Strings in Python

Check Validity of a PAN

In any of the country's official documents, the PAN number is listed 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.

PAC For PAN Problem

Input	Processing	Output
PAN number	Take each character and check if alphabets and digits are appropriately placed	Print Valid or Invalid

Pseudocode

```
READ PAN
If length of PAN is not ten then print “Invalid” and exit
FOR x=0 to5
        if PAN[x] is not a upper case character THEN
                PRINT ‘invalid’
                BREAK;
        END IF
END FOR
FOR x=5 to 9
        if PAN[x] is not a digit THEN
                PRINT ‘invalid’
                BREAK;
        END IF
END FOR
IF PAN[9] is not a upper case character THEN
        PRINT ‘invalid’
                END IF
PRINT ‘valid’
```

Test Case 1

ABCDE1234R

Valid

Test Case 2

abcde12345

Invalid

Test Case 3

abcd01234r

Invalid

Strings

- Immutable (unchangeable) 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 numbers in a string into a number using int()

String Operations

Operation	Interpretation
`S = ''`	Empty string
`S = "spam's"`	Double quotes, same as single
`S = 's\np\ta\x00m'`	Escape sequences
`S = """...multiline..."""`	Triple-quoted block strings
`S = r'\temp\spam'`	Raw strings (no escapes)
`B = b'sp\xc4m'`	Byte strings
`U = u'sp\u00c4m'`	Unicode strings
`S1 + S2`	Concatenate, repeat
`S * 3`	
`S[i]`	Index, slice, length
`S[i:j]`	

String Operations

`len(S)`	
`"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 (see ahead for all 43): search,
`S.rstrip()`	remove whitespace,
`S.replace('pa', 'xx')`	replacement,
`S.split(',')`	split on delimiter,

String Operations

Operation	Interpretation
S.isdigit()	content test,
S.lower()	case conversion,
S.endswith('spam')	end test,
'spam'.join(strlist)	delimiter join,

Example Strings

- Single quotes: 'spa"m'

```
>>>S= 'spa"m'
>>>print(S)
spa"m
```

- Double quotes: "spa'm"

```
>>> S="spa'm"
>>> print(S)
spa'm
```

Triple quotes: '''... spam ...''', """... spam ..."""

Example Strings

- Triple quotes:
  - You can assign a multiline string to a variable by using three quotes:

```
>>> a = """Zorem ipsum dolor sit amet,
consectetur adipiscing elit,
ut labore et dolore magna aliqua."""
>>> print(a)
Zorem ipsum dolor sit amet,
consectetur adipiscing elit,
ut labore et dolore magna aliqua.
```

Example Strings

- Triple quotes: '''... spam ...''', """... spam ..."""

```
>>> S='''... spam ...'''
>>> print(S)
... spam ...
```

- Escape sequences: "s\tp\na\0m"

```
>>> S="s\tp\na\0m"
>>> print(S)
s	p
a m
```

“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)

Escape Sequences

Represent Special Characters

```
>>> s = R'a\nb\tc’s'

>>> print(s)

a\nb\tc

>>> len(s)

5
```

Escape Sequences

```
>>>txt = 'It\'s alright.'
>>>print(txt)
It's alright.

>>>txt = "This will insert one \\ (backslash)."
>>>print(txt)
This will insert one \ (backslash).
```

Escape Sequences

#A backslash followed by three integers will result in a octal value:

```
>>>txt = "\110\145\154\154\157"
>>>print(txt)
Hello
```

#A backslash followed by an 'x' and a hex number represents a hex value:

```
>>>txt = "\x48\x65\x6c\x6c\x6f"
>>>print(txt)
Hello
```

Raw strings

Python raw string is created by prefixing a string literal with ‘r’ or ‘R’.

Python raw string treats backslash (\) as a literal character.

```
>>> R=r"C:\new\test.spm"
>>> print(R)
C:\new\test.spm
```

Length of a String

```
>>> s = ‘abc'
>>> s
‘abc’
>>> len(s)
3
>>> print(s)
abc
```

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
```

Check this

```
>>> s = "C:\new\text.dat"
>>> print(s)
C:
ew      ext.dat
>>> s1 = r"C:\new\text.dat"
>>> print(s1)
C:\new\text.dat
>>>s2 = "C:\\new\\text.dat“
>>>print(s2)
C:\new\text.dat
```

Basic Operations

```
>>> 'Ni!' * 4

'Ni!Ni!Ni!Ni!‘

>>> print('-' * 10)                    # 10 dashes, the easy way

----------

>>>x = "Python is "

>>>y = "awesome"

>>>z =  x + y

print(z)

