Problem Solving with Python

Priyansh Singh

Practice Questions

Q1. Given a number N. Find the next perfect cube greater than N. Perfect cube is a number whose cube root is a whole number.

Practice Questions

Q2. Write a program to produce the following patterns for the given input.

```
A
B B
C C C
D D D D
```

Practice Questions

Q3. Write a program to produce the following patterns for the given input.

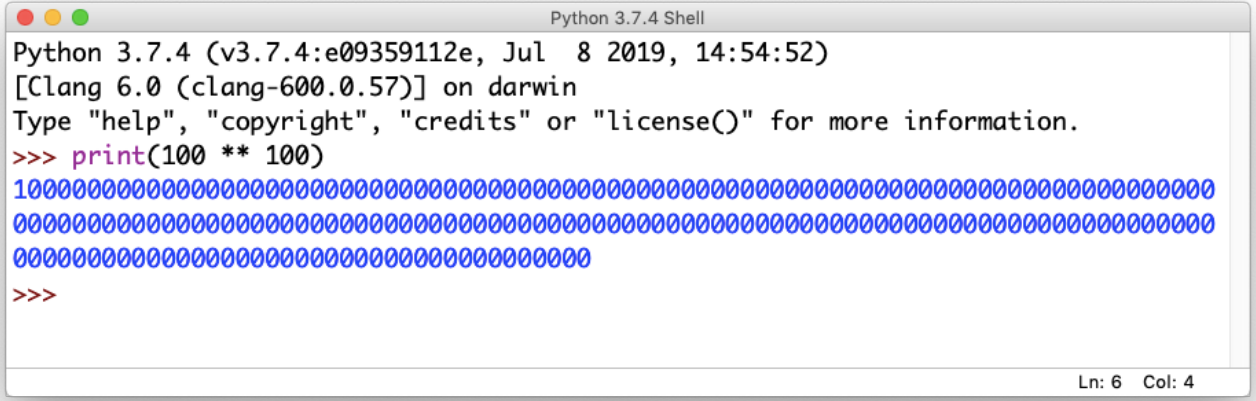
```
P     M
 R   A
  O R
   G
  O R
 R    A
P      M
```

Numeric Types

There are three distinct numeric types: integers, floating point numbers, and complex numbers. In addition, booleans are a subtype of integers.

In Python, **value of a numeric type is not restricted by the number of bits and can expand to the limit of the available memory**.

eg.



Examples of Numeric Types

Integer: 48, -48, 0x260, -0x260, 0o41, -0o131, 0b111001000, -0b11100110.
Float: 3.9, -45.6, 32.3E18, 23.19E-3
Complex: 2j, 2.3j, 3 + 4j

When a binary arithmetic operator has operands of different numeric types, the operand with the narrower type is widened to that of the other, where integer is narrower than floating point, which is narrower than complex.

Examples of Numeric Types

- When the type of the input numbers are same the result will be a number of the same type. **Except the case of division**.

Python 3.7.4 Shell

```
Python 3.7.4 (v3.7.4:e09359112e, Jul  8 2019, 14:54:52)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> res = 3/2
>>> print(type(res))
<class 'float'>
>>> |
```

Ln: 7 Col: 4

Sequence Types

- There are six sequence types - **strings, byte sequences** (bytes objects), **byte arrays** (bytearray objects), **lists, tuples, and range**.
- Strings, lists, tuples and range are the most frequently used.

Strings

- Strings are a special type of sequence that can only store characters.
- Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.
- **Python does not have a character data type**, a single character is simply a string with a length of one.

Sequence Types

- You can embed single quote within a double quoted string and double quote within a single quoted string. Backward slash can be used to escape quotes. Triple quotes can be used for strings spanning multiple lines.

