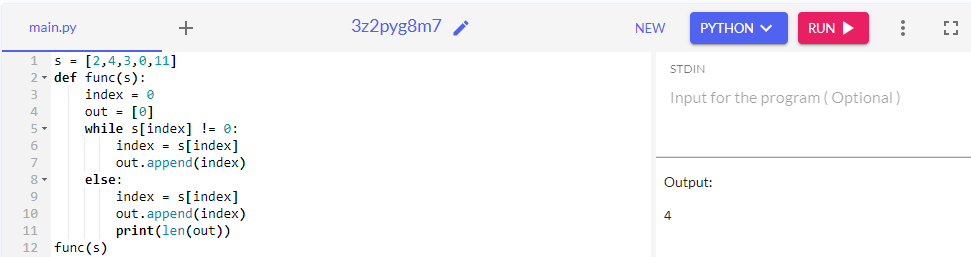
1. **Jumping Game**



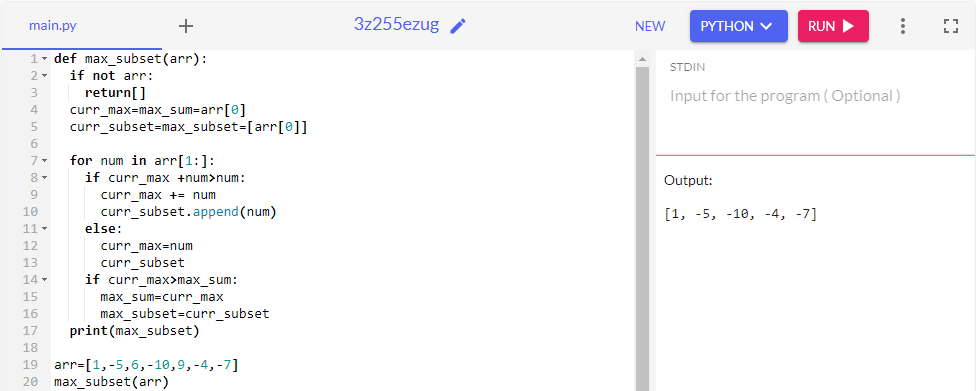
1. **String-Reversal**



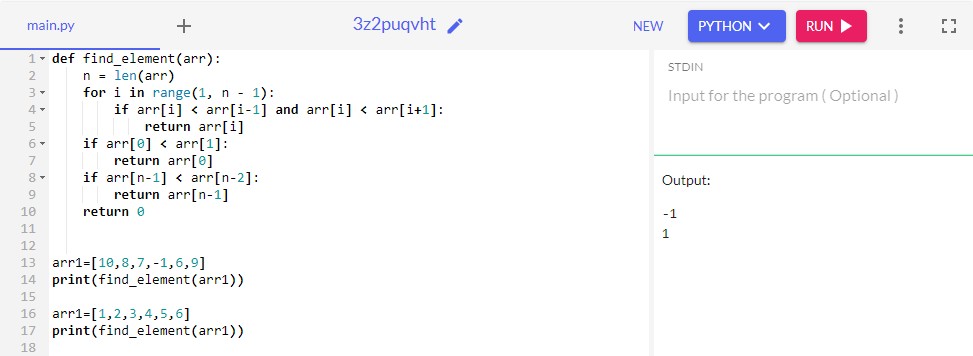
1. **Arrays-1**

****

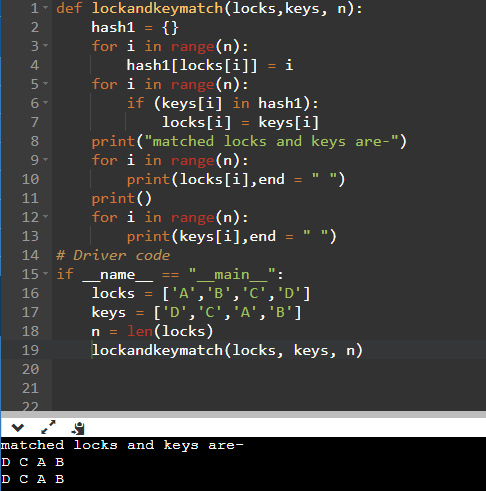
1. **Binary- Trees**
2. **Find Max subset**

