# Strings

### String Index: From start to end

| H | e | l | l | o | | W | o | r | l | d | ! |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

### String Index: From end to start (Negative Indexing)

| H | e | l | l | o | | W | o | r | l | d | ! |
|---|---|---|---|---|---|---|---|---|---|---|---|
| -12 | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

```
# Intialize a string

string = "Hello World!"
string[0:4]
```

```
'Hell'
```

```
string[1]
```

```
'e'
```

```
string = 'Hello World!'

string[0]
```

```
'H'
```

```
# Length of the string
len(string)
```

```
12
```

```
len("nidgfwegf")
```

```
9
```

## Empty string

```
emptyString =''

len(emptyString)
```

```
0
```

```
type(emptyString)
```

```
str
```

# Slicing

Notation - **[StartIndex:EndIndex]**

## Index of string 'batman':

| b | a | t | m | a | n |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

Negative indexing of string 'batman'

| b | a | t | m | a | n |
|---|---|---|---|---|---|
| -6 | -5 | -4 | -3 | -2 | -1 |

```
word = 'batman'
print(word[3:5])
```

```
ma
```

```
word[-3:-1]
```

```
'ma'
```

```
#To get 3 last character, end index goes one index beyond

print(word[3:6])
```

```
man
```

## Index assumed

```
print(word[3:]) # End is assumed
```

```
man
```

```
print(word[:3]) # Start is assumed
```

```
bat
```

```
#Negative numbering

# Get last Charater
print(word[-1])
```

```
n
```

```
#Get last 3 Character using negative indexing

print(word[-3:])
```

```
man
```

```
print(word[-3:-4]) # (Incorrect Indexing) cannot interpret the indexing start index must have lower value than end index
```

## Format Specification

### Field width

- If the values are less than the size of the given field, spaces are added
- Numbers will be right-aligned
- Any other type will be left-aligned

```
#Add Field width

print(f'{"Student Name":20}{"Marks":10}')
print('-'*30)
print(f'{"Malcola Reynolds":20}{60:10}')
print(f'{"Zoe Wasburne":20}{70:10}')
```

```
Student Name        Marks
------------------------------
Malcola Reynolds            60
Zoe Wasburne                70
```

## Alignment

- Left-aligned **<**
- Right-aligned **>**
- Centered **^**

```
# Marks Center Aligned

print(f'{"Student Name":20}{"Marks":^10}')
print('-'*30)
print(f'{"Malcola Reynolds":20}{60:^10}')
print(f'{"Zoe Wasburne":20}{70:^10}')
```

```
Student Name          Marks
------------------------------
Malcola Reynolds        60
Zoe Wasburne            70
```

```
# Marks Left Aligned

print(f'{"Student Name":20}{"Marks":10}')
print('-'*30)
print(f'{"Malcola Reynolds":20}{60:<0}')
print(f'{"Zoe Wasburne":20}{70:<10}')
```

```
Student Name        Marks
------------------------------
Malcola Reynolds    60
Zoe Wasburne        70
```

```
# Marks Right Aligned

print(f'{"Student Name":20}{"Marks":>10}')
print('-'*30)
print(f'{"Malcola Reynolds":20}{60:>10}')
print(f'{"Zoe Wasburne":20}{70:>10}')
```

```
Student Name             Marks
------------------------------
Malcola Reynolds            60
Zoe Wasburne                70
```

## Fill Characters

- By default space is added as fill(pad)
- You can also assign fill character
- Can be used in conjunction with the alignment character to determine if the fill characters appear before (>), after (<) or around (^)

```
#Adding prefix

print(f'{10:0>6}')
```

```
000010
```

```
#Adding prefix
```

```
print(f'{"Test"}')

print(f'{"Test":-^2}')#Here the padding character is A
```

```
Test
Test
```

```
#How it helps?
print(f'{"Student Details":^26}')
print('-'*26)
print(f'{"Name":<20}{"ID":^6}')
print('-'*26)
print(f'{"Malcola Reynolds":<20}{345:0>6}')
print(f'{"Zoe Wasburne":<20}{1070:0>6}')
```

```
      Student Details
--------------------------
Name                  ID
--------------------------
Malcola Reynolds    000345
Zoe Wasburne        001070
```

