4. String

October 17, 2022

1 Introduction

- A string is a sequence of characters.
- A character is simply a symbol. For example, the English language has 26 characters.
- Computers do not deal with characters, they deal with numbers (binary). Even though you may see characters on your screen, internally it is stored and manipulated as a combination of 0s and 1s.
- This conversion of character to a number is called encoding, and the reverse process is decoding. ASCII and Unicode are some of the popular encodings used.
- In Python, a string is a sequence of Unicode characters. Unicode was introduced to include every character in all languages and bring uniformity in encoding.
- In Python strings can be created by enclosing characters inside a single quote or double-quotes. For Example: `'Hello'` or `"Hello"`
- When a string contains numbers, it is still a string
- We can convert numbers string into a number using `int()` or `float()`

[ ]:
```
str1 = "ACTS"
str2 = 'DBDA'
```

[ ]:
```
print(str1)
```

[ ]:
```
str1
```

[ ]:
```
str3 = "123"
```

[ ]:
```
type(str3)
```

[ ]:
```
a = int(str1)
```

[ ]:
```
print(a)
```

[ ]:
```
type(a)
```

- String literals can be defined with single quotes or double quotes.
- Can use other type of quotes inside the string.

[ ]:
```
bond = "I am 'Bond'...'James Bond'"
```

[ ]:
```
print(bond)
```

2 Multiline Stings

- Multiline strings can be initialized using ''' str''' or """ str """

[ ]:
```
str1 = ''' I am
"Bond"...
"James Bond"
   '''
```

[ ]:
```
print(str1)
```

3 Keyboard input

[ ]:
```
building_name = input("Enter name of your Apartment: ")
```

[ ]:
```
type(building_name)
```

[ ]:
```
house_no = input("Enter your House No: ")
```

[ ]:
```
type(house_no)
```

- Convert the data as required

[ ]:
```
house_no = input("Enter your House No: ")
house_no = int(house_no)
```

[ ]:
```
type(house_no)
```

[ ]:
```
house_no = int(input("Enter your House No: "))
```

[ ]:
```
type(house_no)
```

[ ]:
```
print(house_no)
```

4 Escape Characters

- New line: \n
- Tab: \t

[ ]:
```
print("Hello\nClass")
```

[ ]:
```
print("Hello\tClass")
```

5 Length of String

- `len()` function

[ ]:
```
str1 = "F.R.I.E.N.D.S"
```

[ ]:
```
len(str1)
```

6 Operations on Strings

1. Concatenation
2. Repetition
3. Access a char using index
4. Slicing
5. Membership
6. Raw string

6.1 Concatenation

- Joining 2 or more strings
- Strings can be joined using + operator

[ ]:
```
first_name = "Amitabh"
last_name = "Bachchan"
```

[ ]:
```
full_name = first_name + last_name
```

[ ]:
```
full_name
```

[ ]:
```
first_name + " " + last_name
```

[ ]:
```
"Amitabh" + " " + last_name
```

6.2 Repetition

- A string can be repeated multiple times * operator

[ ]:
```
"Knock" * 3
```

[ ]:
```
"Knock " * 3 + "Penny"
```

[ ]:
```
print("-"*28)
print("\tGood Morning")
print("-"*28)
```

6.3 Access a char using index

- We can access individual characters using indexing.

- Index starts from `0`(zero).
- Trying to access a character out of index range will raise an `IndexError`.
- The index must be an integer. We cannot use floats or other types, this will result into `TypeError`.
- Python allows negative indexing for its sequences for indexing from the end.
- The index of `-1` refers to the last item, `-2` to the second last item and so on.

[ ]:
```
b = "BIRTHDAY"
```

[ ]:
```
b[0]
```

[ ]:
```
b[4]
```

[ ]:
```
len(b)
```

[ ]:
```
b[8]
```

[ ]:
```
b[-1]
```

[ ]:
```
b[-8]
```

[ ]:
```
b[-9]
```

6.4 Slicing

- We can access range of characters using slicing.
- We can access a range of items in a string by using the slicing operator `:`(colon)
- Syntax: `string[ start : end+1 ]`

[ ]:
```
b
```

[ ]:
```
b[1:4]
```

[ ]:
```
b[:4]
```

[ ]:
```
b[3:]
```

[ ]:
```
b[-4:-1]
```

6.5 Membership

- Operators: `in`, `not in`

[ ]:
```
b
```

[ ]:
```
'h' in b
```

[ ]:
```
'H' not in b
```

[ ]:
```
'IRT' in b
```

[ ]:
```
'IRH' in b
```

6.6 Raw string

- Suppress the meaning of escape chars

[ ]:
```
hb = r"Happy\n\tBirthday"
```

[ ]:
```
print(hb)
```

7 Strings are Immutable

- Cannot modify a char in string
- This means that elements of a string cannot be changed once they have been assigned.
- We can simply reassign different strings to the same name.

[ ]:
```
b
```

[ ]:
```
b[1] = 'i'
```

[ ]:
```
b = 'BiRTHDAY'
```

[ ]:
```
b
```

8 `del`: To delete objects from kernel

- The `del` keyword is used to delete objects. In Python everything is an object, so the del keyword can also be used to delete any object like int, float, string, list dictionary, etc., or any user defined object.