****

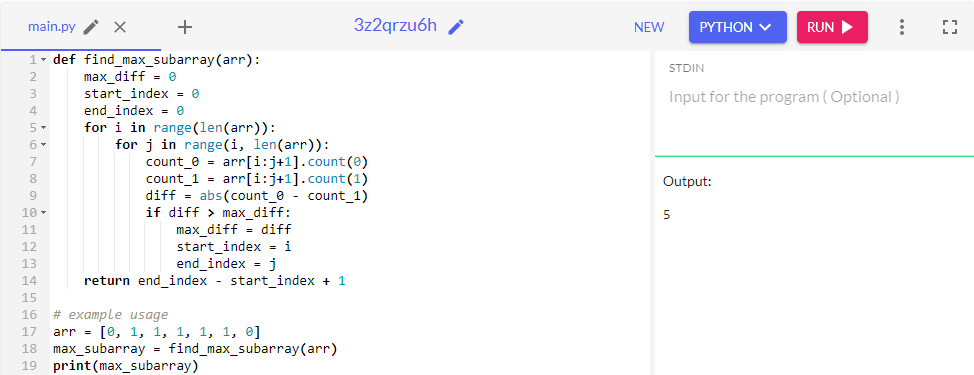
1. **N length of array is given. All the elements are unique (No duplicate). You need to find out such element, that is lesser than its immediate right element as well as immediate left element and return that element, if found. For first element you should only check for its immediate right element and for last element you should only check for its immediate left element. If there is no such element, then return 0. Solve this in optimal way.**

****

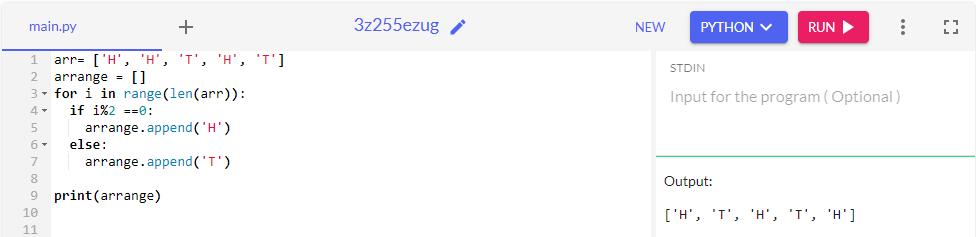
1. **Stack is given, that contains N elements, and all are unique (No Duplicate). We need to find out how many, minimum Push () and Pop () operations are required to sort that stack. Without using any extra space. Solve this in optimal way.**
2. **Four locks and four keys are given, need to find out key for respective lock. All are unique (No duplicate, means for one lock, one key). One lock\_logic(lock,key) function is given that takes lock name and key name and return output as (0 : if lock and key matches, 1 : if key size is greater than lock size, -1 : if key size is lesser than lock size). Solve this in optimal way.**

****

1. **One array is given of size N. That contains only 0 or 1. You need to find out sub-array that have max() value by using below equations.**

****

1. **Arrange coins alternatively as Heads and Tails with least number of steps taken for it. Given the inputs in format as *“H T H T T T”* for example.**

****

1. **Maximum possible valid time(as in 24hrs digital clock) with given 4 input digits. Sample input: “2 6 1 3”. Valid time could be: a. 13:26, b. 21:36, etc. (same solution for Q-38)**

****

1. **A robot starts from a coordinate (0,0) and can move 1 unit at a time only in the following directions Up, Down, Left, Down denoted as {‘U’,’D’,’L’,’R’} respectively.**

**A string of directions will be passed and the function should return the final coordinate the bot is in.  ( same solution for Q-33)**

****

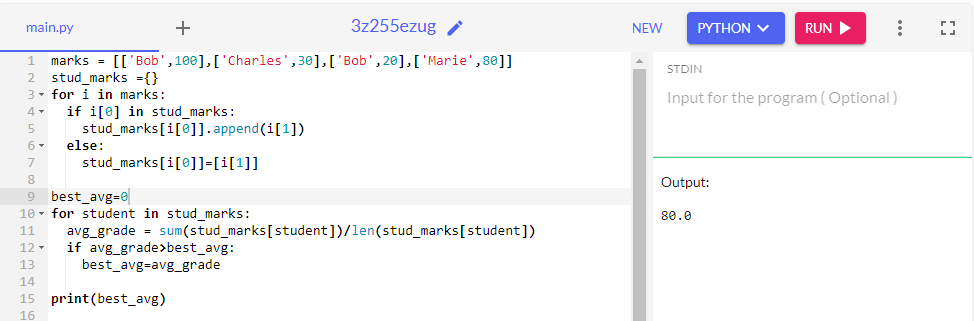
1. **Find the best average grade of a student in a pool of class marks.**