Python is awesome
```

Basic Operations

If you try to combine a string and a number, Python will give you an error:

```
>>>x = 5
>>>y = “6"
>>>print(x + int(y))
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Using ‘in’ Operator in Strings

```
>>> myjob = "hacker"
>>> "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
```

Counting

```
>>>txt = "I love apples, apple are my favorite fruit"
>>>x = txt.count("apple")
>>>print(x)
2
```

string.count(value, start, end)

value - Required. A String. The string to value to search for

start - Optional. An Integer. The position to start the search. Default is 0

end - Optional. An Integer. The position to end the search. Default is the end of the string

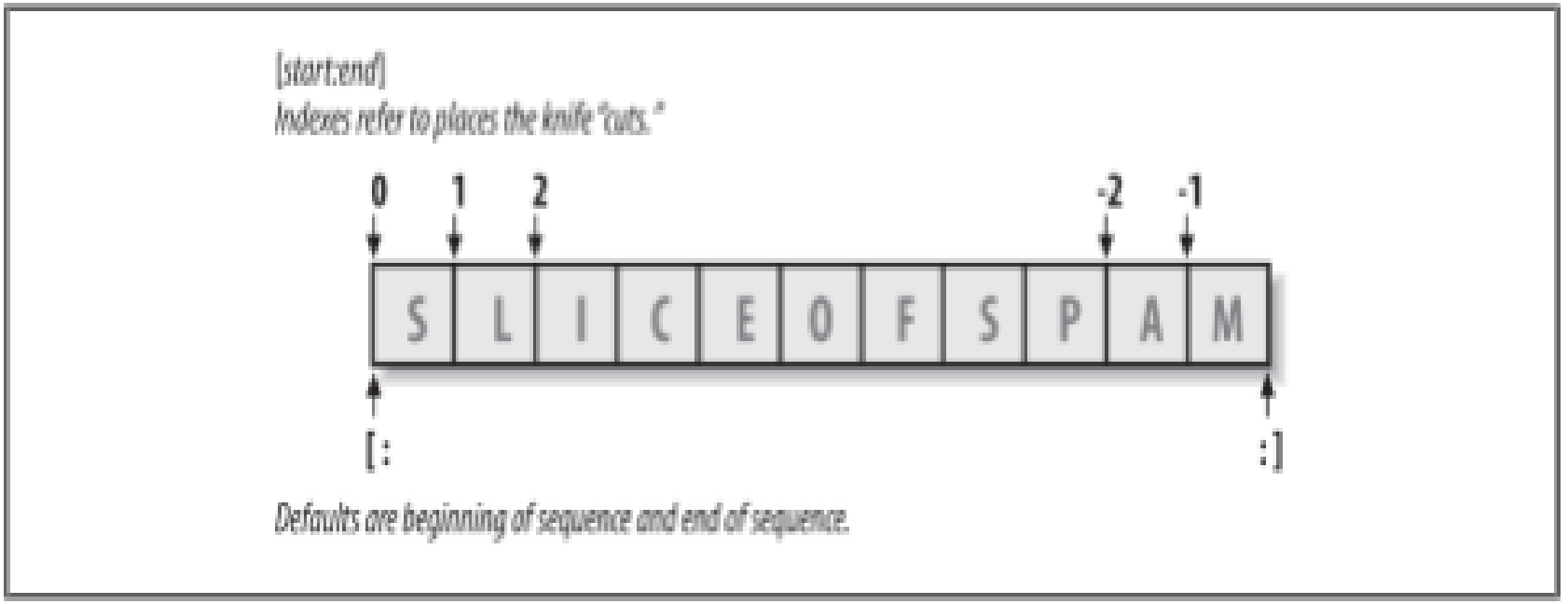
Counting

```
>>>txt = "I love apples, apple are my favorite fruit"

>>>x = txt.count("apple", 10, 24)

>>>print(x)

1
```

