

PYTHON PROGRAMMING

Lecture 2

Unit 2

STRINGS

Name: Mr. Dheeraj Sundaragiri

Assistant Professor
Department of Computer Science and Engineering
SNIST

STRINGS

- Accessing Strings
- Basic Operations
- String slices
- Function and Methods

STRINGS INTRODUCTION

- A string is a sequence of characters.
- I Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double quotes.
- Creating strings is as simple as assigning a value to a variable.

STRINGS INTRODUCTION

☐ When a string contains numbers it is still a string.

```
For example:
```

```
var1 = 'Hello World!'
var2 = "Python Programming"
var3 = """Python
    Programming"""
```

ACCESSING STRINGS

- Dython does not support a character type; these are treated as strings of length one thus also considered a substring.
- ☐ To access substrings use the square brackets for slicing along with the index or indices to obtain your substring.

ACCESSING STRINGS

- Example:
 - o var1 = 'Hello World!'
 - var2 = "Python Programming"
 - o print "var1[0]: " var1[0]
 - o print "var2[1:5]: " var2[1:5]
- ☐ This will produce following result:
 - o var1[0]: H
 - var2[1:5]: ytho

STRING OPERATIONS

- concatenate with + or neighbors
 - \circ word = "Help" + "x"
 - word = "Help" "a"
- subscripting of strings
 - \circ Hello[2] 1
 - slice: Hello[1:2] el
 - o word[-1] last character
 - len(word) 5
- immutable: cannot assign to subscript

SLICING STRINGS

>>> str = 'Monty Python'



- The string position starts at zero
- We can also look at any continuous section of a string using a colon operator

>>> print str[0:4] # Mont

SLICING STRINGS

- The second number is one beyond the end of the slice - "up to but not including" >>> print str[6:7] # P
- If the second number is beyond the end of the string it stops at the end
 >> print str[6:20] # Python
- Slicing reverse>> print(str[-6:]) # Python

STRINGS: Functions and Methods

- Python has a number of string functions which are in the string library
- These functions are already built into every string we invoke them by appending the function to the string variable

var1 = 'Hello World!'
dir(var1)

STRINGS: Functions and Methods

• These functions do not modify the original string instead they return a new string that has been altered

- 1. capitalize 6. endswith 11. index
- 2. casefold 7. expandtabs 12. isalnum
- 3. center 8. find 13. isalpha
- 4. count 9. format 14. isdecimal
- 5. encode 10. format_map 15. isdigit

STRINGS: Functions and Methods

16. isidentifier 26. lstrip 36. split 17. islower 27. maketrans 37. splitlines 18. isnumeric 28. partition 38. startswith 19. isprintable 29. replace 39. strip 20. isspace 30. rfind 40. swapcase 31. rindex 21. istitle 41. title 22. isupper 32. rjust 42. translate 23. join 33. rpartition 43. upper 24. ljust 44. zfill 34. rsplit 12 25. lower 35. rstrip

STRINGS: Methods 1 of 20

capitalize casefold center count encode endswith expandtabs find format format_map 10. index isalnum isalpha isdecimal

isdigit

15.

>>> str="hello" >>> str.capitalize() # 'Hello' >>> str="hello PYTHON" >>> str.casefold()# 'hello python' >>> **str.center(24)** # width hello PYTHON >>> **str.center(24,'*')** # fillchar '****hello PYTHON***** 13

STRINGS: Methods 2 of 20

```
>>> str="hello PYTHON"
                 >>> str.count('o') #substring: 1
   <del>center</del>
   count
                 >>> str.count('O',0,5)
   encode
   endswith
                 # start, end: 0
   expandtabs
   find
                 >>> string = 'pythön!'
   format
   format_map
                # string.endswith(suffix[, start[, end]])
10.
   index
                 >>> string.endswith('!') #
   isalnum
   isalpha
                 >>> string.endswith('.') #
   isdecimal
   isdigit
```

True

False⁴

STRINGS: Methods 3 of 20

```
<del>count</del>
     encode
     <del>endswith</del>
     expandtabs
     find
     format
     format_map
10.
     index
     isalnum
     isalpha
     isdecimal
     isdigit
15.
```

```
>>> string = 'pythön!'
# encode to default Utf-8 Encoding
>>> string.encode() b'pyth\xc3\xb6n!'
# ignore error
>>> string.encode("ascii", "ignore")
   b'pythn!'
# replace error
>>> string.encode("ascii", "replace")
   b'pyth?n!'
```

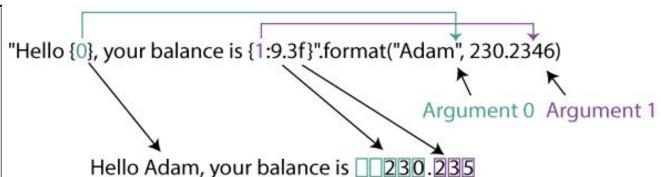
STRINGS: Methods 4 of 20

1. 9	capitalize	>>> str="Hi i am in\t#INDIA"
2. 2.	-center	>>> str.expandtabs() # tab size
4.	count	_
5.	encode	'Hi i am in #INDIA'
6.	endswith	>>> atmosphandtaba(2) # tab air
7.	expandtabs	>>> str.expandtabs(2) # tab siz
8.	find	'Hi i am in #INDIA'
9.	format	π
10.	format_map	>>> data= str.find('#')
11.	index	
12.	isalnum	>>> data # 11
13.	isalpha	
14.	isdecimal	>>> str[data+1:] # INDIA
15.	isdigit	

dtabs() # tab size 8 #INDIA' dtabs(2) # tab size 2 NDIA' find('#') 16 # INDIA

STRINGS: Methods 5 of 20

- capitalize center count expandtabs find format format_map
- 10.
- index
- isalnum
- isalpha
- isdecimal
- isdigit 15.



