8.Strings

December 15, 2018

1 Strings

Very important data type, especially for bioinformaticians * "The gray fox jump over the lazy dog" - text * "ATGTGTCGTGATGCGTTG" - DNA * "qwerty@server.domain" - service strings

1.1 String representation

Strings are made of characters and each character is reepresented as a numeric code internally. One of the simplest representation is ASCII

- `chr()` - get character represented by this code
- `ord()` - get numeric code of character

```
In [2]: chr(67)

Out[2]: 'C'

In [3]: ord('a')

Out[3]: 97
```

1.2 Creation

As you already know * `''`, `""`, triple variants - direct text * `str()` - constructor to make strings from other objects

```
In [2]: "My STRING"

Out[2]: 'My STRING'

In [1]: str([1, 2, 3])

Out[1]: '[1, 2, 3]'
```

1.3 Operations with strings

There are 2 allowed operations for strings * + - concatenate 2 strings * * - multiply string - concatenate string with itself n times

```
In [8]: 'Hello' + 'World'
```

```
Out[8]: 'HelloWorld'
```

```
In [12]: 'Hello' + ' ' + 'World'
```

```
Out[12]: 'Hello World'
```

```
In [13]: 'Hi!' * 3
```

```
Out[13]: 'Hi!Hi!Hi!'
```

```
In [74]: 3 * 'Hi!'
```

```
Out[74]: 'Hi!Hi!Hi!'
```

1.4 String methods

Strings have quite a big number of useful methods. Strings are immutable iterable object * General purpose methods * `index(substring, [begin, end])` - find 1st start of substring in string starting from begin to end; begin and end are 0 and end index of string by default * `count(substring, [begin, end])` - count non-overlapping occurences of substring in string starting from begin to end; begin and end are 0 and end index of string by default

```
In [67]: 'Hi everyone here!'.index('eve')
```

```
Out[67]: 3
```

```
In [72]: 'Hi everyone here!'.index('and me')
```

```
        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-72-9f2f9aac7565> in <module>()
    ----> 1 'Hi everyone here!'.index('and me')


        ValueError: substring not found
```

```
In [71]: 'Hi everyone here!'.count('er')
```

```
Out[71]: 2
```

```
In [73]: 'There is only light'.count('darkness')
```

```
Out[73]: 0
```

- String representation
  - `upper()` - make all characters UPPER CASE
  - `lower()` - MAKE ALL CHARACTERS lower case
  - `title()` - Make All Characters Title
  - `swapcase()` - mAKE aLL cHARACTERS tITLE
  - `capitalize()` - Make 1st character upper and other lowercase

```
In [12]: 'Make All Characters swapped'.capitalize()
```

```
Out[12]: 'Make all characters swapped'
```

```
In [6]: 'AWESOME natural NuMbEr - 2.71828'.title()
```

```
Out[6]: 'Awesome Natural Number - 2.71828'
```

```
In [7]: 'atgtcgtgtcgtgtcgtaatgagtctatatatatat'.upper()
```

```
Out[7]: 'ATGTCGTGTCGTGTCGTAATGAGTCTATATATATAT'
```

```
In [8]: 'E.Mail@gmail.com'.lower()
```

```
Out[8]: 'e.mail@gmail.com'
```

```
In [10]: 'E.Mail@gmail.com'.swapcase()
```

```
Out[10]: 'e.mAIL@GMAIL.COM'
```

- Determine type of character
  - `isalpha()` - whether string contains only letters
  - `isdigit()` - whether string contains only digits
  - `isalnum()` - whether string contains only digits and letters
  - `isupper()` - whether letters in string only upper
  - `islower()` - whether letters in string only lower
  - `isspace()` - whether string contains only whitespace characters
  - `startswith(substring)` - whether substring is a start of string
  - `endswith(substring)` - whether substring is an end of string

```
In [16]: 'abc'.isalpha()
```

```
Out[16]: True
```

```
In [17]: ''.isalpha()
```

```
Out[17]: False
```

```
In [18]: '12'.isdigit()
```

```
Out[18]: True

In [19]: '1'.isdigit()

Out[19]: True

In [33]: 'Aa'.isupper()

Out[33]: False

In [34]: 'AAA'.isupper()

Out[34]: True

In [35]: '123A'.isupper()

Out[35]: True

In [32]: ' \t \n'.isspace()

Out[32]: True

In [78]: 'And I\'m in combat!'.startswith('A')

Out[78]: True

In [79]: 'Cause every hour in my head'.startswith('Cau')

Out[79]: True

In [80]: 'Sigh'.startswith('s')

Out[80]: False

In [80]: 'Is it true'.endswith('e')

Out[80]: False
```

- String transformation
  - `replace(old, new, n)` - replace each old substring in string with new one n times; replace every substring by default, non-overlapping
  - `join(iterable)` - create string from elements in iterable and interleave theem with string; elements in iterable should be str for this method
  - `maketrans(original, new)` and `translate(table)` - methods to translate characters

```
In [76]: 'reverse transcription'.replace('e', 'i')

Out[76]: 'rivirsi transcription'

In [47]: # Non overlapping
         'ATATATGTCG'.replace('ATA', 'TUT')
```

```
Out[47]: 'TUTTATGTCG'
```

```
In [75]: 'The gray fox jump over the lazy dog'.replace('the', 'not')
```

```
Out[75]: 'The gray fox jump over not lazy dog'
```

```
In [48]: ', '.join(('a', 'b', 'c', 'd'))
```

```
Out[48]: 'a, b, c, d'
```

```
In [51]: '*'.join({1, 2, 'c', True, '4.65'})
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-51-b905a80d4ae0> in <module>()
    ----> 1 '*'.join({1, 2, 'c', True, '4.65'})


        TypeError: sequence item 1: expected str instance, int found
```

```
In [2]: '*'.join({str(1), str(2), 'c', str(True), '4.65'})
```

```
Out[2]: '1*c*True*2*4.65'
```

```
In [24]: # Reverse TRANSCRIPTION
         'AUGUGCGUGA'.translate(str.maketrans('AUGC', 'TACG'))
```

```
Out[24]: 'TACACGCACT'
```

- Useful methods for string processing
  - `strip()` - get rid of leading nd trailing spaces
  - `split(separator, n)` - convert string to a list of its parts - split it by separator (whitespaces by default); n is equal to number of separator in string by default
  - `splitlines()` - nice method to split text by lines

```
In [12]: '   Inconsistency in spaces is part of originality. \n'.strip()
```

```
Out[12]: 'Inconsistency in spaces is part of originality.'
```

```
In [9]: 'there is no hope'.strip('therp ')
```

```
Out[9]: 'is no ho'
```

```
In [29]: 'There is no faith'.split()
```

```
Out[29]: ['There', 'is', 'no', 'faith']
```

```
In [30]: 'There \nis no  \t faith'.split()

Out[30]: ['There', 'is', 'no', 'faith']

In [41]: print('I am\na\nfucking\r\ntext\n')

I am
a
fucking
text



In [43]: # By every whitespace character
         'I am\na\nfucking\r\ntext\n'.split()

Out[43]: ['I', 'am', 'a', 'fucking', 'text']

In [46]: # By UNIX newline character
         'I am\na\nfucking\r\ntext\n'.split('\n')

Out[46]: ['I am', 'a', 'fucking\r', 'text', '']

In [47]: # By every newline character
         'I am\na\nfucking\r\ntext\n'.splitlines()

Out[47]: ['I am', 'a', 'fucking', 'text']
```