Table of contents

---

1. Convert to string: `str()`

(go to top)

In [3]:
```
c = 5.6
print(type(c))
```

```
<class 'float'>
```

In [6]:
```
d = str(c)
print(d)
print(type(d))
```

```
5.6
<class 'str'>
```

2. Strings in Loops / Sequences

(go to top)

In [7]:
```
message = "Random"

for i in message:
    print(i)
```

```
R
a
n
d
o
m
```

In [13]:
```
e = 'python'
list(e)
```

Out[13]:
```
['p', 'y', 't', 'h', 'o', 'n']
```

3. Strings when using `input()`

(go to top)

In [9]:
```
x, y = input("Enter the coodrinates (x,y): ").split(',')
```

```
Enter the coodrinates (x,y): 5,9
```

In [10]:
```
month, day, year = input("Enter date as mm/dd/yyyy: ").split('/')
```

```
Enter date as mm/dd/yyyy: 03/17/2021
```

In [11]:
```
print(x,y)
```

```
5 9
```

In [12]:
```
print(month, day, year)
```

```
03 17 2021
```

---

4. Slicing / Indexing

(go to top)

- slice produces the substring starting at the position given by start and running up to, but not including, position end.

In [1]:
```
message = 'random message'
```

In [2]:
```
message
```

Out[2]:
```
'random message'
```

In [3]:
```
message[0]
```

Out[3]:
```
'r'
```

```
In [5]: message[-1]

Out[5]: 'e'

In [31]: message[1]

Out[31]: 'a'

In [6]: message[-2]

Out[6]: 'g'

In [32]: message[2]

Out[32]: 'n'

In [33]: message[3]

Out[33]: 'd'

In [7]: message[:]

Out[7]: 'random message'

In [34]: message[0:1]

Out[34]: 'r'

In [35]: message[0:2]

Out[35]: 'ra'

In [36]: message[:2]

Out[36]: 'ra'

In [37]: message[1:3]

Out[37]: 'an'

In [38]: message[1:4]

Out[38]: 'and'
```

In [39]:
```
print(message[0],  message[3])
```

```
r d
```

In [40]:
```
print(message[0:13:1])
```

```
random messag
```

In [41]:
```
print(message[0:13:2])
```

```
rno esg
```

5. Backslash

(go to top)

In [14]:
```
f = 'this is a \n new line'
print(f)
```

```
this is a
 new line
```

In [15]:
```
g = 'how to add a backslash \\ to a string'
print(g)
```

```
how to add a backslash \ to a string
```

6. Raw format

(go to top)

- If you have a string with a lot of backslashes and no special characters, you might find this a bit annoying. Fortunately you can preface the leading quote of the string with r, which means that the characters should be interpreted as is:

In [44]:
```
h = r'this\has\no\special\xters'
print(h)
```

```
this\has\no\special\xters
```

7. Formatting / Templating

(go to top)

In [53]:
```
template = '{0:.02f} {1:s} are worth US${2:d}'
```

• {0:.02f} means to format the first argument as a floating-point number with two decimal places.
• {1:s} means to format the second argument as a string.
• {2:d} means to format the third argument as an exact integer.

In [54]:
```
template.format(4.5560, 'Argentine Pesos', 1)
```

Out[54]:
```
'4.56 Argentine Pesos are worth US$1'
```

In [5]:
```
"Hello {0}, you may have won ${1} {2}".format("dude", 100000, 'lottery
```

Out[5]:
```
'Hello dude, you may have won $100000 lottery'
```

In [38]:
```
"This int, {0:5}, was placed in a field of width 5".format(7)
```

Out[38]:
```
'This int,     7, was placed in a field of width 5'
```

In [26]:
```
'{0:0.02f}'.format(55569.56457)
```

Out[26]:
```
'55569.56'
```

In [46]:
```
'{0:10.02f}'.format(55.56956457)
```

Out[46]:
```
'     55.57'
```

In [43]:
```
'{0:10.02}'.format(55.56956457)
```

Out[43]:
```
'   5.6e+01'
```

In [24]:
```
#if you have. a float result 1.5 but you want to express it as $1.50

'${0:0.02f}'.format(1.5)
```
Out[24]:
```
'$1.50'
```

