Strings

Strings

- **string: A sequence of text characters in a program.**
  - Strings start and end with quotation mark `"` or apostrophe `'` characters.
  - Examples:

```
"hello"
"This is a string"
"This, too, is a string.   It can be very long!"
```

- **A string may not span across multiple lines or contain a " character.**

```
"This is not
a legal String."

"This is not a "legal" String either."
```

Variable of Type String

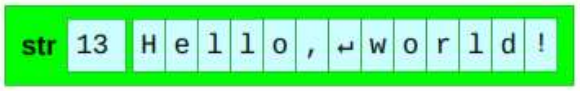
- Examples:

start= "hello"

```
tstr="This is a string"
str1="Hello\tthere\nHow are you?"
```

Strings

- **tstr="Hello, world!"**



- **Strings are Immutable and sequence**

Indexes

- **Characters in a string are numbered with *indexes* starting at 0:**
  - Example:

```
name = "P. Diddy"
```

index	0	1	2	3	4	5	6	7
character	P	.		D	i	d	d	y

- **Accessing an individual character of a string:**

  variableName `[` ***index*** `]`

  - Example:

```
print (name, "starts with", name[0])
```

  Output:

```
P. Diddy starts with P
```

Indexes

[start:end]
Indexes refer to places the knife "cuts."

Defaults are beginning of sequence and end of sequence.