Python 3.7.4 Shell

```
>>>
>>> string = "Welcome to Python Class"
>>> str1 = 'Hello World'
>>> str2 = ''' ABES Engineering College
Ghaziabad '''
>>> print (string, str1, str2)
Welcome to Python Class Hello World  ABES Engineering College
Ghaziabad
>>>
```

Ln: 18 Col: 0

Sequence Types

- Strings are immutable (once created they cannot be modified).

```
Python 3.7.4 Shell
>>>
>>>
>>> string = "Welcome to Python Class"
>>> string[1] = 'r'
Traceback (most recent call last):
  File "<pyshell#20>", line 1, in <module>
    string[1] = 'r'
TypeError: 'str' object does not support item assignment
>>>
Ln: 31    Col: 4
```

- String objects can be created by string literals or by calling built-in functions like str().

```
Python 3.7.4 Shell
>>>
>>>
>>> string = str("Welcome to Python Class")
>>> print (string)
Welcome to Python Class
>>>
>>>
Ln: 39    Col: 4
```

Lists

- List can be written as a list of comma-separated values (items) between square brackets.
- Important thing about the list is that items in a list need not be of the same type.
- Lists are mutable (they can be changed).

Using the type constructor

Python 3.7.4 Shell

```
Python 3.7.4 (v3.7.4:e09359112e, Jul  8 2019, 14:54:52)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> mystring = "Python"
>>> mylist = list(mystring)
>>> print(mylist)
['P', 'y', 't', 'h', 'o', 'n']
>>> mylist = list(range(5))
>>> print(mylist)
[0, 1, 2, 3, 4]
>>>
```

Ln: 11 Col: 4

Python 3.7.4 Shell

```
>>>
>>> input_string = input("Enter a list element separated by space ")
Enter a list element separated by space 1 2 3 4 5 6 7 8 6 5 4 3 2 1
>>> lis1 = input_string.split()
>>> print(lis1)
['1', '2', '3', '4', '5', '6', '7', '8', '6', '5', '4', '3', '2', '1']
>>> a = [int(x) for x in input().split()]
1 2 4 3 2 1
>>> print(a)
[1, 2, 4, 3, 2, 1]
>>>
```

Ln: 81 Col: 4

Using list comprehensions.

- **Using list comprehensions.**
- List comprehensions provide a concise way to create lists.
- Common applications are to make new lists where each element is the result of some operations applied to each member of another sequence or iterable, or to create a subsequence of those elements that satisfy a certain condition.
- Syntax is

```
new_list = [expression for_clause [multiple for or if clauses]]
```

Example of List Comprehension

Python 3.7.4 Shell

```
>>>
>>> squares = [x**2 for x in range(5)]
>>> print(squares)
[0, 1, 4, 9, 16]
>>>
>>> list_b = [0, 2, 8, 9, 28]
>>> list_a = [1, 2, 3, 9, 18]
>>> common = [a for a in list_a for b in list_b if a == b]
>>> print(common)
[2, 9]
```

Ln: 94 Col: 4

Example of List Comprehension

Python 3.7.4 Shell

```
>>> # Prints numbers from 11 to 19
>>> my_list = [i for i in range(100) if i > 10 if i < 20]
>>> print(my_list)
[11, 12, 13, 14, 15, 16, 17, 18, 19]
>>>

# Return numbers from the list which are not equal as a tuple
>>> list_a = [1, 2, 3]
>>> list_b = [0, 2, 8]
>>> different = [(a, b) for a in list_a for b in list_b if a != b]
>>> print(different)
[(1, 0), (1, 2), (1, 8), (2, 0), (2, 8), (3, 0), (3, 2), (3, 8)]
>>>

# Apply a function to a list (similar to map())

>>> my_list = ['quick', 'brown', 'fox', 'jumps']
>>> modified = [str.upper() for str in my_list]
>>> print(modified)
['QUICK', 'BROWN', 'FOX', 'JUMPS']
>>> |
```

Ln: 121 Col: 4

Lists Functions

- list.append(*x*) Add an item to the end of the list. Equivalent to a[len(a):] = [x].
- list.extend(*iterable*)Extend the list by appending all the items from the iterable. Equivalent to a[len(a):] = iterable.
- list.insert(*i*, *x*)Insert an item at a given position. The first argument is the index of the element before which to insert
- list.remove(*x*)
- list.pop([*i*])

Lists Functions

- list.clear() Remove all items from the list. Equivalent to del a[:].
- list.index(*x*[, *start*[, *end*]]) Return zero-based index in the list of the first item whose value is equal to *x*. Raises a ValueError if there is no such item.
- list.count(*x*) Return the number of times *x* appears in the list.
- list.sort(*key=None*, *reverse=False*) Sort the items of the list in place (the arguments can be used for sort customization, see sorted() for their explanation).
- list.reverse() Reverse the elements of the list in place.
- list.copy() Return a shallow copy of the list. Equivalent to a[:].