In [5]:
```
'{0:f}'.format(55569)
```
Out[5]:
```
'55569.000000'
```

In [15]:
```
'{0:d}'.format(556)
```
Out[15]:
```
'556'
```

In [10]:
```
'{0:s}'.format('a string here')
```
Out[10]:
```
'a string here'
```

In [95]:
```
# {index: width.precision}

"Compare {0} and {0:15} || and  {0:0.4f} and {0:0.4} || and {0:0.04f}
```
Out[95]:
```
'Compare 3.1 and             3.1 || and  3.1000 and 3.1 || and 3.1000
and 3.1 || and 3.1000000000000000888'
```

---

8. String Operators

(go to top)

- Concatenation builds a string by "gluing" two strings together.
- Repetition builds a string by multiple concatenations of a string with itself

In [8]:
```
message = 'Random Message'
text = 'Text Text'
```

- Concatenation / Addition

In [48]:
```
addition = message + text
print(addition)
```

```
Random MessageText Text
```

In [9]:
```
print(message + text)
```

```
Random MessageText Text
```

In [49]:
```
addition = message + ' ' + text
print(addition)
```

```
Random Message Text Text
```

- Repetition / Multiplication

In [50]:
```
multiplication = message * 2
print(multiplication)
```

```
Random MessageRandom Message
```

---

9. String Methods

(go to top)

In [1]:
```
message = 'random message'
words = 'JUST TEXT IN UPPER CASE'
```

`len()`

(go to top)

In [17]:
```
print(len(message))
```

```
14
```

.split()

(go to top)

In [3]:
```
student_scores = 'temi 50 60 90'
name, *scores = student_scores.split()
print(name)
print(scores)
print(type(scores))
```

```
temi
['50', '60', '90']
<class 'list'>
```

In [10]:
```
split_msg = message.split()
print(split_msg)
```

```
['Random', 'Message']
```

In [19]:
```
split_msg = words.split(' ')
print(split_msg)
```

```
['JUST', 'TEXT', 'IN', 'UPPER', 'CASE']
```

In [20]:
```
split_msg = words.split('T')
print(split_msg)
```

```
['JUS', ' ', 'EX', ' IN UPPER CASE']
```

```
.split ("\t") for a tab seperated file line
file.readline().split('\t')
```

.capitalize()

(go to top)

In [21]:
```
print(message.capitalize())
print(words.capitalize())
```

```
Random message
Just text in upper case
```

`.upper()` and `.lower()`

(go to top)

```
In [57]: print(message.upper())
         print(words.lower())
```

```
RANDOM MESSAGE
just text in upper case
```

```
In [3]: print(message)
        print(message.isupper())
        print(message.islower())
```

```
random message
False
True
```

'swapcase()'

```
In [6]: print(message)
        print(message.swapcase())
```

```
random message
RANDOM MESSAGE
```

`.title()`

(go to top)

```
In [58]: print(words.title())
         print(message.title())
```

```
Just Text In Upper Case
Random Message
```

`in` and `not in`

(go to top)

In [59]:
```
print('R' in message)
print('R' not in message)
```
```
False
True
```

In [60]:
```
print('R' in words)
print('R' not in words)
```
```
True
False
```

.count()

(go to top)

In [61]:
```
print(message.count('s'))
print(words.count('T'))
```
```
2
3
```

replace()

(go to top)

In [22]:
```
print(message.replace('s','P'))
print(words.replace('T','S'))
```
```
random mePPage
JUSS SEXS IN UPPER CASE
```

.center()

(go to top)

In [17]:
```
print(len(message))
```
```
14
```

```
In [17]: message.center(14)

Out[17]: 'random message'
```

```
In [19]: message.center(15)

Out[19]: ' random message'
```

```
In [20]: message.center(16)

Out[20]: ' random message '
```

```
In [21]: message.center(20)

Out[21]: '   random message   '
```