Positive and negative indices

```
>>> s = 'a string'
```

Positive index: count from the left, starting with 0

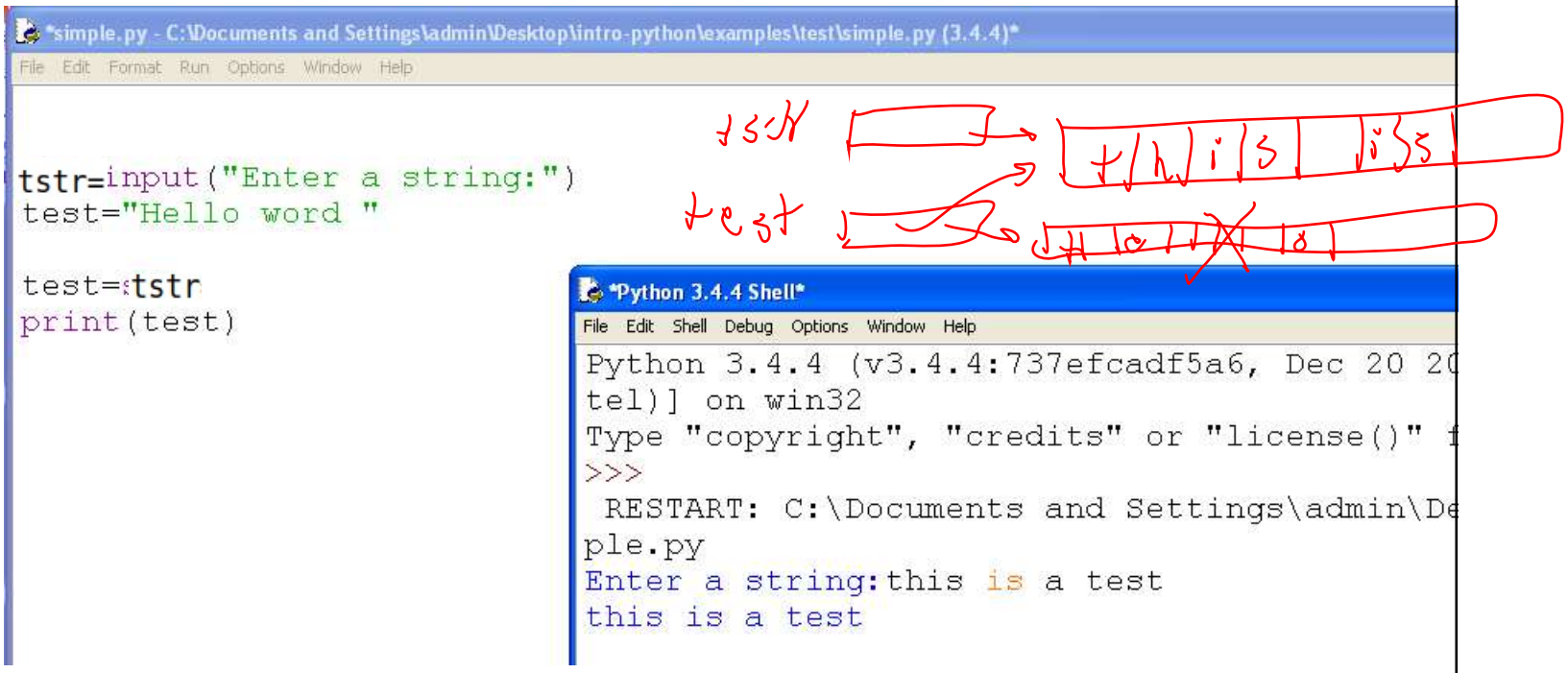
```
>>> s[2]
's'
```

Negative index: count from right, starting with –1

```
>>> s[-3]
'i'
```

Updating Strings

- **You can "update" an existing string by (re)assigning a variable to another string. The new value can be related to its previous value or to a completely different string altogether. For example**



Updating Strings

- **You can not change an element of a string because they immutables**

```
>>> S = 'Spam'
>>> S
'Spam'

>>> S[0] = 'z'              # Immutable objects cannot be changed
...error text omitted...
TypeError: 'str' object does not support item assignment
>>> S[0]                    # The first item in S, indexing by zero-based position
'S'
>>> S[1]                    # The second item from the left
'p'
```

Escape Characters