Indexing and Slicing

```
>>> S = 'spam'
```

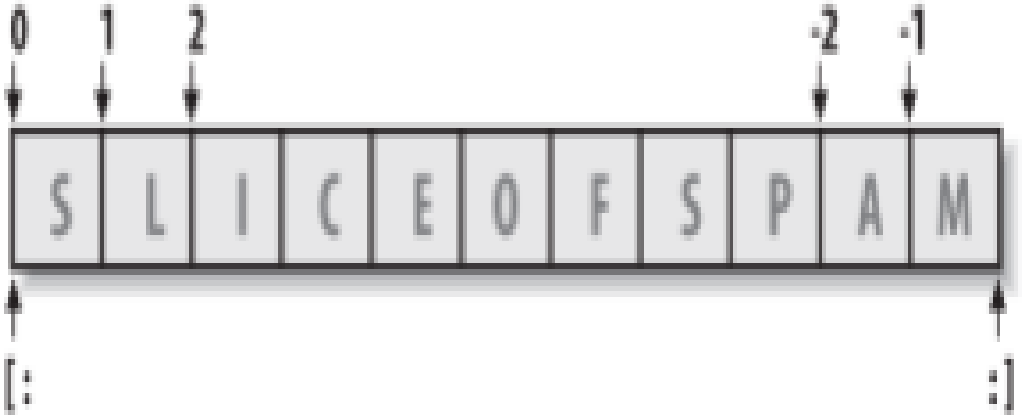
Last character in the string has index -1 and the one before it has index -2 and so on



Indexing and Slicing

[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='spam'
>>> 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 = ‘spam’

>>> **S[1:3]**

'pa’

Slicing

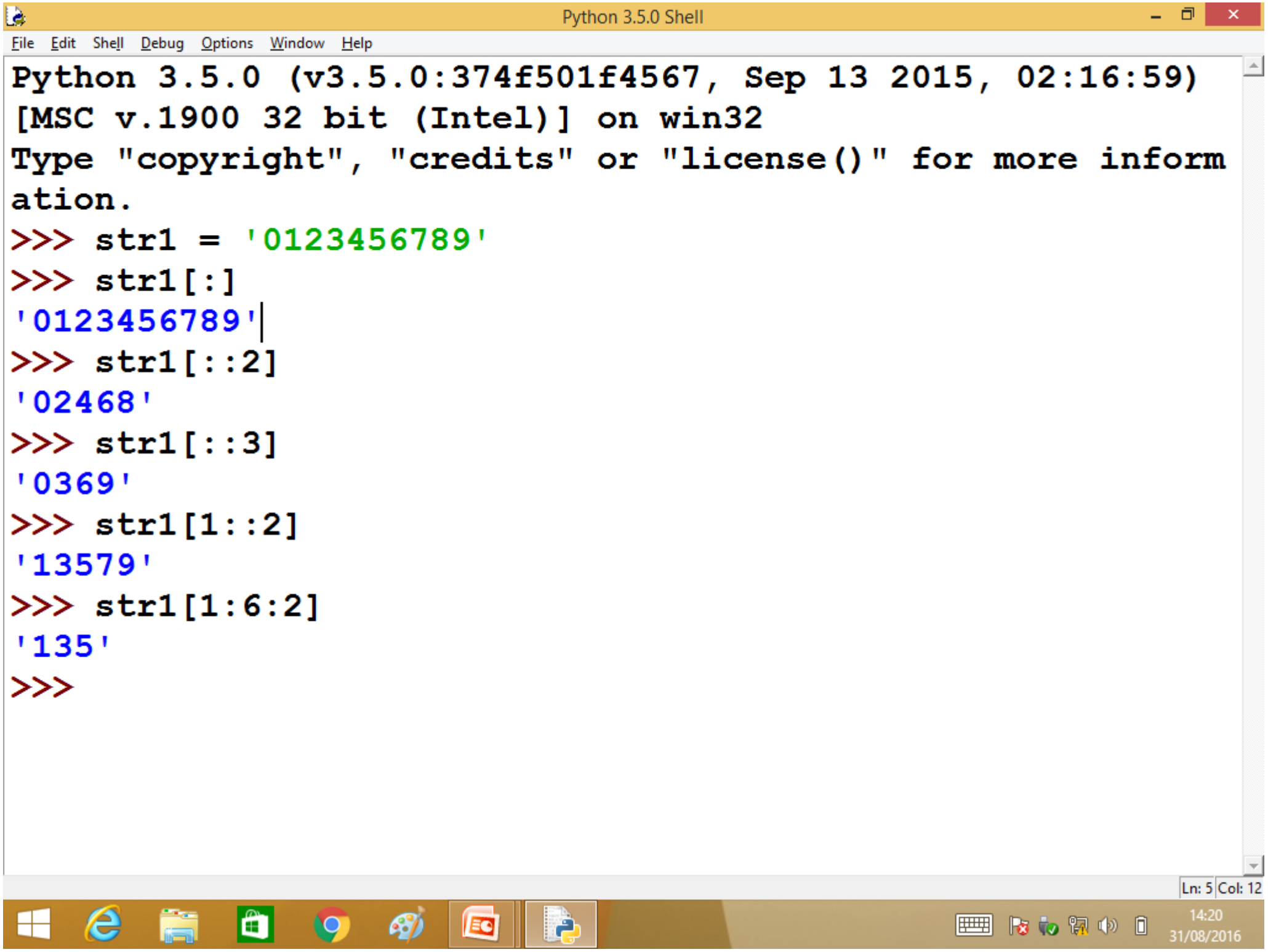
- If the second number is beyond the end of the string, it stops at the end
- If we leave off the first or last number of the slice, it is assumed to be beginning or end of the string respectively
- s = ‘spam’
- >>>s[:3]
- ‘spa’
- >>>s[1:]
- ‘pam’

