Strings in Python

Prepared by,

Dr. K. Vallidevi

Single or double quotes

- `'hello'` is the same as `"hello“`
- print("Hello")
  print('Hello’)
- Output:

Hello
Hello

- a = "Hello“   #Assigning to a variable
  print(a)
- Output:

Hello

Multiline Strings – single or double quotes

- a = ""“This is a demo to show,
  strings could spread to multiple lines.
  Three quotes in the way through which
  this could be achieved."""
- print(a)
- Output:
- This is a demo to show,
  strings could spread to multiple lines.
  Three quotes in the way through which
  this could be achieved.

Python - Slicing Strings

- Get the characters from position 2 to position 5 (not included):
-
```python
b = "Hello, World!"
print(b[2:5])
```
- Output:

```
llo
```

- Slice From the Start
-
```python
b = "Hello, World!"
print(b[:5])
```
- Output:

```
Hello
```

- Note: By leaving out the start index, the range will start at the first character

Python - Modify Strings

- Python has a set of built-in methods that can be used on strings.

-
  ```
  a = "Hello, World!"
  print(a.upper())
  ```
- Output: HELLO, WORLD!

-
  ```
  a = "Hello, World!"
  print(a.lower())
  ```
- Output: hello, world!

Remove Whitespace

- a = " Hello, World! "
- print(len(a))
- print(a.strip())
- print(len(a))
- b=a.strip()
- print(b)
- print(len(b))

```
15
Hello, World!
15
Hello, World!
13
```

String Concatenation

-
  ```
  a = "Hello"
  b = "World"
  c = a + b
  print(c)
  ```
- Output: `HelloWorld`

-
  ```
  a = "Hello"
  b = "World"
  c = a + " " + b
  print(c)
  ```
- Output: `Hello World`

Cannot Combine String and Int together with concatenation operator

-
  ```
  age = 36
  txt = "My name is John, I am " + age
  print(txt)  # Error
  ```
- Can combine strings and numbers by using the format() method

- The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are seen.

Use of format()

- Use the format() method to insert numbers into strings:

```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
```

- Output:

```
My name is John, and I am 36
```

Unlimited number of arguments for the format()

- The format() method takes unlimited number of arguments, and are placed into the respective placeholders:

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
✓print(myorder.format(quantity, itemno, price))
```

- Output:

```
I want 3 pieces of item 567 for 49.95 dollars.
```

Use of index numbers in format()

- You can use index numbers {0} to be sure the arguments are placed in the correct placeholders:

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

- Output:

```
I want to pay 49.95 dollars for 3 pieces of item 56
```

Need of Escape Character

- `txt = "We are the so-called "Vikings" from the north.“`
- Will get an error if double quotes is used inside a string that are surrounded by double quotes
- To insert characters that are illegal in a string, use an escape character.
- An escape character is a backslash \ followed by the character to be inserted.
- An example of an illegal character is a double quote inside a string that is surrounded by double quotes.
- txt = "We are the so-called \"Vikings\" from the north."

  print(txt)
- Output: We are the so-called "Vikings" from the north.

Escape Characters – Single Quotes, Backslash, newline

- Example 1
- txt = 'It\'s alright.’
  print(txt)
- Output: It's alright.
- Example 2
- txt = "This will insert one \\ (backslash)."
  print(txt)
- Output: This will insert one \ (backslash).
- Example 3:
- txt = "Hello\nWorld!"
  print(txt)
- Output:

```
Hello
World!
```

Escape Characters Continued

- Example 1: Tab character:
- txt = "Hello\tWorld!"
  print(txt)

Output: Hello   World!

- Example 2:

#This example erases one character (backspace):

txt = "Hello $\underline{\text{\textbackslash b}}$World!"

print(txt)

Escape Characters Continued

- ✓ #A backslash followed by three integers will result in a octal value:

```
txt = "\110\145\154\154\157"
print(txt)
```

Output: Hello

- ✓ Hexadecimal value:
- #A backslash followed by an 'x' and a hex number represents a hex value:

```
txt = "\x48\x65\x6c\x6c\x6f"
print(txt)
```

Output: Hello

String Methods

strings are immutable

- All string methods return new values. They do not change the original string.
- To uppercase the first letter in the sentence:
- txt = "hello, and welcome to my world."

  x = txt.capitalize()

  print (x)

Output: Hello, and welcome to my world.

string.capitalize()

-
  ```
  txt = "python is FUN!"
  x = txt.capitalize()
  print (x)
  ```
- `Output: Python is fun!`
- txt = "36 is my age."

x = txt.capitalize()

print (x)

36 is my age.

Use of count()

- txt = "I love apples, apple are my favorite fruit"

x = txt.count("apple")

print(x)

- Ouptut: 2
- *string*.count(*value, start, end*)
- txt = "I love apples, apple are my favorite fruit"

x = txt.count("apple", 10, 24)

print(x)

Output: 1

Change to title case

- txt = "Welcome to my world"

```
x = txt.title()
print(x)
Output: Welcome To My World

txt = "hello b2b2b2 and 3g3g3g"
x = txt.title()
print(x)
Output: Hello B2B2B2 And 3G3G3G
```

Use of translate()

- Example 1:

-
  ```
  #use a dictionary with ascii codes to replace 83 (S) with 80 (P):
  mydict = {83:  80}
  txt = "Hello Sam!"
  print(txt.translate(mydict))
  ```
- The translate() method returns a string where some specified characters are replaced with the character described in a dictionary, or in a mapping table.

- Example 2:
- txt = "Hi Sam!"