[ ]:
```
b
```

[ ]:
```
del b
```

[ ]:
```
b
```

9 String Comparison

[ ]:
```
strc = "Python"
```

[ ]:
```
print(strc)
```

[ ]:
```
strc == "python" # Equal to
```

[ ]:
```
strc != "Python" # Not Equal to
```

[ ]:
```
strc > "python" # Greater than
```

[ ]:
```
strc < "python" # Less Than
```

10 Searching in a string

- Search for a substring within another string using `find()`
- Syntax: `S.find(sub[, start[, end]])`
- `find()` returnd the first occurrence of the substring
- If the substring is not found, it returns `-1`

[ ]:
```
b = "BIRTHDAY"
```

[ ]:
```
b.find('DAY')
```

[ ]:
```
b.find("Z")
```

[ ]:
```
b.find('RTH',4)
```

[ ]:
```
b.find('RTH',2,6)
```

[ ]:
```
b.find('DAY',2,8)
```

11 Find and Replace

- Search for a substring and replace with another substring using `replace()`
- Syntax: `S.replace(old, new[, count])`
- Replace all occurrences of the substring
- Return the string after the replacement

[ ]:
```
h = "Hello World"
```

[ ]:
```
h.replace('World','India')
```

[ ]:
```
h
```

[ ]:
```
h.replace('l','L',2)
```

[ ]:
```
h.replace('l','L')
```

[ ]:
```
h.replace('l','L',2)
```

12 Removing Whitespaces

- `strip()` - Remove whitespaces at the beginning and at the end
- `lstrip()` - Remove whitespaces at the beginning of left side
- `rstrip()` - Remove whitespaces at the end of Right side

[ ]:
```
j = "   Joey Doesn't Share Food!!!     "
```

[ ]:
```
j.strip()
```

[ ]:
```
j.lstrip()
```

[ ]:
```
j.rstrip()
```

[ ]:
```
j # Returns output string. That means original object is not modified.
```

13 String Formatting

- `format()` method returns a new string with its replacement fields in its string replaced with its arguments.
- Each replacement field is identified by a index number OR field name in braces.

[ ]:
```
print("Joey Doesn't Share Food!!!")
```

[ ]:
```
first_name = input("Name: ")
fav_item =input("Favourite Food Item: ")
```

[ ]:
```
print(first_name,"Doesn't Share",fav_item,"!!!")
```

[ ]:
```
print(first_name+" Doesn't Share "+str(fav_item)+"!!!")
```

[ ]:
```
print("{0} Doesn't Share {1}!!!".format(first_name,fav_item))
```

[ ]:
```
print("{name} Doesn't Share {favourite_item}!!!".format(name=first_name,␣
↪favourite_item=fav_item))
```

[ ]:
```
print("{name} Doesn't Share {favourite_item}!!!".
↪format(favourite_item=fav_item, name=first_name))
```

13.1 Format Specifications

[ ]:
```
s="I am learning Python"
```

[ ]:
```
len(s)
```

[ ]:
```
"{0:25}".format(s)  #Minimum Width 25
```

[ ]:
```
"{0:10}".format(s)  #Minimum Width 10
```

[ ]:
```
"{0:>25}".format(s) # right align, minimum width 25
```

[ ]:
```
"{0:^25}".format(s) # center align, minimum width 25
```

[ ]:
```
"{0:-^25}".format(s) # fill, center align, minimum width 25
```

[ ]:
```
"{0:.10}".format(s) # maximum width 10
```

14 f-Strings: A New and Improved Way to Format Strings

- Introduced in Python 3.6

[ ]:
```
name = "Akshay"
age = 26
print(f"im {name}, im {age}")
```

15 More string methods

- `capitalize()` - Converts the first character to upper case.
- `casefold()` - Converts string into lower case.
- `count()` - Returns the number of times a specified value occurs in a string.
- `endswith()` - Returns true if the string ends with the specified value.
- `index()` - Searches the string for a specified value and returns the position of where it was found.
- `isalnum()` - Returns True if all characters in the string are alphanumeric.
- `isalpha()` - Returns True if all characters in the string are in the alphabet.
- `isdecimal()` - Returns True if all characters in the string are decimals.
- `isdigit()` - Returns True if all characters in the string are digits.
- `isidentifier()` - Returns True if the string is an identifier.
- `islower()` - Returns True if all characters in the string are lower case.
- `isnumeric()` - Returns True if all characters in the string are numeric.
- `isprintable()` - Returns True if all characters in the string are printable.
- `isspace()` - Returns True if all characters in the string are whitespaces.
- `istitle()` - Returns True if the string follows the rules of a title.
- `isupper()` - Returns True if all characters in the string are upper case.
- `join()` - Joins the elements of an iterable to the end of the string.
- `lower()` - Converts a string into lower case.
- `split()` - Splits the string at the specified separator, and returns a list.
- `startswith()` - Returns True if the string starts with the specified value.
- `swapcase()` - Swaps cases, lower case becomes upper case and vice versa.
- `title()` - Converts the first character of each word to upper case.
- `upper()` - Converts a string into upper case.

[ ]:
```
d = "DOG"
```

[ ]:
```
d.upper()
```

[ ]:
```
d.isupper()
```

15.1 For Loop on String

[ ]:
```
string1 = "BIRTHDAY"
```

[ ]:
```
for i in string1:
    print(i)
```

16 More about `print` function

- praameters `end`, `sep`
  - `sep`: string inserted between values, default a space.
  - `end`: string appended after the last value, default a newline.

[ ]:
```
print("Hello", end=";")
print("World")
print("gm")
```

[ ]:
```
print("Hello","World", sep=",")
```

17 Help in Python

[ ]:
```
help(print)
```

- **Keyboard Shortcut in Jupyter for help:**
  - Keep cursor in middle or at at the end of function name whose help you are seeking and press `shift + tab`