Programming for Data Science (CSE3041)

Ramesh Ragala

VIT Chennai Campus

July 26, 2020

Problem - Check Validity of a PAN

In any of the country's official documents, the PAN number is formatted as follows:
<alphabet> <alphabet> <alphabet> <alphabet>
<alphabet> <digit> <digit> <digit> <digit> <alphabet>
Your task is to figure out if the PAN number is valid or not. A valid PAN number will have all its letters in uppercase and digits in the same order as listed above.

Strings in Python

Test Case -1

ABCDE1234R

Test Case -1

```
ABCDE1234R
Valid
```

Strings in Python

Test Case -1

ABCDE1234R
Valid

Test Case -2

ABCDE12345

Test Case -1

ABCDE1234R
Valid

Test Case -2

ABCDE12345
Invalid → Last Character should be character

Test Case -1

ABCDE1234R
Valid

Test Case -2

ABCDE12345
Invalid → Last Character should be character

Test Case -3

abcd01234r

Strings in Python

Test Case -1

ABCDE1234R
Valid

Test Case -2

ABCDE12345
Invalid → Last Character should be character

Test Case -3

abcd01234r
Invalid → All characters should be in upper case

Strings in Python

Strings

- **Immutable** sequence of characters
- A string literal uses quotes
- ’Hello’ or ”Hello” or ”’Hello”’
- For strings, + means ”concatenate”
- When a string contains numbers, it is still a string
- We can convert a string into a number using int()→ typecasting

Strings in Python

Operation	Interpretation
S = ”	Empty String
S = ”VIT’s”	Double Quotes, same as Single
S = ’s\np\ta\x00m’	Escape Sequence
S = ””” ... multiline..”””	Triple - quoted block strings
S = r’\temp\spam’	Raw Strings(no escapes)
S = b’sp\xc4m’	Byte Strings in 2.6, 2.7 and 3.X
S = u’sp\u00c4m’	Unicode Strings in 2.X and 3.3+
S1 + S2	Concatenate
S * 3	Repeat
S[i]	Index
S[i:j]	Slice
len(S)	length of the string

Strings in Python

Operation	Interpretation
"a %s parrot" % kind	String formatting Expression
"a {0} parrot".format(kind)	String formatting method in 2.6, 2.7 and 3.X
S.find('pa')	String methods, search
S.rstrip()	remove whitespace
S.replace('pa','xx')	replacement
S.split(',')	split on delimiter
S.isdigit()	Content Test
S.lower()	Case Conversion
S.endswith('spam')	End Test
'spam'.join(strlist)	Delimiter Join

Strings in Python

Example Strings

- Single quotes: ’spa”m’
- Double quotes: ”spa’m”
- Triple quotes: ’ ’ ’... spam ...’ ’ ’, ”””... spam ...”””
- Escape sequences: ”s\tp\na\0m ”
- Raw strings: r”C:\new\test.spm ”

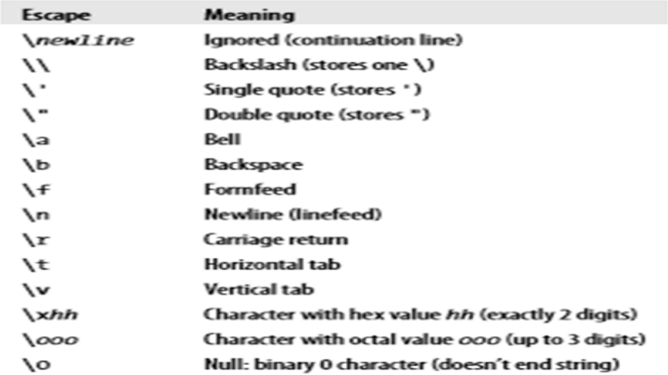
Escape Sequences

- Represent Special Characters

```
>>> s = 'a\nb\tc'
>>> s
'a\nb\tc'
>>> print(s)
a
b    c
>>> len(s)
5
```

