Data Structures in Python

- **Some of the Data Structures(built-in container types) available in Python are:**

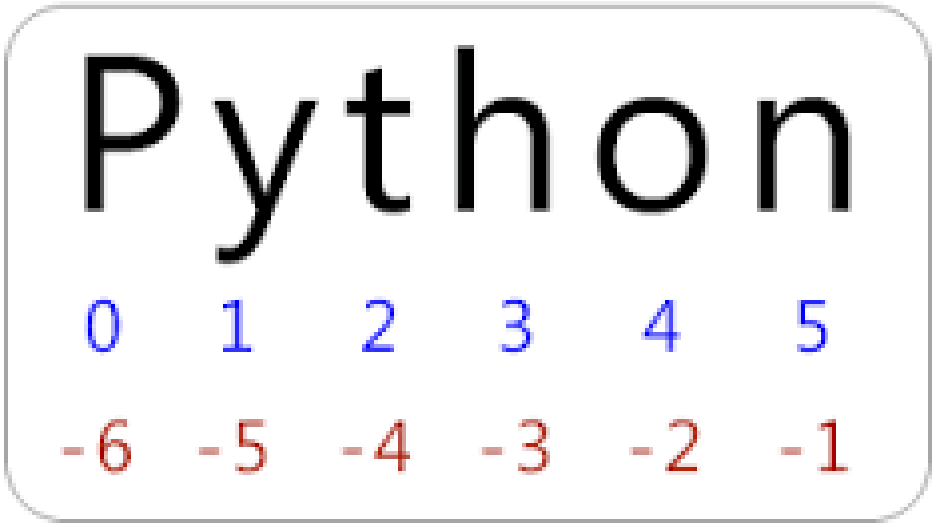
1. **Strings**
2. **List**
3. **Tuples**
4. **Sets**
5. **Dictionaries**

String

- Accepts 3 types of quotes to assign a string to a variable.
- single ('), double (") and triple (''' or """)
- Strings should start and end with same type of quotes
- Triple quotes are used to span string across multiple lines.
- Index starts from zero.

Accessing by negative index

- Can also be accessed using negative indices.
- Last character will start with -1 and traverses from right to left.



String Operators

Concatenation:

- Strings can be concatenated with '+' operator e.g. "Hello" + "World" will result in HelloWorld

Repetition:

- Repeated concatenation of string can be done using asterisk operator "*"
- **"Hello " * 3** will result in **Hello Hello Hello**

Indexing:

- str="Python"
- print(**str[0]**, **len(str)**,**str[len(str)-1]**)#prints **P** , **6** ,n
- "Python"[0] will also result in "P"

Strings: A sequence of characters

Python has great support for strings

- h = 'hello' *# String literals can use single quotes*
- w = "world" *# or double quotes;*
- *x=‘’’python’’’ #or triple quotes it does not matter.*
- **For the strings spanning to multiple lines we have to use triple quotes e.g.(Important)**
- Y= “”””Python is a Object Oriented Programming Language adopted by many companies also used for machine learning and analytics.”””””
- **print** (len(“hello”)) *# String length; prints "5“*

String Slicing

- **Substrings** are created using two indices in a square bracket separated by a **':'**
- **str[start:stop]** returns group of characters from **start** index to index **stop-1**
- **str[start:stop:step]** print in **increments** of index by **step** from **start** index to **stop-1** index

String

```
str = 'Hello World!'
print (str)          # Prints complete string
print (str[0])       # Prints first character of the string
print (str[2:5])     # Prints characters starting from
                       3rd(index 2) to 5th(index 4)
print (str[2: ])     # Prints string starting from 3rd
                       character
print(str[: : -1]    # reverses the string
print (str * 2)      # Prints string two times
print (str + "TEST“) # Prints concatenated string
```

- s="*harsh*"
- print(s[-3:]) # prints: rsh
- print(s[:-2]) # prints : har
- print(s.endswith(″sh")) # True

Built-in String Methods

- **str.capitalize()**
- It returns a copy of the string with only its **first character capitalized.**
- **str.center(12, 'a')**
- The method center() returns centered in a string of length width. **Padding is done using** the specified **fillchar.** Default filler is a space.
- **str.count(sub, start= 0,end=len(string))**
- The method **count()** returns the **number of occurrences of substring sub** in the range [start, end]. Optional arguments start and end are interpreted as in slice notation.