**Given input as [[‘Bob’,100],[‘Charles’,30],[‘Bob’,20],[‘Marie’,80]]**

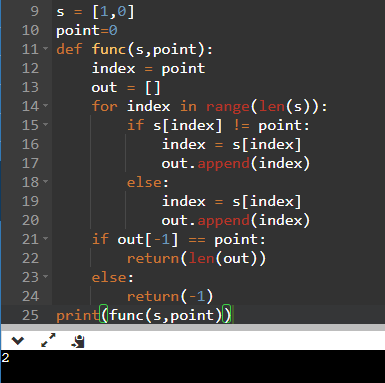
**Output:80**

***Solved. Passed all test cases.***

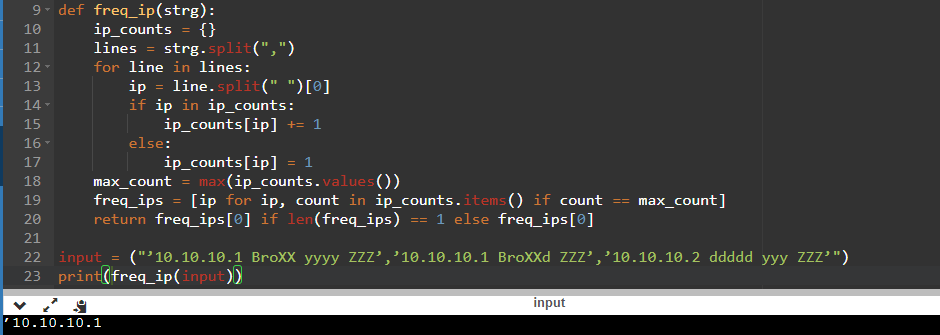
**Asked to optimize the code if the list size is very high: (same solution for Q- 34 )**



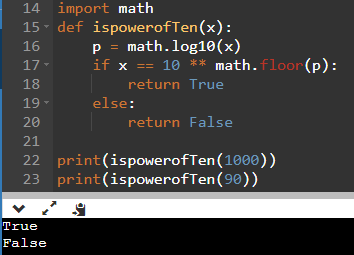
1. **You have a list of N elements and each element is >=0. You need to find length of cycle from given starting index in the list, If no cycle, return -1 else return length of cycle.**

****

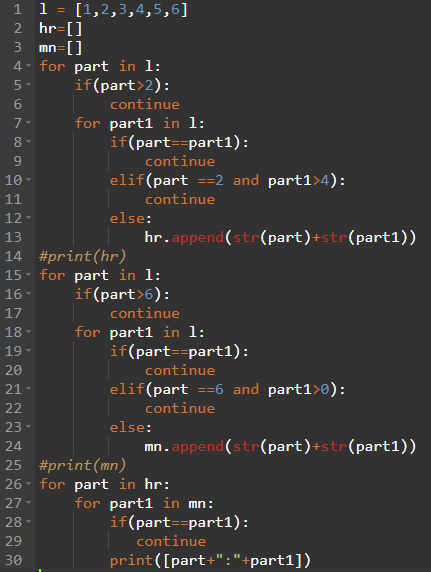
1. **Given a IP information list find the most used IP. In input IP is followed by a a blank space. Ex: input = [’10.10.10.1 BroXX yyyy ZZZ’,’10.10.10.1 BroXXd ZZZ’,’10.10.10.2 ddddd yyy ZZZ’] Output : 10.10.10.1 (repeated two times) Was asked to implement if there were two or more IPs with same number of entries (=max entires) in the file**.**( same solution for Q-36)**

****

1. **Create a function that checks if the number is a power of 10.**

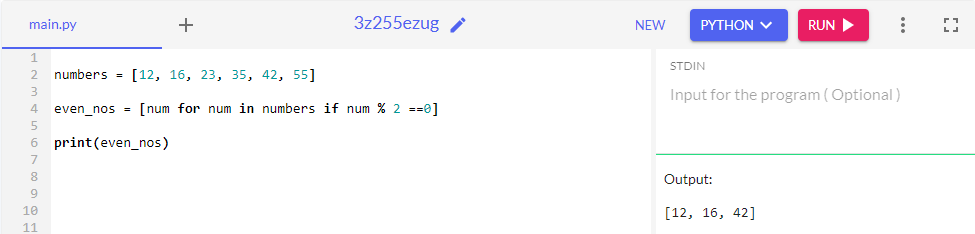
****

1. Given a list, find the all valid timestamps (hrs:mins) in 24hrs digital clock.



Output is list of all the possible timestamps.

1. **How to extract all the even elements in a list?**



1. **What does ‘self’ do in Python?**

**Ans:** Self is a reference to the current instance of a class. It is an implicit parameter that is passed to a class method when the method is called on an instance of the class.

1. **What python datatypes are you aware of?**

**Ans:** I’m aware of the following python datatypes:

Numeric: int, float, complex number

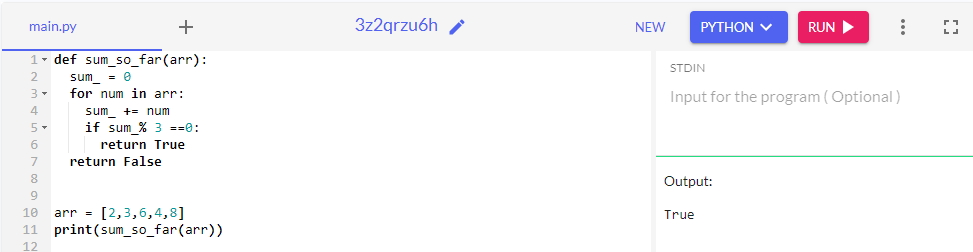
Sequence: List, Tuple, String

Boolean

Set

Dictionary

1. **Suppose we are getting numbers as an input that is dynamic (size is not defined) and we need to find out whether the sum of all inputs received so far, is divisible by 3 or not, and return a Boolean value (True or False). How can we do this?**

**Ans: **