Sets and methods

January 31, 2023

1 str.split()

- Split the given string based on the given delimiter
- The delimiter can be any string
- Split() always returns a list as output

[7]:
```
a, b, c = map(int, input().split())
print(a + b + c)
```

```
10 20 30
60
```

[8]:
```
lst = list(map(int, input().split()))
print(lst)
```

```
10 20 30
[10, 20, 30]
```

[9]:
```
s = '10 20 30'
print(s.split()) # delimiter = space by default
```

```
['10', '20', '30']
```

[ ]:
```
10, 20, 30, 40
```

[17]:
```
lst = list(map(int, input().split(', ')))
print(lst)
```

```
10, 20, 30
[10, 20, 30]
```

[13]:
```
s = '10, 20, 30'
print(s.split())
```

```
['10,', '20,', '30']
```

[15]:
```
int('10,')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_21320\4041501396.py in <cell line: 1>()
```

```
----> 1 int('10,')

ValueError: invalid literal for int() with base 10: '10,'
```

[19]:
```
# H:M:S = 4:15:57
H, M, S = map(int, input().split(":"))
print(f'Hours{H}\nMinutes{M}\nSeconds{S}')
```

```
4:15:57
Hours4
Minutes15
Seconds57
```

[20]:
```
s = 'this is python'
words = s.split() # returns a list
print(words)
```

```
['this', 'is', 'python']
```

[21]:
```
s = 'this is python'
words = s.split('is') # returns a list
['th', ' ', ' python']
print(words)
```

```
['th', ' ', ' python']
```

[28]:
```
def count_vowels(string: str) -> int:
    # returns the count of vowels in the string
    cnt = 0
    for i in string:
        if i in 'aeiouAEIOU':
            cnt += 1
    return cnt

s = 'this is python'
# How many vowels are present in each word
words = s.split()
print(words)
vowels_count = [count_vowels(word) for word in words]
print(vowels_count)
```

```
['this', 'is', 'python']
[1, 1, 1]
```

[29]:
```
s = 'python'
words = s.split(' ')
print(words)
```

```
['python']
```

2 str.join()

- Joint string.join(iterable) # an iterable of strings

[34]:
```
words = ['this', 'is', 'python']
sentence = ' '.join(words)
print(sentence)
```

```
this is python
```

[35]:
```
lst_nums = [10, 20, 30]
print(''.join(lst_nums))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_21320\1872902486.py in <cell line: 2>()
      1 lst_nums = [10, 20, 30]
----> 2 print(''.join(lst_nums))

TypeError: sequence item 0: expected str instance, int found
```

3 sets

- Set is an unordered collection of immutable elements which are unique
- List of sets –>
- Set of lists –> Doesn't work because lists are mutable
- Set of strings –> Because strings are immutable
- Sets will never hold a duplicate value
- We cannot subscript a set using indexes
- We can create an empty set using set() function

[36]:
```
lst = [10, 20, 30, 30, 20, 10]
print(lst)
```

```
[10, 20, 30, 30, 20, 10]
```

[37]:
```
s = {10, 20, 30, 30, 20, 10, 40, 40, 50, 50}
print(s)
```

```
{50, 20, 40, 10, 30}
```

[38]:
```
s = {'a', 'b', 'b', 'a', 'c', 'c', 'd'}
print(s)
```

```
{'b', 'c', 'd', 'a'}
```

[39]:
```
s = {[10, 20], [20, 10], [10, 20]}
print(s)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_21320\140942372.py in <cell line: 1>()
----> 1 s = {[10, 20], [20, 10], [10, 20]}
      2 print(s)

TypeError: unhashable type: 'list'
```

[40]:
```
s = {{10, 20, 10}, {20, 10, 10}, {20, 20, 20}}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_21320\3666460485.py in <cell line: 1>()
----> 1 s = {{10, 20, 10}, {20, 10, 10}, {20, 20, 20}}

TypeError: unhashable type: 'set'
```

[41]:
```
s = {(10, 20), (20, 10), (10, 20)}
print(s)
```

```
{(10, 20), (20, 10)}
```