- **str.endswith(suffix[, start[, end]])**
- It **returns True** if the string **ends with the specified *suffix***, otherwise return False optionally restricting the matching with the given indices *start* and *end*.
- **str.find(str1, beg=0, end=len(string))**
- Determine if str1 occurs in string or in a substring of string if starting index ***beg*** and ending index ***end*** are given.
- **Returns index if found** and **-1 otherwise**.

- **str.index(str, beg=0 end=len(string))**
- It determines if string *str* occurs in string or in a substring of string if starting index *beg* and ending index *end* are given. **This method is same as find()**, **but raises an exception if sub is not found.**

isalnum()

- The method checks whether the string consists of **alphanumeric characters**.
- This method returns true if **all characters in the string are alphanumeric**, false otherwise.
- **str = "this2009“** # No space in this string print
  print(str.isalnum()) #Will print True
- **str = "this is string example....wow!!!”**
  print(str.isalnum()) #Will print False

strip() function

- str = "0000 !string example!!wow..!!!000"
- **print (str.strip('0'))** ***# Removes all leading and trailing whitespace of string***
- **print (str.lstrip('0'))** ***# Removes all leading hitespace of string***
- **print (str.rstrip('0'))** ***# Removes all trailing whitespace of string***
- **print (str.rstrip('0!.'))** ***# Removes trailing 0's,!'s and dots***
- st = " 07899*$Data088!@34507 "
- **print(st.strip())** **#remove whitespace**
- **print(st.lstrip())** **#remove whitespace from left**
- **print(st.rstrip())** **#remove whitespace from right**

split() function

- str="python, jython, ironPython, are the implementations"
- str.split(‘,’) *# returns a list of words separated by comma*
- Output : ['python', 'jython', 'ironPython', 'are the implementations']
- str.split() *# return a list of words separated by blank space*
- st.split(maxsplit=3) *#splits from left with 3 splits*
- st.rsplit(maxsplit=3) *#splits from right with 3 splits*

Caseless Matching

- **str.casefold()**
- Return a **casefolded copy** of the string.
- Casefolded strings may be **used for caseless matching**.
- **a**=*'Harsh Dev'*
- **e**=*'harsh dev‘*
- if(**e == a.casefold**()):
- print(*"string a and e are same")*

Using string module

- import string
- string.ascii_letter) *# all alphabets*
- string.ascii_lowercase *# all lowercase*
- string.ascii_uppercase *# all uppercase*
- string.punctuation *# all punctuation symbols*
- string.hexdigits *#0123456789abcdefABCDEF*
- string.octdigits *#01234567*
- string.capwords("harsh dev“)
- string.digits
- string.printable
- *'''if need to loop over string, e.g.’”*
- for i in string.ascii_lowercase:
- print(i)

Python Raw String

- Python raw string is created by prefixing a string literal with ‘r’ or ‘R’. Python raw string treats backslash (\) as a literal character. This is useful when we want to have a string that contains backslash and don’t want it to be treated as an escape character.
- s = 'Hi\nHello'
- print(s)
- Hi
- Hello
- Let’s see how raw string helps us in treating backslash as a normal character.
- raw_s = r'Hi\nHello'
- print(raw_s)
- Hi\nHello

Python Raw String and Quotes

- When a backslash is followed by a quote in a raw string, it’s escaped. However, the backslash also remains in the result. Because of this feature, we can’t create a raw string of single backslash. Also, a raw string can’t have an odd number of backslashes at the end.
- r'\' # missing end quote because the end quote is being escaped
- r'ab\\\' # first two backslashes will escape each other, the third one will try to escape the end quote.
- raw_s = r'\''
- print(raw_s)
- raw_s = r'ab\\'
- print(raw_s)
- raw_s = R'\\\"' # prefix can be 'R' or 'r'
- print(raw_s)