- **Following table is a list of escape or non-printable characters that can be represented with backslash notation.**
- **An escape character gets interpreted; in a single quoted as well as double quoted strings.**

Backslash notation	Hexadecimal character	Description
\a	0x07	Bell or alert
\b	0x08	Backspace
\cx		Control-x
\C-x		Control-x
\e	0x1b	Escape
\f	0x0c	Formfeed
\M-\C-x		Meta-Control-x
\n	0x0a	Newline
\nnn		Octal notation, where n is in the range 0.7
\r	0x0d	Carriage return
\s	0x20	Space
\t	0x09	Tab
\v	0x0b	Vertical tab
\x		Character x
\xnn		Hexadecimal notation, where n is in the range 0.9, a.f, or A.F

Escape Characters

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

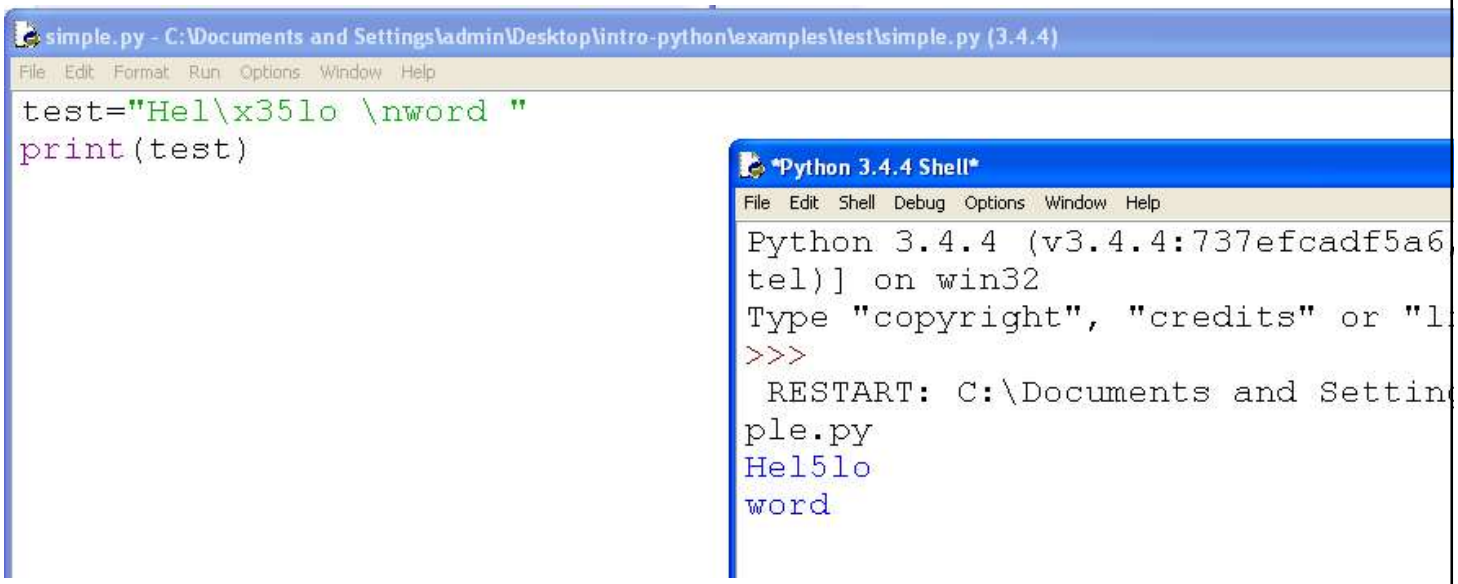
```
test="Hel\tlo \nword "
print(test)
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6,
tel)] on win32
Type "copyright", "credits" or "lic
>>>
 RESTART: C:\Documents and Settings
ple.py
Hel     lo
word
```