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


Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59)
[MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more inform
ation.
>>> str1 = '0123456789'
>>> str1[:]
'0123456789'
>>> str1[::2]
'02468'
>>> str1[::3]
'0369'
>>> str1[1::2]
'13579'
>>> str1[1:6:2]
'135'
>>>
Ln: 5 Col: 12
14:20
31/08/2016

Extended slicing Examples

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

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

Character code Conversions

chr () – Does **inverse of ord**

```
>>> chr(115)
's'
```

Character code Conversions - Example

```
>>> c = '5'
>>> ord(c)
53
>>> x = chr(ord(c) + 1)
>>> x
'6'
```

Character code Conversions - Example

```
>>> ord('5') - ord('0')     # 53 - 48
        5
>>> int('1101', 2) # Convert binary to integer
13
>>> bin(13)    # Convert integer to binary
'0b1101'
```

int() Function

- The int() function converts the specified value into an integer number.

int(value, base)

value - A number or a string that can be converted into an integer number

base - A number representing the number format. Default value: 10

```
>>>x = int("12")
>>>print(x)
```

int() Function

```
>>>x = int("12")
>>>print(x)
12
>>>num = 13
>>>Str = '187'
>>>r = int(Str) + num
>>>print(res)
200
```

int() Function

```
>>>str = '100'
>>>print(int(str, 2))
4
>>>print(int(str, 4))
16
>>>print(int(str, 8))
64
>>>print(int(str, 16))
256
```

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

String.replace()

```
>>> S = 'splot'
>>> S = S.replace('pl', 'pamal')
>>> S
'spamalot'
>>> 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! {3} {4}' .format(1, 'dead‘, 23.5, ‘raj’)

'That is 1 dead bird! 23.4 raj'

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.

String Library

```
>>> greet = 'Hello Arun'
>>> zap = greet.lower()
>>> print (zap)
      hello arun


>>> print (‘Hi There'.upper())
           HI THERE
```

Searching a String

find() – find where in the text is the string

find() - finds the first occurrence of the substring

If the substring is not found, find() returns -1

```
>>> name = 'kumar‘
>>> pos = name.find(‘ma')
>>> print (pos)
2
```

Searching a String

```
>>> aa = “fruit”.find('z')

>>> print (aa)

-1

>>>txt = "Hello, welcome to my world."

>>>x = txt.find("welcome")

>>>print(x)

7
```

string.rstrip()

rstrip() - Remove any white spaces at the end of the string.

```
>>>txt = "     banana     "

>>>x = txt.rstrip()

>>>print("of all fruits", x, “ is my favorite")

