Chapter 8: Strings

Last Update: April 12, 2023

**Program 1**:

08 11-ord chr.py

```
 1 def next():
 2     input("Next> ")
 3
 4 ch ='7'
 5 print(f'{ch} converts into {ord(ch)-ord("0")}.')
 6 next()
 7
 8 i = 6
 9 print(f'{i} converts into {chr(i+ord("0"))}.')
10 next()
11
12 for i in range(10):
13     print(f'"{i}" -> {i+ord("0")}')
14 next()
 1
16 for i in range(48,58):
17     print(f'{i} -> "{chr(i)}"')
```

**Program 2**:

08 dd-xxxx.py

```
 1 def next():
 2     input("Next> ")
 3
 4 for i in range(26):
 5     ch = chr(i+ord("A")) # int to chr
 6     print(f'character "{ch}" -> ordinal number {ord(ch)}')
 7 next()
 8
 9 for i in range(26):
10     ch = chr(i+ord("a")) # int to chr
11     print(f'character "{ch}" -> ordinal number {ord(ch)}')
12 next()
13
14 for i in range(ord("A"),ord("Z")+1):
15     print(f'Ordinal number {i} -> character "{chr(i)}"')
16 next()
17
18 for i in range(ord("a"),ord("z")+1):
19     print(f'Ordinal number {i} -> character "{chr(i)}"')
```

**Program 3**:

```
08 | dd-xxxx.py
 1 | str = 'Python'
 2 | print(f' 5 [{str.center(5)}]')
 3 | print(f' 6 [{str.center(6)}]')
 4 | print(f' 7 [{str.center(7)}]')
 5 | print(f' 8 [{str.center(8)}]')
 6 | print(f' 9 [{str.center(9)}]')
 7 | print(f'10 [{str.center(10)}]')
 8 | print()
 9 |
10 | test = '12345' # Won't work for integers
11 | print(f'[{test.zfill(10)}]')
12 | print(f'[{test.ljust(10)}]')
13 | print(f'[{test.rjust(10)}]')
14 | print(f'[{test.rjust(10,"0")}]')
15 | print(f'[{test.rjust(10,"*")}]')
```

**Program 4**:

```
08 | 22-clean str.py
 1 | # This one does not work for some cases.
 2 | # Try to find out the reason.
 3 | def next():
 4 |     input("Next> ")
 5 |
 6 | def getStrClean():
 7 |     str = input("Enter a string: ")
 8 |     clean = ''
 9 |     space = True
10 |     for ch in str:
11 |         if ch==' ':
12 |             if not space:
13 |                 clean = clean + ch
14 |             space = True
15 |         else:
16 |             clean = clean + ch
17 |             space = False
18 |     return(clean)
19 |
20 | print(f"[{getStrClean()}]")
21 | next()
```

**Program 5**:

08 23-clean str.py

```
 1 # Trailing blanks
 2 # Better but not perfect
 3 def next():
 4     input("Next> ")
 5
 6 def getStrClean():
 7     str = input("Enter a string: ")
 8     clean = ''
 9     space = True
10     for ch in str:
11         if ch==' ':
12             if not space:
13                 clean = clean + ch
14             space = True
15         else:
16             clean = clean + ch
17             space = False
18     return(clean)
19
20 print(f"[{getStrClean()}]")
21 next()
```

**Program 6**:

08 24-clean str.py

```
 1 # This is the better solution
 2 def next():
 3     input("Next> ")
 4
 5 def getStrClean():
 6     str = input("Enter a string: ")
 7     clean = ''
 8     space = False
 9     for ch in str:
10         if ch==' ':
11             space = True
12         else:
13             if space and clean!='':
14                 clean = clean+' '
15             clean = clean + ch
16             space = False
17     return(clean)
18
19 print(f"[{getStrClean()}]")
20 next()
```

**Program 7**:

08 25-find.py

```
 1 def next():
 2     input('Next> ')
 3
 4 def find1(word, letter):
 5     index = 0
 6     while index<len(word):
 7         if word[index] == letter:
 8             return index
 9         index += 1
10     return None
11
12 def find2(word, letter):
13     for i, ch in enumerate(word):
14         if ch == letter:
15             return i
16     return None
17
18 word = 'Mississippi'
19 letter = 'p'  # Try q
20
21 print("\t *** Case 1")
22 index = find1(word,letter)
23 print(f"Searching for letter '{letter}' in word '{word}'. ")
24 if index is not None:
25     print(f'word[{index}] = {letter}')
26 else:
27     print(f'{letter} not found')
28 next()
29
30 print("\t *** Case 2")
31 index = find2(word,letter)
32 print(f"Searching for letter '{letter}' in word '{word}'. ")
33 if index is not None:
34     print(f'word[{index}] = {letter}')
35 else:
36     print(f'{letter} not found')
37 next()
```