## Numeric Precision

```
x = 4.89387766
print(f'{x:.3f}')
```

```
4.894
```

```
x = 4.89387766
print(f'{x:0>6.3f}')
```

```
04.894
```

## String Methods

### replace()

```
title = 'The new adventures of Indiana Jones'

print(title.replace('new','continuing')) #title.replace('new','continuing') creates new string
```

```
The continuing adventures of Indiana Jones
```

```
title #Immutable
```

```
'The new adventures of Indiana Jones'
```

```
print(title.replace('test','continuing'))
```

```
The new adventures of Indiana Jones
```

```
phrase = 'This is very,ververy,y,very long'

print(phrase.replace('very,',''))
```

```
This is very long
```

```
print(phrase.replace('very,','',2)) # Replace first 2 occurrence of 'very,'
```

```
This is very,very long
```

### find()

```
phrase = 'This is very,very,very,very long'
```

```
phrase.find('This') # Returns staring index of the substring
```

```
0
```

```
phrase.find('very') # Returns staring index of first occurrence of the substring 'very'
```

```
8
```

```
phrase.find('test') #Returns -1 if the substring is not present
```

```
-1
```

```
phrase.find('very',9) # Returns staring index of first occurrence of the substring 'very' after index 9
```

```
13
```

```
phrase.find('is',3,20) # Returns staring index of first occurrence of the substring 'is' between index 3 and 20
```

```
5
```

```
phrase.rfind('is')# Search substring from last index
```

```
5
```

```
phrase.rfind('very',-11,-2) #Re
```

```
23
```

## String Comparision

- Strings can be compared using:
  - relational operators (<, <=, >=, >)
  - equality operator (==, !=)
  - Membership operators (in, not in )
  - Identity operators (is, is not)
- Comparison uses the encoded values of the characters (ASCII/Unicode)

```
print(ord('a'))
print(ord('A'))
print(ord('$'))
```

```
97
65
36
```

```
'$' > 'a'
```

```
False
```

```
'a' == 'A'
```

```
False
```

```
'Apple' < 'app'
```

```
True
```

In above code the comparison starts with the first characters of each string: 'A' and 'a'. Since 'A' has a lower Unicode value than 'a'. Thus results TRUE.

```
'bat' > 'ball'
```

```
True
```

```
print(ord('t'))
print(ord('l'))
```

```
116
108
```

The above code compares the string 'bat' with the string 'ball' using the less than operator (<).

Python compares characters one by one.

- 'b' is equal to 'b'
- 'a' is equal to 'a'
- 't' has a higher Unicode value than 'l'.

Thus, 'bat' < 'ball' will result in FALSE

## Membership checks for substring's presence in string

```
'bat' in 'batman'
```

```
True
```

```
'bat' not in 'batman'
```

```
False
```

## Identity operator

```
string1 = "apple"
string2 = "Apple"
check = string1 is string2 #Python checks if two variables refer to the same object in memory
print(check)
```

```
False
```

```
string1 = "apple"
string2 = "apple"
check = string1 is string2 #Python checks if two variables refer to the same object in memory
print(check)
```

```
True
```

> What happens in below code? ⏰ 😉

```
string1
```

```
'apple'
```

```
string2
```

```
'Apple'
```

```
string1 == string2.lower()
```

```
True
```

```
string1 is string2.lower()
```

```
False
```

## Iterate through the string

> using for loop

```
word ='batman'
for char in word:
  print(char)
```

```
b
a
t
m
a
n
```

## any() function returns True if at least one element in the iterable is True

- It checks if any element in the iterable is True.
- It stops as soon as it finds the first True value.

```
word = 'batman'
any(char == 'a' for char in word)
```

```
True
```

```
any(char == 'x' for char in word)
```

```
False
```

## all() function returns True only if all elements in the iterable are True. If any element is False, it returns False.

- It checks if all elements in the iterable are True.
- It stops as soon as it finds the first False value.

```
word = 'batman'
all(char == 'a' for char in word)
```

```
False
```

```
word = 'aaa'
any(char == 'a' for char in word)
```

```
True
```

## other string comparsion methods

- isalnum() – returns True if all characters are lowercase or uppercase letters or numbers 0-9
- isdigit() – returns True is all characters are numbers 0-9
- islower() – returns True if all characters are lowercase
- isupper() if all characters are uppercase
- isspace() – returns True if all characters are whitespace
- startswith(x) – returns True if it starts with this character
- endswith(x) – returns True if it ends with this character