Q. Get the largest and smallest numbers, and the average of all the numbers in a given list, tuple and dictionary. In order to find the smallest use two different approaches - using built in function (min) and looping and compare the performance of the two approaches.

Tuple

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The major differences between tuples and lists are - tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets. Tuples are constructed by the comma operator (not within square brackets), with or without enclosing parentheses.

Python 3.7.4 Shell

```
>>>
>>>
>>> tuple1 = 'name', 'address', 45, 23.6
>>> tuple2 = ('name', 'address', 45, 23.6)
>>> print(tuple1)
('name', 'address', 45, 23.6)
>>> print(tuple2)
('name', 'address', 45, 23.6)
>>>
```

Ln: 130 Col: 4

Why do we need Tuples, when we have Lists?

- Immutable objects allow substantial optimization. For the same amount of data tuples take less space than lists. See example below

Python 3.7.4 Shell

```
>>>
>>> tuple1 = ('quick', 'brown', 'fox', 'jumps')
>>> list1 = ['quick', 'brown', 'fox', 'jumps']
>>> sys.getsizeof(tuple1)
88
>>> sys.getsizeof(list1)
104
>>> |
```

Ln: 158 Col: 4

Tuples

- Operations with tuples are much faster than operations with lists. The performance difference can be partially measured using the timeit library which allows you to time your Python code. The code below, which simply creates a tuple and a list consisting of exactly the same elements, is run 1 million times and its total time of execution is measured using the timeit library.

```
import timeit
timeit.timeit('x=(1,2,3,4,5,6,7,8,9,10,11,12)', number=1000000)
timeit.timeit('x=[1,2,3,4,5,6,7,8,9,10,11,12]', number=1000000)
```

```
0.02018076300737448
0.1307151880027959
```

Operations Applicable to All Sequences

Inclusion Check

- The operator in returns True if the given item is part of the given sequence and vice-versa. The operator not in returns True if the given item is not part of the given sequence and vice-versa. If this is applied over a string this operation is equivalent to substring test.

```
string = 'healthy wealthy and wise'
my_list = ['wise', 48, 29.3]
substring = 'health'
substring in string
substring not in string
substring  in my_list
```

```
True
False
False
```

Concatenation

- The operators + and * concatenates given sequences. The asterisk operator creates a **shallow copies** of the sequence and appends them to the sequence.

- A *shallow copy* constructs a new compound object and then (to the extent possible) inserts *references* into it to the objects found in the original.

- A *deep copy* constructs a new compound object and then, recursively, inserts *copies* into it of the objects found in the original.

Concatenation

```
str_one = 'hello'
str_two = 'python'
print(str_one + str_two)
print(str_one * 3)
```

```
hellopython
hellohellohello
```

- **Concatenating immutable sequences always results in a new object**. This means that building up a sequence by repeated concatenation will have a quadratic runtime cost in the total sequence length. If we are dealing with strings, it is recommended to use str.join(seq) method instead, which assures consistent linear concatenation performance across versions and implementations. A TypeError will be raised if the argument is non string type.

```
str_one = 'hello'
str_two = 'python'
print(str_one.join(str_two))
```

```
hellopython
```

Element Access

- Using square brackets and 0 based indexing individual elements can be accessed.

```
tuple1 = (‘health’, 48, 23.9, ‘wealth’)
print(tuple1[2])
```

```
23.9
```

Slicing

- Getting sub-sequences is called slicing.
- Subsequences from start to end (not included) with a step of step can be obtained by using square brackets and colon operator.
- The general format is seq[start:end:step].

Slicing

- The default value of start, end and step are 0, len(seq) and 1 respectively.
- **Negative indices are counted from the end and are not 0 based. That is, -3 is actually the 3rd last character**.
- See examples below assuming the string is = 'healthy wealthy and wise'. **It is to be noted that slicing returns an altogether new sequence.**

`string[3:6]`	`althy`	Substring from 3rd index to 5th index.
`string[3:]`	`lthy wealthy and wise`	Substring from 3rd index to the last.
`string[:6]`	`health`	Substring from start to the 5th index.
`string[6:-3]`	`y wealthy and w`	Substring from 6th index to the 4th last character.
`string[-1:6]`		Makes no sense. Will not print anything.
`string[-15:-3]`	`ealthy and w`	Substring from 15th last to 4th last character.
`string[2:10:3]`	`ahw`	Substring consisting of every 3rd character, starting from 2nd and closest to 10th character but never including it.
`string[-1:3:-3]`	`ewnylwh`	Starting from the last character moves towards the character at 3rd index, picking every 3rd character. Moves closest to the character at third index but does not include it (last character is not included)

Length, Minimum and Maximum

- The methods to obtain length, minimum and maximum values from a sequence are len(seq), min(seq) and max(seq).

