Introduction To Python

Dr.Hajialiasgari

Tehran University
Of
Medical Science

November 3, 2024

TEHRAN UNIVERSITY
OF
MEDICAL SCIENCES

1 Strings

1 Strings

String Data Type

- A string is a sequence of characters.
- A string literal uses quotes: `'Hello'` or `"Hello"`.
- For strings, + means concatenate.
- When a string contains numbers, it is still a string.
- We can convert numbers in a string into a number using `int()`.

String Example Code

```
1  str1 = "Hello"
2  str2 = 'World!'
3  str3 = str1 + str2
4  print(str3)  # Output: HelloWorld!
5
6  str4 = '123'
7  str4 = str4 + 1  # TypeError: cannot concatenate 'str' and 'int' obj
8
9  x = int(str4) + 1
10 print(x)  # Output: 124
```

Looking Inside Strings

- Access any single character in a string using an index specified in square brackets.
- Index value must be an integer and starts at zero.
- Index value can be an expression that is computed.

Indexing Example

```
1 fruit = 'banana'
2 letter = fruit[1]
3 print(letter)  # Output: a
4
5 x = 3
6 w = fruit[x - 1]
7 print(w)  # Output: n
```

A Character Too Far

- Accessing beyond the end of a string results in an error.
- Be careful when constructing index values and slices.

IndexError Example

```
1 zot = 'abc'
2 print(zot[5])  # IndexError: string index out of range
```

Strings Have Length

- The built-in function `len` gives the length of a string.

Indexing Example

Index	0	1	2	3	4	5
Character	b	a	n	a	n	a

Table 1: Indexing of Each Character in "banana"

- Each character in the word "banana" has a unique index starting from 0.
- Accessing `fruit[1]` will return "a".

Length Example

```
1 fruit = 'banana'
2 print(len(fruit))  # Output: 6
```

Looping Through Strings

- Using a `while` statement, iteration variable, and `len` function to construct a loop.

Looping Example

```
1  fruit = 'banana'
2  for letter in fruit:
3      print(letter)
4
5  # Output
6  # 0 b
7  # 1 a
8  # 2 n
9  # 3 a
10 # 4 n
11 # 5 a
```

Looping and Counting

- Count the number of times a character appears in a string.

Counting Example

```
word = 'banana'
count = 0
for letter in word:
    if letter == 'a':
        count += 1
print(count)  # Output: 3
```

Slicing Strings

- Use a colon operator to access a continuous section of a string.
- The second number is up to but not including.

Slicing Example

```
s = 'Monty Python'
print(s[0:4])  # Output: Mont
print(s[6:7])  # Output: P
print(s[6:20]) # Output: Python
```

String Concatenation

- The + operator is used for concatenation.

Concatenation Example

```
1 a = 'Hello'
2 b = a + 'There'
3 print(b)  # Output: HelloThere
4
5 c = a + ' ' + 'There'
6 print(c)  # Output: Hello There
```

Using `in` as a Logical Operator

- The `in` keyword checks if one string is in another.
- Returns `True` or `False`.

in Example

```
fruit = 'banana'
print('n' in fruit)  # Output: True
print('m' in fruit)  # Output: False
print('nan' in fruit) # Output: True

if 'a' in fruit:
    print('Found it!')
```

String Library

- Python has many built-in string functions.
- Functions are invoked by appending them to the string variable.

String Functions Example

```
greet = 'Hello Bob'
zap = greet.lower()
print(zap)         # Output: hello bob
print(greet)       # Output: Hello Bob
print('Hi There'.lower())  # Output: hi there
```

Searching a String

- Use `find()` to search for a substring.
- Returns the index of the first occurrence, or -1 if not found.

Search Example

```
1 fruit = 'banana'
2 pos = fruit.find('na')
3 print(pos)  # Output: 2
4
5 aa = fruit.find('z')
6 print(aa)  # Output: -1
```

Search and Replace

- The `replace()` function replaces occurrences of a substring.

Replace Example

```
1 greet = 'Hello Bob'
2 nstr = greet.replace('Bob', 'Jane')
3 print(nstr)  # Output: Hello Jane
4
5 nstr = greet.replace('o', 'X')
6 print(nstr)  # Output: HellX BXb
```

Stripping Whitespace

- `lstrip()`, `rstrip()`, and `strip()` remove whitespace from strings.

Stripping Example

```
greet = '   Hello Bob   '
print(greet.lstrip())  # Output: 'Hello Bob   '
print(greet.rstrip())  # Output: '   Hello Bob'
print(greet.strip())   # Output: 'Hello Bob'
```

Prefixes

- Use `startswith()` to check if a string starts with a specific prefix.

Prefix Example

```
1 line = 'Please have a nice day'
2 print(line.startswith('Please'))  # Output: True
3 print(line.startswith('p'))       # Output: False
```

End of Strings