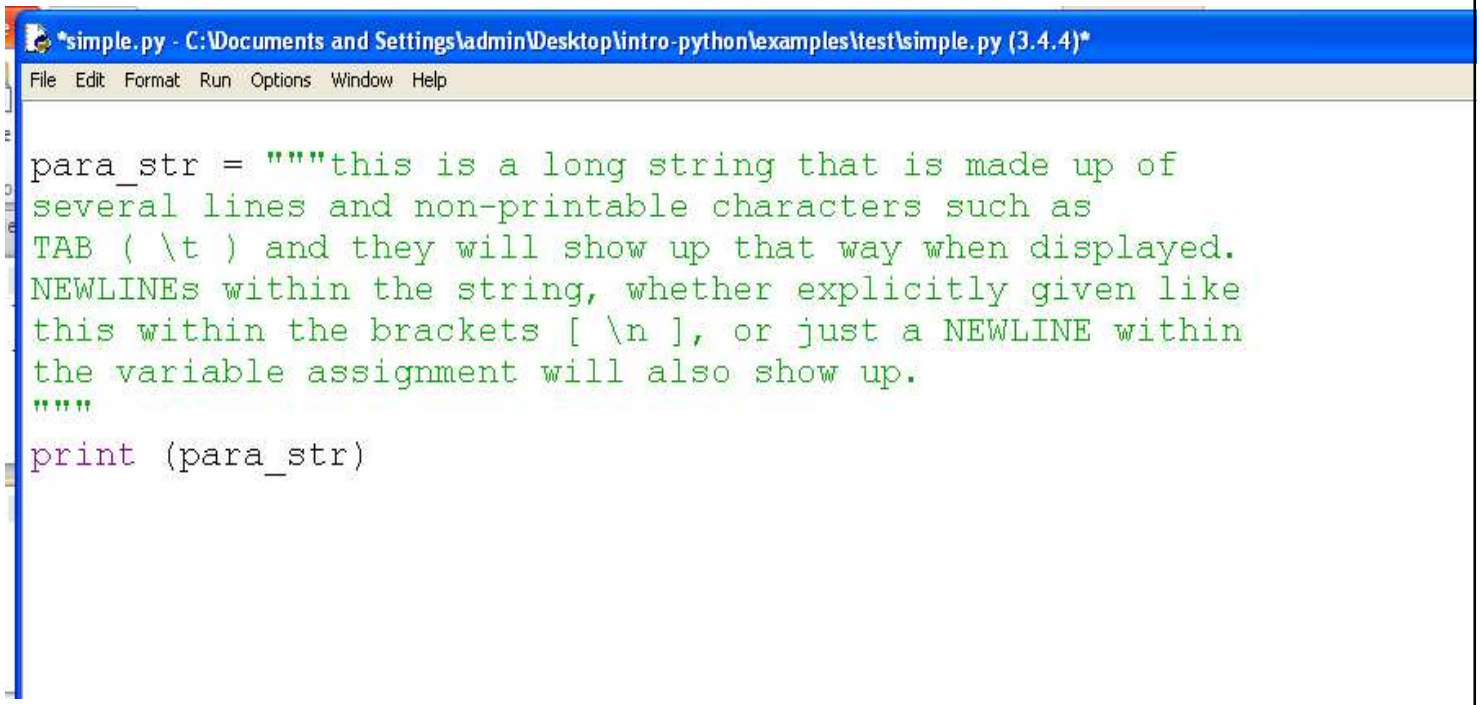
Escape Characters



Triple Quotes

- **Python's triple quotes comes to the rescue by allowing strings to span multiple lines, including verbatim NEWLINEs, TABs, and any other special characters.**
- **The syntax for triple quotes consists of three consecutive single or double quotes.**

Triple Quotes



String Special Operators

- **Assume string variable a holds 'Hello' and variable b holds 'Python', then −**

Operator	Description	Example
+	Concatenation - Adds values on either side of the operator	a + b will give HelloPython
*	Repetition - Creates new strings, concatenating multiple copies of the same string	a*2 will give -HelloHello
[]	Slice - Gives the character from the given index	a[1] will give e
[:]	Range Slice - Gives the characters from the given range	a[1:4] will give ell
in	Membership - Returns true if a character exists in the given string	H in a will give 1
not in	Membership - Returns true if a character does not exist in the given string	M not in a will give 1
r/R	Raw String - Suppresses actual meaning of Escape characters. The syntax for raw strings is exactly the same as for normal strings with the exception of the raw string operator, the letter "r," which precedes the quotation marks. The "r" can be lowercase (r) or uppercase (R) and must be placed immediately preceding the first quote mark.	print r'\n' prints \n and print R'\n'prints \n
%	Format - Performs String formatting	See at next section

Slicing: return copy of a string

```
>>> s = 'a string'
```

Return a copy of the container with a subset of the original members. Start copying at the first index, and stop copying *before* second.

```
>>> s[2:5]
'str'
```