Strings in Python

Escape Sequences



Escape	Meaning
\newline	Ignored (continuation line)
\\\\	Backslash (stores one \\)
\\'	Single quote (stores ')
\\"	Double quote (stores ")
\\a	Bell
\\b	Backspace
\\f	Formfeed
\\n	Newline (linefeed)
\\r	Carriage return
\\t	Horizontal tab
\\v	Vertical tab
\\x*hh*	Character with hex value *hh* (exactly 2 digits)
\ooo	Character with octal value *ooo* (up to 3 digits)
\\0	Null: binary 0 character (doesn't end string)

Strings in Python

Length of a String

```
>>> s = 'a\0b\0c'
>>> s
'a\x00b\x00c'
>>> len(s)
5
>>> print(s)
a b c
```

Strings in Python

Length of a String

- a binary 1 and 2 (coded in octal), followed by a binary 3 (coded in hexadecimal):

```
>>> s = '\001\002\x03’
>>> s
'\x01\x02\x03’
>>> len(s)
3
```

Strings in Python

Backslash in Strings

- if Python does not recognize the character after a \as being a valid escape code, it simply keeps the backslash in the resulting string:
- >>> x = ”C:\py\code”
- # Keeps \literally (and displays it as \\)
- >>> x
- ’C:\\py\\code’
- >>> len(x)
- 10

Strings in Python

Check this

```
>>> s = ”C:\new\text.dat”
>>>s
>>>print(s)
>>> s1 = r”C:\new\text.dat”
>>>s1
>>>print(s1)
>>> s2 = ”C:\\new\\text.dat”
>>>print(s2)
>>>s2
```

Opening a File

- myfile = open('C:\new\text.dat', 'w') - Error
- myfile = open(r'C:\new\text.dat', 'w')
- Alternatively two backslashes may be used
- myfile = open('C:\\new\\text.dat', 'w')
- >>> path = r'C:\new\text.dat'
- >>> print(path)   # User-friendly format C:\new\text.dat
- >>> len(path)
- 15

Strings in Python

Basic Operations

```
>>> 'Ni!’ * 4
'Ni!Ni!Ni!Ni!’
>>> print(’-’ * 80)      # 80 dashes, the easy way
>>> myjob = ”hacker”
>>> for c in myjob:
print(c, end=’ ’)
h a c k e r
```

Strings in Python

Using ’in’ Operator in Strings

```
>>> ”k” in myjob        # Found
True
>>> ”z” in myjob        # Not found
False
>>> ’spam’ in ’abcspamdef’
# Substring search, no position returned
True
```

Counting

Count the number of ’a’

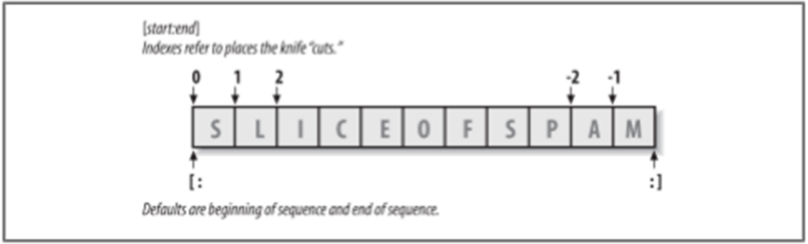
Example:

```
word = ’Btechallbranches’
count = 0
for letter in word :
if letter == ’a’ :
count = count + 1
print count
```

Strings in Python

Indexing and Slicing

- >>> S = 'spam'
- Last character in the string has index -1 and the one before it has index -2 and so on



[start:end]

Indexes refer to places the knife "cuts."

Defaults are beginning of sequence and end of sequence.

Indexing and Slicing

- Take one letter from a word at a time
- Use square bracket and give the index of the letter to be extracted
- Indexing can be done either from front or from end
- >>> S[0], S[-2]
- ('s', 'a')

Slicing

- Take a part of a word
- Square bracket with two arguments with a colon
- First value indicates the starting position of the slice and second value indicates the stop position of the slice
- Character at the stop position is not included in the slice
- >>> S[1:3]
- 'pa'

Strings in Python

Slicing

- If the second number is beyond the end of the string, it stops at the end
- If we leave off the first or last number of the slice, it is assumed to be beginning or end of the string respectively
- s = ‘spam’
- >>> s[:3]
- ’spa’
- >>> s[1:]
- ’pam’