- >>> str="Hi {0}, Today is
- {1}".format("PYTHON","Good Day")
- >>> str
- 'Hi PYTHON, Today is Good Day'

STRINGS: Methods 6 of 20

```
<del>count</del>
     expandtabs
     find
     <del>format</del>
     format map
10.
     index
     isalnum
     isalpha
     isdecimal
     isdigit
```

```
>>> point = \{'x':4,'y':-5\}
>>> print('{x} {y}'.format(**point))
>>> print('{x} {y}'.format_map(point))
 # Both gives same output 4 -5
 # str.format(**mapping) copies the
 dict
 # str.format_map(mapping) makes
 a new dictionary during method
 call.
```

STRINGS: Methods 7 of 20

isdigit

```
# str.index(sub[, start[, end]] )
            >>> str = 'Hi i am in\t#INDIA'
<del>count</del>
            >>> str.index('in')
            >>> str.index('in',8)
                                           8
expandtabs
            >> str.index('IN',8,-3)
find
            >> str.index('i',2,30)
<del>format</del>
format_map
            >>> str[1:2].isalnum()
                                        # True
index
isalnum
            >>> str[1:2].isalpha()#
                                        True
isalpha
isdecimal
            >>> str.isalpha()
                                            False
```

STRINGS: Methods 8 of 20

```
>>> str = 'Hi i am in\t#INDIA'
    <del>isalnum</del>
                   >>> str.isdecimal()
    <del>isalpha</del>
    isdecimal
                   >>> str.isdigit()
15.
    isdigit
                   # The superscript and subscripts are
16.
    isidentifier
    islower
                   considered digit characters but not
18.
    isnumeric
    isprintable
19.
                   decimals.
    isspace
20.
                   >>> str[9:10].isidentifier() #
    istitle
    isupper
                   >>> str[1:-5].islower()  # True
23.
    join
    ljust
    lower
```

False

False

STRINGS: Methods 9 of 20

```
str = '1242323'
<del>index</del>
 <del>-isalnum</del>
                print(str.isnumeric())
                                                       # True
 <del>isalpha</del>
                \#str = '23455'
 <del>isdecimal</del>
 isdigit
                str = '\u00B23455'
                print(str.isnumeric())
 <del>-islower</del>
                                                       # True
 isnumeric
                \# \text{ str} = \frac{1}{2}
 isprintable
 isspace
                str = '\u00BD'
 istitle
                print(str.isnumeric())
                                                       # True
 isupper
 ioin
                s='python12'
 ljust
                print(s.isnumeric())
 lower
```

STRINGS: Methods 10 of 20

```
>>> str = ' '
19.
   isprintable
20.
   isspace
               >>> str.isprintable() # True
   istitle
               >>> str = '\n'
22.
   isupper
23.
   ioin
               >>> str.isprintable() # False
   ljust
25.
   lower
               >>> str = "
26.
   lstrip
               >>> str.isprintable())# True
   maketrans
28. partition
               replace
29.
               >>> str.isspace() # True
30.
   rfind
   rindex
31.
               >>> str = 'Monty @ Python'
32.
   rjust
               >>> str.istitle() # True
   rpartition
```

22

STRINGS: Methods 11 of 20

```
19. isprintable
                >>> str = 'Monty @ Python'
   isspace
                >>> str.isupper() # False
   <del>-istitle</del>
   isupper
                >>> lst=['1','2','3','4','5']
23.
    join
                >>> ch='-'
   ljust
25.
   lower
                >>> ch.join(lst) # '1-2-3-4-5'
   lstrip
26.
                >>> str.ljust(20,'*')
   maketrans
28.
   partition
                'Monty @ Python*****
29.
   replace
                >>> str.lower()
30.
   rfind
   rindex
                'monty @ python'
   rjust
   rpartition
```

STRINGS: Methods 12 of 20

```
26.
    lstrip
                 >>> str = 'Monty @ Python'
    maketrans
                 >>> str.lstrip()
28.
    partition
    replace
29.
                 'Monty @ Python '
30.
    rfind
31.
    rindex
                 >>> str.rstrip()
32.
    riust
                    Monty @ Python'
    rpartition
33.
34.
    rsplit
                 >>> str.strip()
35.
    rstrip
36.
    split
                 'Monty @ Python'
    splitlines
37.
                 >>> str.strip().strip('on')
    startswith
38.
39.
    strip
                 'Monty @ Pyth'
40.
    swapcase
```