of all fruits     banana is my favorite
```

string.rstrip()

string.rstrip(characters)

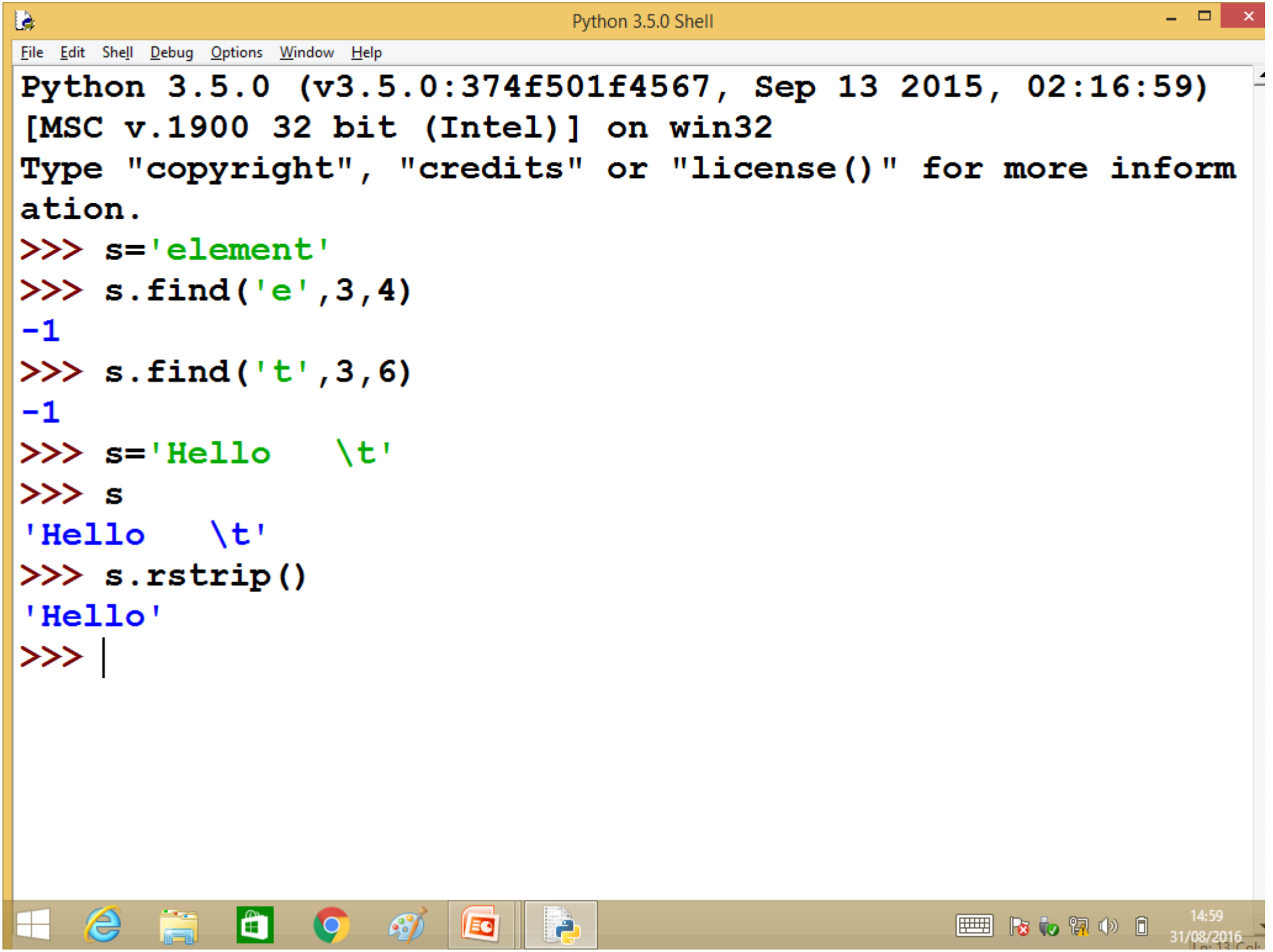
characters - Optional. A set of characters to remove as trailing characters

#Remove the trailing characters if they are commas, s, q, or w:

```
>>>txt = "banana,,,,,"
>>>x = txt.rstrip(", ")
>>>print(x)
banana
>>>txt = ",,,banana,,,,,ssqqqww....."
>>>x = txt.rstrip(",.qsw")
>>>print(x)
,,,banana
```

string.rstrip()

```
>>> line = "The knights who say Ni!\n“

>>> line.rstrip()

'The knights who say Ni!'

>>> line.upper()

'THE KNIGHTS WHO SAY NI!‘
```


Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59)
[MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more inform
ation.
>>> s='element'
>>> s.find('e',3,4)
-1
>>> s.find('t',3,6)
-1
>>> s='Hello \t'
>>> s
'Hello \t'
>>> s.rstrip()
'Hello'
>>>
14:59
31/08/2016

Other Common String Methods in Action

- >>> line = "The knights who say Ni!\n“
- >>> line**.isalpha()**
- **# Check if all the characters in the string are letters**
- False
- >>> line**.endswith('Ni!\n')**
- **#Check if the string ends with the given string**
- True
- >>> line**.startswith('The')**
- **#Check if the string starts with “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
```

Other Common String Methods in Action

```
>>> sub = 'Ni!\n'
>>> line.endswith(sub)  # End test via method call
```

or slice

```
True
>>>sub="MyVIT"
>>> line[-len(sub):] == sub
True
```

To check if all letters in a String are in Uppercase

isupper() function is used to check if all letters in a string are in upper case.

Examples

```
>>> 'a'.isupper()
False
>>> 'A'.isupper()
True
>>> 'AB'.isupper()
True
>>> 'ABc'.isupper()
False
```

To check if all letters in a String are in Lowercase

islower() function is used to check if all letters in a string are in lower case.

Examples

```
>>> 'a'.islower()
True
>>> 'aB'.islower()
False
```

To check if a sentence is in Title case

istitle() function is used to check if all letters in a string are in upper case.

Examples

```
>>> 'Apple Is A Tree'.istitle()
True


>>> 'Apple Is A tree'.istitle()
False
```