Properties of Slicing

- S[1:3] fetches items at offsets 1 up to but not including 3.
- S[1:] - fetches items at offset 1 through the end
- S[:3] - fetches items at offset 0 up to but not including 3
- S[:-1] - fetches items at offset 0 up to but not including last item
- S[:] - fetches items at offsets 0 through the end—making a top-level copy of S

Extended slicing

- X[I:J:K] - means "extract all the items in X, from offset I through J-1, by K."
- Third limit, K, defaults to +1
- If you specify an explicit value it is used to skip items
- Extraction is reversed when negative value is given for K-1
- Each time K-1 items are skipped

Strings in Python

Extended slicing Example

```
>>> S = ’abcdefghijklmnop’
>>> S[1:10:2]        # Skipping items
’bdfhj’
>>> S[::2]
’acegikmo‘
>>> S = ’hello’
>>> S[::-1]        # Reversing items
’olleh’
```

Strings in Python

String Conversion Tools

```
>>> ”42” + 1
TypeError: Can’t convert ’int’ object to str implicitly
>>> int(”42”), str(42)        # Convert from/to string
(42, ’42’)
int(”42”) + 1
43
>>> ”42” + str(1)
’421’
```

Character code Conversions

- ord () - Convert a single character to its underlying integer code (e.g., its ASCII byte value) - this value is used to represent the corresponding character in memory.
- >>> ord('s')
- 115
- chr () – Does inverse of ord
- >>> chr(115)
- 's'

Strings in Python

Character code Conversions - Example

```
>>> S = ’5’
>>> S = chr(ord(S) + 1)
>>> S
’6’
>>> S = chr(ord(S) + 1)
>>> S
’7’
>>> ord(’5’) - ord(’0’)
5
>>> int(’1101’, 2)        # Convert binary to integer
13
>>> bin(13)       # Convert integer to binary
’0b1101’
```

Strings in Python

Concatenation

```
>>> S1 = ’Welcome’
>>> S2 = ’Python’
>>> S3 = S1 + S2
>>> S3
’WelcomePython’
```

Changing Strings

- String - "immutable sequence"
- Immutable - you cannot change a string in place
- >>> S = 'spam'
- >>> S[0] = 'x'   # Raises an error!
- TypeError: 'str' object does not support item assignment
- But S = 'Apple' works
- How??
- >>> S = S + 'SPAM!'   # To change a string, make a new one
- >>> S
- 'spamSPAM!'
- >>> S = S[:4] + 'Burger' + S[-1]
- >>> S
- 'spamBurger!'

Strings in Python

Replace

- >>> S = 'splot'
- >>> S = S.replace('pl', 'pamal')
- >>> S
- 'spamalot'

Formatting Strings

- >>> 'That is %d %s bird!' % (1, 'dead')
- That is 1 dead bird!
- >>> 'That is {0} {1} bird!'.format (1, 'dead')
- 'That is 1 dead bird!'

Strings in Python

String Library

- Python has a number of string functions which are in the string library
- These functions do not modify the original string, instead they return a new string that has been altered

Example:

```
>>> greet = 'Hello Arun'
>>> zap = greet.lower()
>>> print (zap)
hello arun
>>> print ('Hi There'.lower())
hi there
```

Searching a String

- find() - function to search for a string within another
- find() - finds the first occurrence of the substring
- If the substring is not found, find() returns -1

Example:

```
>>> name = 'pradeepkumar'
>>> pos = name.find('de')
>>> print pos
3
>>> aa = "fruit".find('z')
>>> print (aa)
-1
>>> name = 'pradeepkumar'
>>> pos = name.find('de',5,8)
>>> pos
-1
```

Other Common String Methods in Action

```
>>> line = ”The knights who say Ni!\n”
>>> line.rstrip()
’The knights who say Ni!’
>>> line.upper()
’THE KNIGHTS WHO SAY NI!\n’
>>> line.isalpha()
False
>>> line.endswith(’Ni!  \n’)
True
>>> line.startswith(’The’)
True
```

Other Common String Methods in Action

- Length and slicing operations can be used to mimic endswith:

```
>>> line = 'The knights who say Ni!\n'
>>> line.find('Ni') != -1
True
>>> 'Ni' in line
True
>>> sub = 'Ni! \n'
>>> line.endswith(sub)        # End test via method call or slice
True
>>> line[-len(sub):] == sub
True
```

Thank you