[ ]:
```
# union
# intersection
# difference
# update
# intersection_update
# difference_update
# symmetric_difference
# symmetric_difference_upate
```

[42]:
```
s1 = {10, 20, 30}
s2 = {20, 30, 40}
s3 = s1.union(s2) # s2.union(s1)
print(s3)
```

```
{20, 40, 10, 30}
```

[44]:
```
s1 = {10, 20, 30}
s2 = {20, 30, 40}
s3 = s1.intersection(s2) # s2.intersection(s1)
print(s3)
```

```
{20, 30}
```

[45]:
```
s1 = {10, 20, 30}
s2 = {20, 30, 40}
s3 = s1.difference(s2) # elements present in s1 but not in s2
print(s3)
```

```
{10}
```

[46]:
```
s1 = {10, 20, 30}
s2 = {20, 30, 40}
s3 = s2.difference(s1) # elements present in s2 but not in s1
print(s3)
```

```
{40}
```

[48]:
```
# Symmetric Difference
# Every element in s1 and s2 except the intersection
# symmetric difference (s1.union(s2)).difference((s1.intersection(s2)))
s1 = {10, 20, 30}
s2 = {20, 30, 40}
s3 = s1.symmetric_difference(s2)
s4 = (s1.union(s2)).difference((s1.intersection(s2)))
print(s3)
print(s4)
```

```
{40, 10}
{40, 10}
```

[51]:
```
string = 'this is tutorial on sets'
s = {character for character in string}
print(s)
print(len(string))
print(len(s))
```

```
{'a', 'h', 's', 'o', ' ', 'i', 'r', 'u', 'l', 't', 'n', 'e'}
24
12
```

[ ]:
```
union --> update
intersection --> intersection_update
difference --> difference_update
symmetric_difference --> symmetric_difference_update
```

[52]:
```
s1 = {10, 20, 30}
s2 = {20, 30, 40}
s3 = s1.union(s2)
print(s3)
```

```
{20, 40, 10, 30}
```

[56]:
```
s1 = {10, 20, 30}
s2 = {20, 30, 40}
s1.update(s2)
# update updates the set on the left hand side with the result (union)
print(s1)
```

```
{20, 40, 10, 30}
```

[58]:
```
s1 = {10, 20, 30}
s2 = {20, 30, 40}
s2.intersection_update(s1)
print(s1)
print(s2)
```

```
{10, 20, 30}
{20, 30}
```

4 GCD of two numbers

–> 12
–> 18
–> Greatest Common Divisor
–> 12 –> 1 2 3 4 6 12
–> 18 –> 1 2 3 6 9 18
–> CD –> 1 2 3 6
–> GCD –> 6

[64]:
```
a = int(input())
b = int(input())
factors_a = {i for i in range(1, a + 1) if a % i == 0}
factors_b = {i for i in range(1, b + 1) if b % i == 0}
common_factors = factors_a.intersection(factors_b)
gcd = max(common_factors)
print(gcd)
```

```
5
7
1
```

4.1 s.add()

[66]:
```
s = {10, 20}
s.add(30)
s.add(40)
s.add(50)
print(s)
```

```
{40, 10, 50, 20, 30}
```

[69]:
```
s = {40, 10, 50, 20, 30}
s.pop()
print(s)
s.pop()
print(s)
s.pop()
print(s)
```

```
{20, 40, 10, 30}
{40, 10, 30}
{10, 30}
```

[70]:
```
s = {40, 10, 50, 20, 30}
s.remove(40)
print(s)
```

```
{50, 20, 10, 30}
```

[71]:
```
s = {40, 10, 50, 20, 30}
s.remove(100)
print(s)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_21320\807592327.py in <cell line: 2>()
      1 s = {40, 10, 50, 20, 30}
----> 2 s.remove(100)
      3 print(s)

KeyError: 100
```

[74]:
```
s = {40, 10, 50, 20, 30}
s.discard(40) # not shows any error if the element is not present
print(s)
```

```
{50, 20, 10, 30}
```

[75]:
```
s = {40, 10, 50, 20, 30}
s.discard(100) # not shows any error if the element is not present
print(s)
```

```
{50, 20, 40, 10, 30}
```