**Program 8**:

08 26-methods.py

```
def next():
    input('> ')

# Calling the find method of the str object
str = "to be or not to be"
print("01234567890123456789")
print(str)
next()
for word in ['be', 'bee', 'to']:
    print(f"Find {word}: {str.find(word)}")
    next()

print("123".isdigit())
next()
print("123".isnumeric())
next()
print("-123".isdigit())
next()
print("-123".isnumeric())
next()
vowels = 'aeiouAEIOU'
str = "to be or not to be"

print(str.replace('be','bee'))
print(str)
next()
for c in str:
    if c in vowels:
        str = str.replace(c,'')
print(str)
```

**Program 9**:

08 27-xxxx.py

```
# Sorting 2d array by multiple columns
def next():
    input("Next> ")

def print_table(table):
    for row in table:
        for elem in row:
            print(f"{elem}", end=', ')
        print()
    print()

# test 1 score, test 2 score, name
list = [
    [80, 100, 'john'], [70, 70, 'bob'], [100, 90, 'matthew'], [70,
100, 'ava'],
    [80, 90, 'kira'], [80, 100, 'irene'], [100, 90,'alice'], [70, 90,
'steve']
]

print("\t*** The original list")
print_table(list)
next()

print("\t*** Case 1: Sort by column 1")
list.sort(key=lambda x:x[1])
print_table(list)
next()

print("\t*** Case 2: sort by column 0, stable, 0 followed by 1")
list.sort(key=lambda x:x[0])
print_table(list)
next()

print("\t*** Case 3: Sort by all columns, 0 followed by 1 followed by
2")
list.sort()
print_table(list)
next()

print("\t*** Case 4: Sort by most improved")
list.sort(key=lambda x:x[1]-x[0], reverse=True)
print_table(list)
next()

print("\t*** Case 5: Sort by most improved, then alphabetically")
list.sort(key=lambda x:[x[0]-x[1], x[2]])
print_table(list)
```

**Program 10**:

```
08  28-sort str.py
 1  def len_alpha(word):
 2      return(len(word), word.casefold())
 3
 4  fruits = ['mango', 'jujube', 'kumquat', 'pear', 'Peach', 'kiwi']
 5  print(sorted(fruits,key=len_alpha))
```

**Program 11**:

```
08  31-split.py
 1  def next():
 2      input("\nNext> ")
 3
 4  str = "This is  a   string for split testing."
 5  print(f"            [{str}]")
 6  next()
 7  print(f"Split()     {str.split()}")
 8  print(f"Split(' ')  {str.split(' ')}")
 9  print(f"Split('  ') {str.split('  ')}")
10  next()
11
12  str = "  This is  a  non-empty string for split testing.  "
13  print(f"            [{str}]")
14  print(f"Split()     {str.split()}")
15  print(f"Split(' ')  {str.split(' ')}")
16  print(f"Split('  ') {str.split('  ')}")
17  next()
18
19  str = "Stephen$Huang$Professor"
20  print(f"                [{str}]")
21  print(f"str.split('$')  {str.split('$')}")
22  print(f"str.split('$',1)  {str.split('$', 1)}")
23  next()
24
25  str = "An apple a day keeps the doctor away. Eat more apple."
26  print(f"[{str}]")
27  print(f"str.split('apple')     {str.split('apple')}")
28  print(f"str.partition('apple') {str.partition('apple')}")
```

**Program 12**:

08 32-strip.py

```
def next():
    input("\nNext> ")

str = "   An apple a day keeps the doctor away.    "
print(f'[{str}]')
print(f'[{str.strip()}]')
print(f'[{str.lstrip()}]')
print(f'[{str.rstrip()}]')
print(f"[{str.strip(' ')}]")
print(f'[{str}]')

next()

str = " xxx,   banana,  apple, cherry, ...   "
print(f'[{str}]')
print(f'[{str.strip()}]')
print(f"[{str.strip('x. y,')}]")
print(f'[{str}]')
```

**Program 13**:

08 33-strip-split.py

```
str = "  Hello,  are all students present today?  "
print(f'The str:     /{str}/')
str1= str.strip('.,? ') # period, comma, question mark, space
words= str1.split()
print(f'After strip: /{str1}/')
print(f'After split: /{words}/')
print(f'The str:     /{str}/')
```

**Program 14**:

08 34-join.py

```
fruits = ['Apple', 'Banana', 'Cherry']
print(f"The list: {fruits}")
print(f"Join by ' ':  {' '.join(fruits)}")
print(f"Join by '.':  {'.'.join(fruits)}")
print(f"Join by ', ': {', '.join(fruits)}")
```

**Program 15**:

08 35-split join.py

```
1  str = input("Enter a string: ")
2  print(f'The Original: [{str}]')
3  print(f'Reformatted:  [{" ".join(str.split())}]')
```

**Program 16**:

08 36-join.py

```
1  import string
2  print(f"{string.punctuation}")
3  str = 'He said: "Don\'t  do   that."'
4  print(f"The str:                [{str}]")
5
6  words = str.split()
7  print(f"After split():          {words}")
8  for i in range(len(words)):
9      words[i] = words[i].strip(string.punctuation)
10 print(f"After strip() & join(): {' '.join(words)}")
```