.ljust()

(go to top)

```
In [48]: message_2 = '    random message'
         message_2

Out[48]: '    random message'
```

```
In [41]: print(len(message_2))

         18
```

```
In [44]: message_2.ljust(19)

Out[44]: '    random message '
```

```
In [46]: message_2.ljust(25)

Out[46]: '    random message       '
```

.find()

(go to top)

```
In [26]: message
Out[26]: 'random message'
```

```
In [24]: message.find('e')
Out[24]: 8
```

```
In [52]: message.find('a')
Out[52]: 1
```

`.rfind()`

(go to top)

- returns the right-most position

```
In [26]: message
Out[26]: 'random message'
```

```
In [49]: message.rfind('e')
Out[49]: 13
```

```
In [53]: message.rfind('a')
Out[53]: 11
```

`.join()`

(go to top)

```
In [27]: message
Out[27]: 'random message'
```

```
In [31]: " ".join('this string is treated like a sequence')
Out[31]: 't h i s   s t r i n g   i s   t r e a t e d   l i k e   a   s e q u
         e n c e'
```

```
In [32]: "||".join(['this',  'list', 'is', 'treated', 'as', 'a', 'sequence'])
Out[32]: 'this||list||is||treated||as||a||sequence'
```

- convert list to string

```
In [1]: " ".join(['this',  'list', 'is', 'treated', 'as', 'a', 'sequence'])
Out[1]: 'this list is treated as a sequence'
```

```
In [3]: print('\n'.join(['this',  'list', 'is', 'treated', 'as', 'a', 'sequenc

this
list
is
treated
as
a
sequence
```

```
In [35]: "JOIN".join(['here', 'here', 'and', 'here', ])
Out[35]: 'hereJOINhereJOINandJOINhere'
```

.lstrip()

(go to top)

- copies the string with leading white pace is removed

```
In [57]: message_2
Out[57]: '    random message'
```

```
In [56]: message_2.lstrip()
Out[56]: 'random message'
```

```
In [58]: x = message_2.lstrip()

         print("this is a", x)

this is a random message
```

`.rstrip()`

(go to top)

- copies the string with leading white pace is removed

In [61]:
```
message_3 = 'random message        '
message_3
```
Out[61]:
```
'random message        '
```

In [62]:
```
message_3.rstrip()
```
Out[62]:
```
'random message'
```

In [64]:
```
y = message_3.rstrip()

print(y, 'has no space')
```
```
random message has no space
```

`.strip()`

(go to top)

- copies the string with white pace is removed

In [15]:
```
s = '   2 5 6 8 6 4 9 3   '
s.strip()
```
Out[15]:
```
'2 5 6 8 6 4 9 3'
```

In [19]:
```
s = 'd d u j i 2 5 6 8 6 4 9 3   '
s.strip('d d u j i')
```
Out[19]:
```
'2 5 6 8 6 4 9 3'
```

In [17]:
```
txt = ",,,,,rrttgg.....banana....rrr"
txt.strip(",.grt")
```
Out[17]:
```
'banana'
```

10. Title

(go to top)

In [61]:
```
lst = []
for i in range(1,int(input())+1):
    lst.append(str(i))
    print(''.join(lst[:(i-1)]) + ''.join(lst[::-1]), )
```

```
5
1
121
12321
1234321
123454321
```

11. Title

(go to top)

In [ ]:

12. Title

(go to top)

In [ ]:

13. Title

(go to top)

```
In [ ]:
```

14. Title

(go to top)

```
In [ ]:
```

15. Title

(go to top)

```
In [ ]:
```

16. Title

(go to top)

In [ ]:

17. Title

(go to top)

In [ ]:

18. Title

(go to top)

In [ ]:

19. Title

(go to top)

In [ ]:

20. Title

(go to top)

In [ ]:

```

```