```
x = "mSa"
y = "eJo"
mytable = txt.maketrans(x, y)
print(txt.translate(mytable))
```

Output: Hi Joe!

Use of 3rd parameter in translate() – removes the characters

- txt = "Good night Sam!"

```
x = "mSa"
y = "eJo"
z = "odnght"
mytable = txt.maketrans(x, y, z)
print(txt.translate(mytable))
```

Output: G i Joe!

Use of dictionary in mapping the values for translate()

```
txt = "Good night Sam!"
mydict = {109: 101, 83: 74, 97: 111, 111: None, 100: None, 110: None,
103: None, 104: None, 116: None}
print(txt.translate(mydict))
```

Use of zfill()

```
txt = "50"
x = txt.zfill(5)
print(x)
Output: 00050
```

```
00000hello
welcome to the jungle
000010.000
```

```
a = "hello"
b = "welcome to the jungle"
c = "10.000"

print(a.zfill(10))
bb=len(b)
print(b.zfill(bb-6))
# print(b.zfill(bb+16))
print(c.zfill(10))
```

```
00000hello
0000000000000000welcome to the jungle
000010.000
```

Use of join()

- myDict = {"name": "John", "country": "Norway"}

```
mySeparator = "TEST"
x = mySeparator.join(myDict)
print(x)
```

Output: nameTESTcountry

Use of join()

```
myTuple = ("John", "Peter", "Vicky")
x = "#".join(myTuple)
print(x)
```

Output: John#Peter#Vicky

Use of isspace()

```
txt = "   s   "
x = txt.isspace()
print(x)
Output: False
```

1 Use of isprintable()

```
txt = "Hello! Are you #1?"
x = txt.isprintable()
print(x)
Output: True
```

Use of isdigit()

```
txt = "50800"
x = txt.isdigit()
print(x)
```

Use of index()

```
txt = "Hello, welcome to my world."
x = txt.index("welcome")
print(x)
```

Use of index()

```
string.index(value, start, end)
```

```
txt = "Hello, welcome to my world."
x = txt.index("e")
print(x)
```

Output: 1

Use of index()

```
txt = "Hello, welcome to my world."
x = txt.index("e", 5, 10)
print(x)
Output: 8

txt = "Hello, welcome to my world."
x = txt.index("welcome")
print(x)
Output: 7
```

Use of swapcase()

```
txt = "Hello My Name Is PETER"
x = txt.swapcase()
print(x)
```

```
hELLO mY nAME iS peter
```

Use of splitlines()

```
txt = "Thank you for the music \n Welcome to the jungle"
x = txt.splitlines()
print(x)
```

Output:

Use of split()

string to list

```
txt = "hello, my name is Peter hello, I am 26 years old"
x = txt.split()
print(x)
y=set(x)
print(y)
print(list(y))
Output:
['hello,', 'my', 'name', 'is', 'Peter', 'hello,', 'I', 'am', '26', 'years', 'old']
{'my', 'years', 'old', 'name', 'I', 'is', 'am', '26', 'Peter', 'hello,'}
['my', 'years', 'old', 'name', 'I', 'is', 'am', '26', 'Peter', 'hello,']
```

Use of rstrip()

```
txt = "banana,,!,,,ssqqqww....."
x = txt.rstrip(",.qsw!")
print(x)
```

```
Output: banana
```

Use of rsplit()

$\textcircled{D} \rightarrow$ diff ?

```
txt = "apple, banana, cherry, apple, cherry"
x = txt.rsplit(", ")
print(x)
Output: ['apple', 'banana', 'cherry', 'apple', 'cherry']
```

Use of rsplit()

```
txt = "apple, banana, cherry"

# setting the maxsplit parameter to 1, will return a list with 2 elements!

x = txt.rsplit(", ", 1)

print(x)

# note that the result has only 2 elements "apple, banana" is the first element, and "cherry" is the last.

['apple, banana', 'cherry']
```

Use of rpartition()

```
txt = "I could eat bananas all day, bananas are my favorite fruit"
x = txt.rpartition("bananas")
print(x)
```

Output:

```
('I could eat bananas all day, ', 'bananas', ' are my favorite fruit')
```

Use of rjust()

```
txt = "banana"
x = txt.rjust(20, "g")
print(x)
Output: ggggggggggggggbanana
```

✓ Use of ljust()

```
txt = "banana"
x = txt.ljust(20, "g")
print(x)
bananagggggggggggggg
```