STRINGS: Methods 13 of 20

```
27.
   maketrans
                 >>> str = 'Monty @ Python'
28.
   partition
29.
   replace
                 >>> str.rjust(20,'*')
30.
   rfind
   rindex
                 '******Monty @ Python'
32.
   rjust
33.
   rpartition
                 >>> str.replace("@","Hello")
34.
   rsplit
   rstrip
                 'Monty Hello Python'
36.
   split
   splitlines
                 >>> str.swapcase()
   startswith
<del>39.</del>
   strip
                 'mONTY @ pYTHON'
40.
   swapcase
41.
   title
                 >>> str.upper()
42.
   translate
43.
   upper
                 'MONTY @ PYTHON'
44.
   zfill
```

STRINGS: Methods 14 of 20

27. maketrans partition replace 30. rfind 31. rindex 32. rjust 33. rpartition 34. rsplit 35. rstrip 36. split splitlines startswith strip 40. swapcase 41 title 42. translate 43. upper zfill

maketrans() creates a mapping of the character's Unicode ordinal to its corresponding translation. >>> str1 = "aeiou"

>>> str2 = "12345" #must have same len

>>> str.maketrans(str1, str2) {97: 49, 101: 50, 105: 51, 111: 52, 117: 53}

STRINGS: Methods 15 of 20

```
27.
   maketrans
               >>> strt = str.maketrans(str1, str2)
28.
   partition
   replace
               >>> str = "this is string
   rfind
30.
                example....wow!!!"
   rindex
32. rjust
               >>> print (str.translate(strt))
33.
   rpartition
   rsplit
34.
                 th3s 3s str3ng 2x1mpl2...w4w!!!
   rstrip
36.
   split
               >>> str3 = "ae"
   splitlines
   startswith
                >>> strt = str.maketrans(str1, str2,
   <del>strip</del>
   swapcase
                str3)
41.
   title
               >>> print (str.translate(strt))
   translate
42.
                                                               27
   upper
                 th3s 3s str3ng xmpl....w4w!!!
   zfill
```

STRINGS: Methods 16 of 20

```
28.
    partition
               >>> str = 'Monty @ Python'
   <del>replace</del>
               >>> str.partition("@") # creates tuple
30.
    rfind
    rindex
               ('Monty', '@', 'Python')
32. rjust
33.
    rpartition
               >>> str.rpartition("on")
    rsplit
   rstrip
               # at last occurrence
36.
    split
    splitlines
               ('Monty @ Pyth', 'on', '')
    startswith
               >>> str.rfind('on')
    strip
    swapcase
                                                            28
               # returns highest index 12
    title
```

STRINGS: Methods 17 of 20

```
<del>partition</del>
                >>> str = 'Monty @ Python'
   <del>-replace</del>
                >>> str.rindex('n')
   <del>-rfind</del>
    rindex
                # returns highest index 13
32. rjust
   <del>-rpartition</del>
                # rfind() returns -1 if the substring is
    rsplit
   rstrip
                not found, whereas rindex() throws
36.
    split
    splitlines
                an exception.
    startswith
                >>> str='hello monty'
    strip
    swapcase
                >>> str.title() # 'Hello Monty'
    title
```

STRINGS: Methods 18 of 20

```
<del>partition</del>
                >>> str = 'Monty @ Python'
29. replace
                >>> str.split("@")
   <del>-rfind</del>
   <del>-rindex</del>
                # ['Monty', 'Python']
32. rjust
   <del>rpartition</del>
                >>> str.rsplit("@")
    rsplit
                # Same output, but Fractional Slow
   rstrip
    split
36.
                >>> str.split(" ",1)
    splitlines
    startswith
                # ['Monty', '@ Python']
    strip
                >>> str.rsplit(" ",1)
    swapcase
                                                                30
                # ['Monty @', 'Python']
```

STRINGS: Methods 19 of 20

```
<del>-rindex</del>
                 >>> str='Monty\n@\rPython'
    <del>-riust</del>
                 >>> str.splitlines()
   <del>rpartition</del>
   <del>-rsplit</del>
                 ['Monty', '@', 'Python']
35. rstrip
36. split
                 >>> str.splitlines(True)
    splitlines
    startswith
                 ['Monty\n', '@\r', 'Python']
<del>39. strip</del>
                 >>> str='Monty @ Python'
    swapcase
    <del>title</del>
                 >>> str.splitlines()
    upper
                 ['Monty @ Python']
    zfill
                 >>> str startswith('Mon')
```

31

STRINGS: Methods 20 of 20

```
<del>-rindex</del>
                >>> str='Monty\n@\rPython'
   <del>-riust</del>
                >>> str.startswith('Mon') # True
   <del>rpartition</del>
<del>34. rsplit</del>
                >>> str.startswith('Tue') # False
35. rstrip
   split
                >>> num="123"
37. splitlines
    startswith
                >>> num.zfill(8)
<del>39. strip</del>
                '00000123'
    swapcase
                >>> num.zfill(8)
                                          # num= "+123"
    upper
                '+0000123'
```

The topic of next lecture is

- Lists
- Tuples
- Dictionaries