Interning

Introduction

Interning is re-using objects of equal value on-demand instead of creating new objects. This is done for memory efficiency. Frequently used for numbers and strings in different programming languages.

```
a = 120
b = 120

print(a is b) # True

c = 2000
d = 2000

print(c is d) # False
```

In the above code, 120 is intered by the Python interpreter but not 2000. Python's integer interning is done only for numbers in the range: [-5, 256]

Python interpreter also interns small strings.

```
a = "abcd"
b = "abcd"

print(a is b) # True

# Both text are the same
c = "Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequentur perferendis iste ipsa nat
d = "Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequentur perferendis iste ipsa nat
print(c is d) # False
```

Strings in python can be manually interned using sys.intern function.

```
a, b=8, 8
c=8
d=8
```

Likewise, in the above code, only 1 integer object is created.

Practice Resources

Programs

The programs are listed in no specific order.

- 1. **is prime number**: A program that takes in a number n and outputs whether its a <u>prime</u> number or not.
- 2. **factors**: Take in a number from user. Output all of its factors.
- 3. **n-th factorial**: A program that takes in a number $m{n}$ and outputs n-th <u>factorial</u>.
- 4. **is perfect number**: A program that takes in a number n and outputs whether its a perfect number.
- 5. **fibonacci numbers**: A program that takes in a number n and prints all <u>fibonacci numbers</u> less than or equal to n.
- 6. **determinant of matrix**: Take in a matrix from user. Output the determinant of the matrix. First try for 2×2 . Then go higher-ordered matrices.
- 7. **pascal's triangle**: Take n from user input. Print <u>pascal's triangle</u> to n rows.
- 8. **is valid palindrome**: Take a string input from user. Output if the input is palindrome or not. A phrase is a palindrome if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers. Try not to use [::-1].
- 9. **armstrong numbers**: Take n from user input. Print all <u>armstrong numbers</u> (in base 10, of course) between 0 and n (inclusive).
- 10. **letter analysis**: Take a text input from user. Find how many times each letter is being used in that string. Use a dictionary to store the data. Output the final results. Try to read the text from a .txt file as well.
- 11. **word length analysis**: Take a string input from user. Print length of each word separated by a space. Try to include the summary using a dictionary.
- 12. **letter expanding**: A program that converts *b3j8k2* to *bbbjjjjjjjjkk*. The number can be 1 to 99.
- 13. **binary addition**: Take in 2 numbers in binary (as strings) and output the sum of both numbers. Try not to use bin function.
- 14. big integer addition: Given a very large integer represented as a list, where each $_{\rm digits[i]}$ is the $i^{\rm th}$ digit of the integer. The digits are ordered from most significant to least significant

in left-to-right order. Increment the large integer by one and return the resulting array of digits. Don't construct a <code>int</code> object.

Platforms

- Codewars https://codewars.com (my most preferred one)
- HackerRank https://hackerrank.com
- Leetcode https://leetcode.com (my least preferred one)

⚠ Hard Problems

If a problem from one of these platforms feels too hard for you, you can just skip and do another problem.

This PDF is saved from https://s1.sahithyan.dev