- The min and max methods are applicable only when the items in the sequence are homogeneous, otherwise TypeError is thrown.

- Characters and strings are compared lexicographically. Integers and floats are compared using their values. Arbitrary objects are compared using their __eq__ method (if the class has implemented the method properly).

Element Location and Count

- To look for the index of first occurrence of a given item we can use seq.index(item) method.
- This method returns the first occurrence of the item in the sequence.
- If the item is not found in the list the method throws ValueError.
- The method seq.count(item) returns the count of item in the given sequence.

Unpacking

- Python provides for a powerful feature called unpacking, wherein the elements of any iterable can be assigned to a set of variables.

- Please note that this feature is available to all iterables (not just sequences). See the below code to understand. If the number of variables on the left hand side is not equal to the values to be unpacked ValueError is raised.

```python
my_list = [24, 'quick fox', 24.9]
my_string = 'adt'

# 24, 'quick fox' and 24.9 will be assigned to x, y and z
respectively.
x, y, z = my_list
print(x, y, z)

# 'a', 'd' and 't' will be assigned to x, y and z
respectively.
x, y, z = my_string
print(x, y, z)
```

```
24 quick fox 24.9
a d t
```

```python
my_list = [24, 'quick', 'fox', 24.9]
my_string = 'jumping jack'

# 24 and 'quick' will be assigned to x and y respectively
# and other elements will be assigned to z.
x, y, *z = my_tuple
print(x, y, z)

# 24 and 24.9 will be assigned to x and z respectively
# and other elements will be assigned to y.
x, *y, z = my_tuple
print(x, y, z)

# 'j' and 'k' will be assigned to a and z respectively
# and others will be assigned to y.
x, *y, z = my_string
print(x, y, z)
```

```
24 quick ['fox', 24.9]
24 ['quick', 'fox'] 24.9
j ['u', 'm', 'p', 'i', 'n', 'g', ' ', 'j', 'a', 'c'] k
```

Operations Applicable to Mutable Sequences

- List is a mutable type sequence, whereas string and tuples are immutable type.

- Following operations are applicable to mutable type sequences.

- In all the following examples positive indices are 0 based and are counted from the start, whereas negative indices are 1 based and are counted from the end.

seq[i] = x	Item at i^{th} index is replaced by x.
del seq[i:j] seq[i:j] = []	Removes the slice from i to j.
del seq	Removes the complete sequence.
seq.append(x)	Appends the element x to the seq.
seq.extend(seq2)	Adds all the items of seq2 to seq.
seq[i:j] = seq2	Replaces the specified slice with the contents of seq2. It actually deletes the slice from i to j and then inserts the contents of seq2 in the same position.
seq[i:j:k] = seq2	Replaces the specified slice with the contents of seq2. The length of seq2 should be exactly equal to the length of slice being replaced. With k = 1, the statement becomes equivalent to the previous one and hence the condition of equality is not required.

seq.insert(i, x)	Inserts the item x at the ith index. Out of bound values of i are replaced with len(seq) – 1 or 0, whichever is closest.
seq.pop(i)	Returns the ith item and deletes it from the sequence. If i is omitted it pops the last item.
seq.remove(x)	Deletes the x from the list. If x is not found it throws ValueError.
seq.reverse()	Reverses the items of seq in place.
seq.sort(reverse=True|False, key=my_func)	Sorts the given sequence. The default value of reverse is False. my_func is the custom comparison function. The default value of key does lexicographical comparison with strings and value comparison with integers and floats.If default value of key is used the sequence should be homogeneous otherwise TypeError will be thrown.

Practice Questions

Given an array of integers, find sum of its elements

Practice Questions

Given an array, find the largest element in it.

```
Input : arr[] = {10, 20, 4}
Output : 20

Input : arr[] = {20, 10, 20, 4, 100}
Output : 100
```

Practice Questions

Given a list, write a Python program to swap first and last element of the list.