## isalnum() – returns True if all characters are lowercase or uppercase letters or numbers 0-9

```
print( 'abc123'.isalnum())
```

```
True
```

```
print( 'abc123$!'.isalnum())
```

```
False
```

isdigit() – returns True is all characters are numbers 0-9

```
print( 'abc123'.isdigit())
```

```
False
```

islower() – returns True if all characters are lowercase

isupper() -returns True if all characters are uppercase

```
print('abc123$'.islower())
print('ABC123'.isupper())
```

```
True
True
```

```
print('aBc123$'.islower())
```

```
False
```

isspace() – returns True if all characters are whitespace

```
print('    '.isspace())
```

```
True
```

```
print('   ABC 123    '.isspace())
```

```
False
```

```
test = "rtty iuidyuew.  "

any(char.isspace() for char in test)
```

```
True
```

startswith(x) – returns True if it starts with this character/substring

endswith(x) – returns True if it ends with this character/substring

```
print('abc123'.startswith('a'))
```

```
True
```

```
print('abc123'.startswith('ab'))
```

```
True
```

```
print('abc123'.endswith('24'))
```

```
False
```

Other important string menthod often used for string manipulation - these string method results new string

- capitalize() returns a copy of the string with the first character capitalized
- lower() or upper() will lowercase or uppercase the string
- strip() removes any leading or trailing whitespace
- title() returns a string with the first letter of every word capitalized

```
phrase = "frozen is my FAVOURITE movie!!!  "
```

```
print(phrase.capitalize()) #  first character capitalized and other characters will be lower
```

```
Frozen is my favourite movie!!!
```

```
print(phrase.lower()) # All characters will be lower
```

```
frozen is my favourite movie!!!
```

```
print(phrase.upper()) # All characters will be upper
```

```
FROZEN IS MY FAVOURITE MOVIE!!!
```

```
print(phrase.title()) # First character of every word will be upper
```

```
Frozen Is My Favourite Movie!!!
```

```
print(phrase.strip()) # Remove any leading or trailing whitespace
```

```
frozen is my FAVOURITE movie!!!
```

```
print("Actual Length:", len(phrase))
print("After Stripping:" ,len(phrase.strip()))
```

```
Actual Length: 34
After Stripping: 31
```

```
print("!!hey!hey!!!".strip('!')) # Remove '!' in leading or trailing positions
```

```
hey!hey
```

## Chaining string Methods

```
print("!!hey!hey!!!".strip('!').upper())
```

```
HEY!HEY
```

## split()

The split() method splits a string into a list of tokens.

- Each **token** is a substring
- A separator is a **character** (or sequence of characters) that indicates where to split the string into a list of tokens
- **By default – the split uses a blank space** as the separator

```
phrase = "I love to watch frozen, Despicable Me, Free Birds "
```

```
print(phrase.split()) #the split uses a blank space as the separator
```

```
['I', 'love', 'to', 'watch', 'frozen,', 'Despicable', 'Me,', 'Free', 'Birds']
```

```
print(phrase.split(','))
```

```
['I love to watch frozen', ' Despicable Me', ' Free Birds ']
```

## join()

This method joins a list of strings around a designated separator

```
list =['https', 'www', 'google', 'com']
charSep = '/'
print(charSep.join(list))
```

```
https/www/google/com
```

## To do Challenge

- Get a email ID & password as input
- Your program should to convert the email ID given as input to lower character and strip any blanks in leading and trailing
- Get password as input from user and check for its validity, conditions for valid password is given below
  - Password must have atleast 8 character
  - Password must contain atleast 1 lower case letter
  - Password must contain atleast 1 upper case letter
  - Password must contain atleast 1 numeric character

## Answer

```
email = input("Enter Email ID: ").lower().strip()
password = input("Enter Password: ")

print(email)

if len(password) >= 8 and any(char.islower() for char in password) and any(char.isupper() for char in password) and any(char.isdi
  print("Password is valid")
else:
  print("Password is invalid")
```

```
Enter Email ID: Aish@gmail.com
Enter Password: Ais@123
aish@gmail.com
Password is invalid
```