**Program 17**:

08 41-format methods.py

```
1  def next():
2      input('> ')
3
4  s = 'Python'
5  num = '12345'
6
7  print(f'Original:    [{s}]')
8  print(f'str.center:  {s.center(10)}')
9  print(f'str.center:  {s.center(10, "*")}')
10 print(f'str.ljust:   {s.ljust(10)}')
11 next()
12
13 print(f'Original:    [{num}]')
14 print(f'num.rjust:   {num.rjust(10, "*")}')
15 print(f'num.zfill:   {num.zfill(10)}')
16 print(f'str.zfill:   {s.zfill(10)}')
```

**Program 18**:

```
08 | 42-C style.py
 1 | def next():
 2 |     input('\nNext> ')
 3 |
 4 | name = 'John Smith'
 5 | acct_id = 123456
 6 | balance = 123456.789
 7 | data = (name, acct_id, balance) # a tuple, not a list
 8 | format_string = "Name: %s\nId: %d\nBalance: $%9.2f"
 9 | next()
10 |
11 | # C-Style, associate by position
12 | print("Name: %s\nId: %d\nBalance: $%10.2f" % (name, acct_id,
13 | balance))
14 | next()
15 |
16 | # Using varables
17 | print(format_string % data)
18 | next()
19 |
20 | # Without using data and format_string variables, associate by names
21 | print("Name: %(name)s\nId: %(id)d\nBalance: $%(bal)9.2f" %
22 |       {"name": name, "id": acct_id, "bal": balance})
```

**Program 19**:

08 43-str_format_method.py

```
def next():
    input('> ')

name = 'John Smith'
acct_id = 123456
balance = 123456.789
data = [name, acct_id, balance]
format_string = "Name: {:s}\nId: {:d}\nBalance: ${:9,.2f}"

print("\t *** Case 1: format string variable")
print(format_string.format(name, acct_id, balance))
next()

print("\t *** Case 2: by relative position")
print("Name: {:s}\nId: {:d}\nBalance: ${:9,.2f}".format(name,
acct_id, balance))
next()

print("\t *** Case 3: by explicit position")
print("Name: {0:s}\nId: {1:d}\nBalance: ${2:9,.2f}".format(name,
acct_id, balance))
next()

print("\t *** Case 4: by keyword")
print("Name: {name:s}\nId: {id:d}\nBalance:
${bal:9,.2f}".format(name=name, id=acct_id, bal=balance))
print("Name: {one:s}\nId: {two:d}\nBalance:
${three:9,.2f}".format(one=name, two=acct_id, three=balance))
```

**Program 20**:

08 44-string_template.py

```
name = 'John Smith'
acct_id = 123456
balance = 123456.789

#3 Template
from string import Template

t = Template("Name: $name\nId: $id\nBalance: $$$bal")
print(t.substitute(name=name, id=acct_id, bal=balance))
```

**Program 21**:

08 45-f-string.py

```
def next():
    input('> ')

name = 'Smith'
id = 123456

print( "Name = ", name, ", ID = ", id, sep='')
next()
print(f"Name = {name}, ID = {id}")
next()
print(f"Name = [{name}]\nID = [{id}]\n")
next()
print(f"Name = [{name:10s}]\nID   = [{id:<10d}]\n")
next()
print(f"Number 1 = [{123:10,.2f}]\nNumber 2 = [{99999.99:10,.2f}]\n")

a, b = 100, 999
print(f"1. Before {a} between {b} after. Variable")
next()
fstr= f"2. Before {a} between {b} after. In a string"
print(fstr)
next()
print(f"3. Before {100} between {999} after. Constant")
next()
print(f"4. Before {a**2} between {b-1} after.  Expression")
next()
```

**Program 22**:

| 08 | 46-evaluation.py |
|---|---|

```
 1 def next():
 2     input('\nNext> ')
 3
 4 a, b = 100, 999
 5 print(f"Case 1:\n\tReset values.  a = {a}, b = {b}.")
 6
 7 a = '???'
 8 print(f"Changing a to {a}")
 9 print(f"\ta = {a}, b = {b}.")
10 next()
11
12 # Case 2
13 a, b = 100, 999
14 str = f"a = {a}, b = {b}."
15 print(f"Case 2:\n\tReset values.  a = {a}, b = {b}.\n\tSaving the f-
16 string in a str = [{str}]")
17 a = '???'
18 print(f"Changing a to {a} and reprint the f-string, not the saved
19 str")
20 print(f"\ta = {a}, b = {b}. No difference from Case 1.")
21 next()
22
23 # Case 3
24 a, b = 100, 999
25 fstr = f"a = {a}, b = {b}."
26 print(f"Case 3:\n\tReset values.  a = {a}, b = {b}.\n\tSaving the f-
27 string in a str = [{str}]")
28 a = '???'
29 print(f"Changing a to {a} and print the saved f-string")
30 print(f"\t{fstr} Different from Case 1")
```