Practice Questions

Given two strings **a** and **b**. The task is to find if a string **'a'** can be **obtained** by **rotating** another string **'b'** by **2 places**.

Practice Questions

Given an array A containing n integers. The task is to check whether the array is Monotonic or not. An array is monotonic if it is either monotone increasing or monotone decreasing.

Practice Questions

Given a list of words in Python, the task is to remove the Nth occurrence of the given word in that list.

Practice Questions

Take a list as user input and reverse it.

Practice Questions

Given a list of numbers, the task is to write a Python program to find the smallest number in the given list.

Practice Questions

Given a list of integers with duplicate elements in it. The task to generate another list, which contains only the duplicate elements. In simple words, the new list should contain the elements which appear more than one.

```
Input : list = [10, 20, 30, 20, 20, 30, 40, 50, -20, 60, 60, -20, -20]
Output : output_list = [20, 30, -20, 60]

Input :  list = [-1, 1, -1, 8]
Output : output_list = [-1]
```

Practice Questions

A left rotation operation on an array shifts each of the array's elements 1 unit to the left.

Given an array *a* of *n* integers and a number, *d* , perform *d* left rotations on the array. Return the updated array to be printed as a single line of space-separated integers.

```
for i in range (0,m): matrix[i] = columns
```

```
matrix = [] for i in range(0,m):
        matrix.append([])
        for j in range(0,n):
                matrix[i].append(0)
```

```
matrix = [[0 for j in range(n)] for i in range(m)]
```

```python
n=int(input("Enter N for N x N matrix : ")) #3 here
l=[] #use list for storing 2D array
#get the user input and store it in list (here IN : 1 to 9)
for i in range(n):
        row_list=[] #temporary list to store the row
        for j in range(n):
                row_list.append(int(input())) #add the input to row list
        l.append(row_list) #add the row to the list
        print(l) # [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
#Display the 2D array
for i in range(n):
        for j in range(n):
                print(l[i][j], end=" ")
        print()
```

Practice Questions

We define an hourglass in A (6x6) to be a subset of values with indices falling in this pattern in arr 's graphical representation:

$$\begin{matrix} a & b & c \\ & d & \\ e & f & g \end{matrix}$$

There are 16 hourglasses in arr, and an hourglass sum is the sum of an hourglass' values. Calculate the hourglass sum for every hourglass in arr , then print the maximum hourglass sum.

Practice Questions

Given an array **C** of size **N-1** and given that there are numbers from **1** to **N** with one element missing, the missing number is to be found.

Other Functions Applicable to Strings Only

- replace(old, new [, count]):Replaces count occurrences of old string with the new string. If count not specified, replaces all occurrences.

- title(), capital(), lower(), islower() and isupper(): Returns a new string in title case, capital case and lower case respectively. The last two methods return True if all the characters in the string are lower and smaller respectively.

- swapcase(): Returns another string with all uppercase characters converted to lowercase and vice versa of the given string.

Other Functions Applicable to Strings Only

- isalpha() and isalnum() and isdigit(): Returns True if the string has only alphabets (abc..zABC..Z). The second function returns True if the string has only alphabets (abc..zABC..Z) and/or digits (12..0), without decimal point. The third function returns True if the string has digits only, without a decimal point.

- strip([str]), lstrip([str]) and rstrip([str]): Strip str from both the ends, leading end or the trailing end respectively. If no argument is provided, whitespaces (including tabs and newlines) are stripped. See the examples below.

```python
str = ' brown'
str.strip(' br')
str.strip('br')
str.strip()
```

```
'own'
' brown'
'brown'
```

Practice Questions

Given a string S. The task is to print all permutations of a given string.

Practice Questions

Given a string, write a python function to check if it is palindrome or not. A string is said to be palindrome if reverse of the string is same as string. For example, “radar” is palindrome, but “radix” is not palindrome.

Practice Questions

Given a string, write a Python program to check if that string is Pangram or not. A pangram is a sentence containing every letter in the English Alphabet.

Input : *The quick brown fox jumps over the lazy dog*
Output : *Yes*

Practice Questions

Given an array and split it from a specified position, and move the first part of array add to the end.