Upswing Academy

DATA TYPES IN PYTHON

Day - 2

Agenda

Table of Contents

- String
- List
- Tuple
- Set
- Dictionary
- Type Conversion

Data Types

Data types represents the kind of value that is assigned to a variable.

Type of the variable need not be declared and can be changed during the run time.

type(<var_name>) returns the data type

Built-In Datatypes

- Strings
  - name= "John"
- Integers
  - age = 25
- Float
  - salary = 20000.00
- Complex
  - complexVal = 1j
- Boolean
  - married = False
- List
  - hobby = ["Reading","Painting"]
- Tuple
  - hobby = ("Reading","Painting")
- Set
  - values = {1,2,3}
- Dictionary
  - details = {"name":"John","age":25}

Strings

Sequence of characters enclosed by single, double or triple quote is referred as “string”.

Python does not have a character data type, a single character is simply a string with a length of 1

Examples :

>>> Name = "John" # double quotes

>>> Name = 'John' #Single quotes

>>> Name = """John is working as an Architect""" # Multi-line string

LENGTH AND TYPE

```
>>> Profession = "Doctor"
>>> print(len(Profession)) # prints 6
>>> print(type(Profession)) # prints <str>
```

INDEXING

```
>>> Profession = "Doctor" # Index 0 to 5
>>> print(Profession[0]) # prints D
```

ITEM ASSIGNMENT NOT SUPPORTED

```
>>> Name = "fruits"
>>>
>>> Name[0] = "F"
Traceback (most recent call last):
  File "<pyshell#124>", line 1, in <module>
    Name[0] = "F"
TypeError: 'str' object does not support item assignment
```

LOOPING THROUGH A STRING

```
>>>for x in "fruits":
        print(x)
```

Strings

Format Strings

- Using Concatenation
- Passing as multiple objects in the print function
- Using the format function
- Directly using the variable in the format

```
>>> Name = "John Doe"
>>>
>>> print("Good Morning " + Name ) # Concatenation
Good Morning John Doe
>>> print("Good Morning ", Name ) # Passing as multiple objects
Good Morning  John Doe
>>> print("Good Morning {}".format(Name)) # using format
Good Morning John Doe
>>> print(f"Good Morning {Name}") # Using the variables directly
Good Morning John Doe
```

STRING OPERATORS

Basic Operators - Concatenation operator (+) ,
Replication Operator (*)

Membership Operators - in, not in

Comparsion Object - ==, <=, >=, !=

```
>>> text1 = "Good"
>>> text2 = "Morning"
>>> text1 + " " + text2
'Good Morning'
>>> text1 = "Good"
>>> text2 = "Morning"
>>> text1 + " " + text2 # Concatenation Operator ( + )
'Good Morning'
>>> text1 * 3 # Replication Operator (*)
'GoodGoodGood'
>>> "G" in text1
True
>>> "g" not in text1
True
>>> alpha = "a"
>>> beta = "b"
>>> alpha == beta
False
>>> alpha != beta
True
>>>
```

Strings

STRING SLICING

Technique of extracting substrings from a string is known as String Slicing.

1. Using the start and end index

```
>>> String = "Python Language"
>>> String[2:6] #prints characters from index 2 to 5
'thon'
>>> String[-7:-4] #prints characters from index -7 to -5
'ang'
```

2. Using either the start or the end index

```
>>> String = "Python Language"
>>> String[:6] #prints characters from index 0 to 5
'Python'
>>> String[3:] #prints characters from index 3 to last character
'hon Language'
>>> String[-3:] #prints characters from index -3 to end
'age'
>>> String[:-6] #prints characters from index 0 to end -7
'Python La'
```

3. Slicing using the stride value - which refers to how many characters to move forward after the first character is retrieved from the string.

```
>>> String = "Python Language"
>>> String[2:10:1]
'thon Lan'
>>> String[2:10:2]
'to a'
>>> String[2:10:4]
't '
>>> String[::-1] # Reverse
'egaugnaL nohtyP'
>>> String[::-3] # Reverse
'euant'
>>>
```

Strings

STRING SLICING

Technique of extracting substrings from a string is known as String Slicing.

1. Using the start and end index

```
>>> String = "Python Language"
>>> String[2:6] #prints characters from index 2 to 5
'thon'
>>> String[-7:-4] #prints characters from index -7 to -5
'ang'
```

2. Using either the start or the end index

```
>>> String = "Python Language"
>>> String[:6] #prints characters from index 0 to 5
'Python'
>>> String[3:] #prints characters from index 3 to last character
'hon Language'
>>> String[-3:] #prints characters from index -3 to end
'age'
>>> String[:-6] #prints characters from index 0 to end -7
'Python La'
```

3. Using neither start or end index

```
>>> String = "Python Language"
>>> String2 = String[:] # Returns entire string
>>> id(String)
2444855339056
>>> id(String2)
2444855339056
>>> String == String2
True
>>> String is String2
True
```

4. Slicing using the stride value - which refers to how many characters to move forward after the first character is retrieved from the string.

```
>>> String = "Python Language"
>>> String[2:10:1]
'thon Lan'
>>> String[2:10:2]
'to a'
>>> String[2:10:4]
't '
>>> String[::-1] # Reverse
'egaugnaL nohtyP'
>>> String[::-3] # Reverse
'euant'
>>>
```

String Methods

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

All string methods returns new values. They do not change the original string.

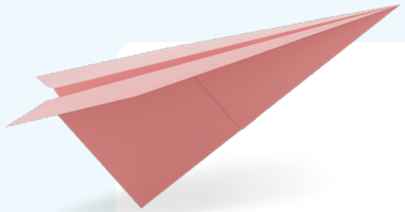
dir() of any string will return all possible string methods.

```
>>> String = "Python Language"
>>> dir(String)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '
__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__
hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__m
od__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod
__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize'
, 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'f
ormat_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifi
er', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'lju
st', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'replac
e', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines',
'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
>>>
```

Most Important String Methods

1. count() - Returns the number of times a specified value occurs in a string
2. endswith() - Returns true if the string ends with the specified value
3. find() - Searches the string for a specified value and returns the position of where it was found
4. index() - Searches the string for a specified value and returns the position of where it was found
5. lower() / upper() - Converts a string into lower/upper case
6. replace() - Returns a string where a specified value is replaced with a specified value
7. split() - Splits the string at the specified separator, and returns a list
8. strip() - Returns a trimmed version of the string
9. swapcase() - Swaps cases, lower case becomes upper case and vice versa
10. isalpha() - Returns True if all characters in the string are in the alphabet

Note : Refer the Python_Basic_Method document for all the methods with examples



Thank You

We hope you learned something new.