Negative indices count from end

```
>>> s[4:-1]
'rin'
```

Slicing

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

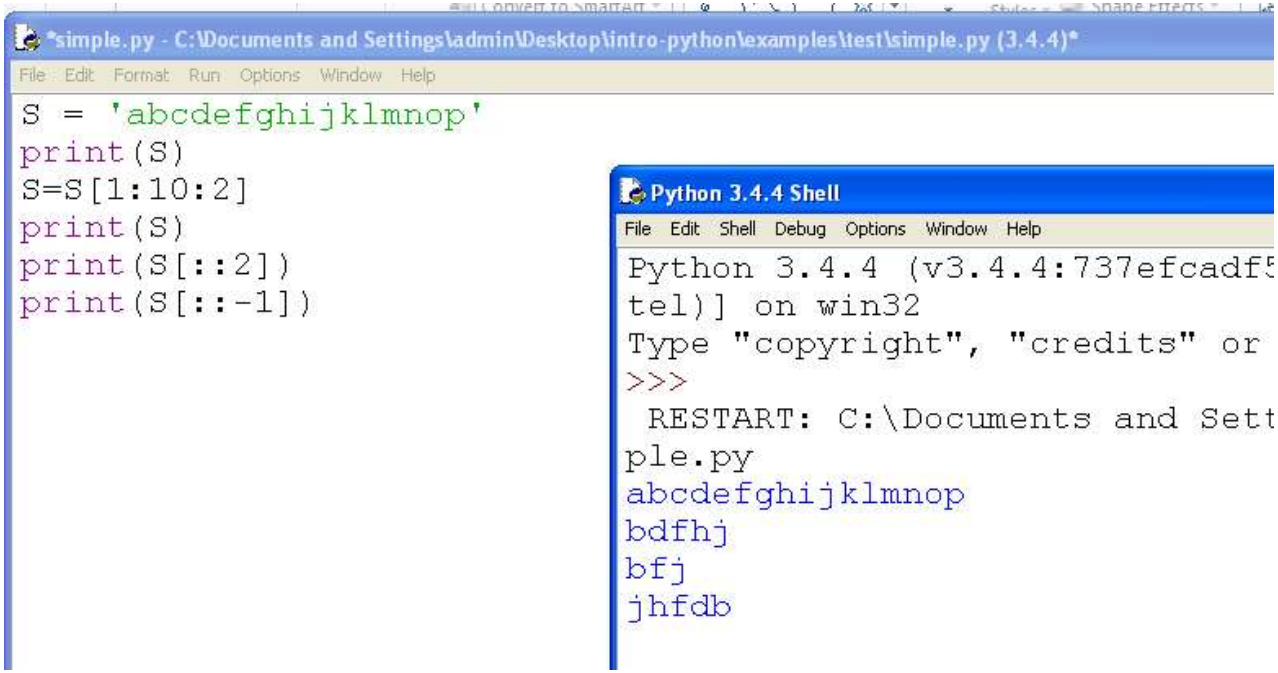
```
tstr=input("Enter a string:")

x=tstr[1]
print(x)

x=tstr[2:8]
print(x)

x=tstr[-3]
print(x)

x=tstr[-3:-1]
print(x)
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

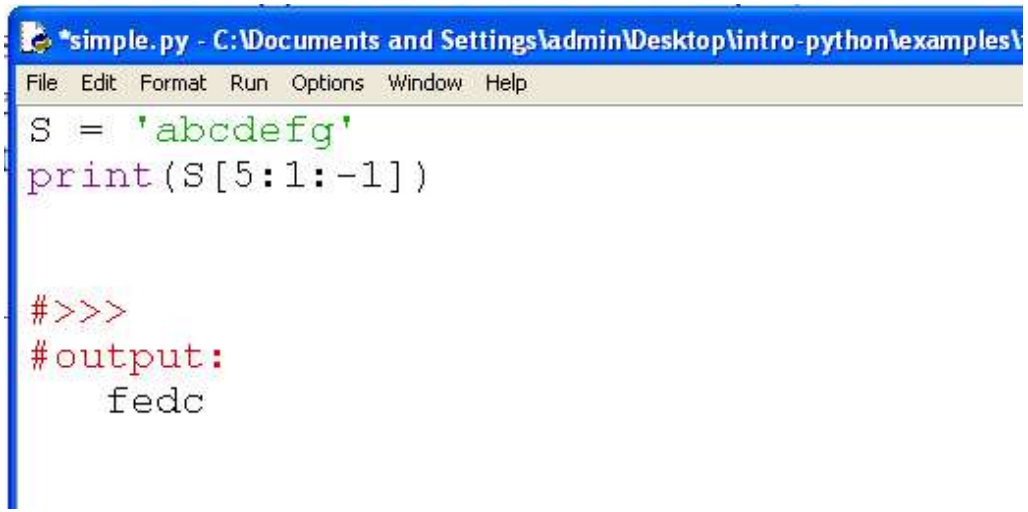
```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 201
Type "copyright", "credits" or "license()" fo
>>>
 RESTART: C:\Documents and Settings\admin\Des
Enter a string:this is a test
h
is is
e
es
>>>
```

Slicing

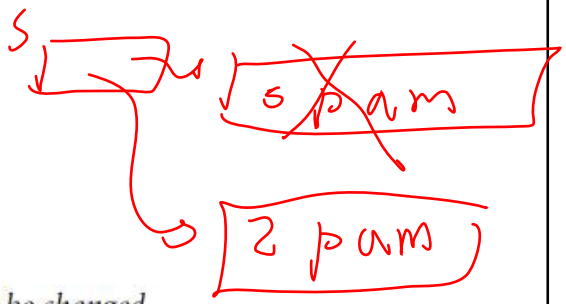

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)
File Edit Format Run Options Window Help
S = 'abcdefghijklmnop'
print(S)
S=S[1:10:2]
print(S)
print(S[::2])
print(S[::-1])
Python 3.4.4 Shell
File Edit Shell Debug Options Window Help
Python 3.4.4 (v3.4.4:737efcadf5
tel)] on win32
Type "copyright", "credits" or
>>>
RESTART: C:\Documents and Sett
ple.py
abcdefghijklmnop
bdfhj
bfj
jhfdb

Slicing

- With a negative stride, the meanings of the first two bounds are essentially reversed.
- That is, the slice S[5:1:−1] fetches the items from 2 to 5, in reverse order (the result
- contains items from offsets 5, 4, 3, and 2):

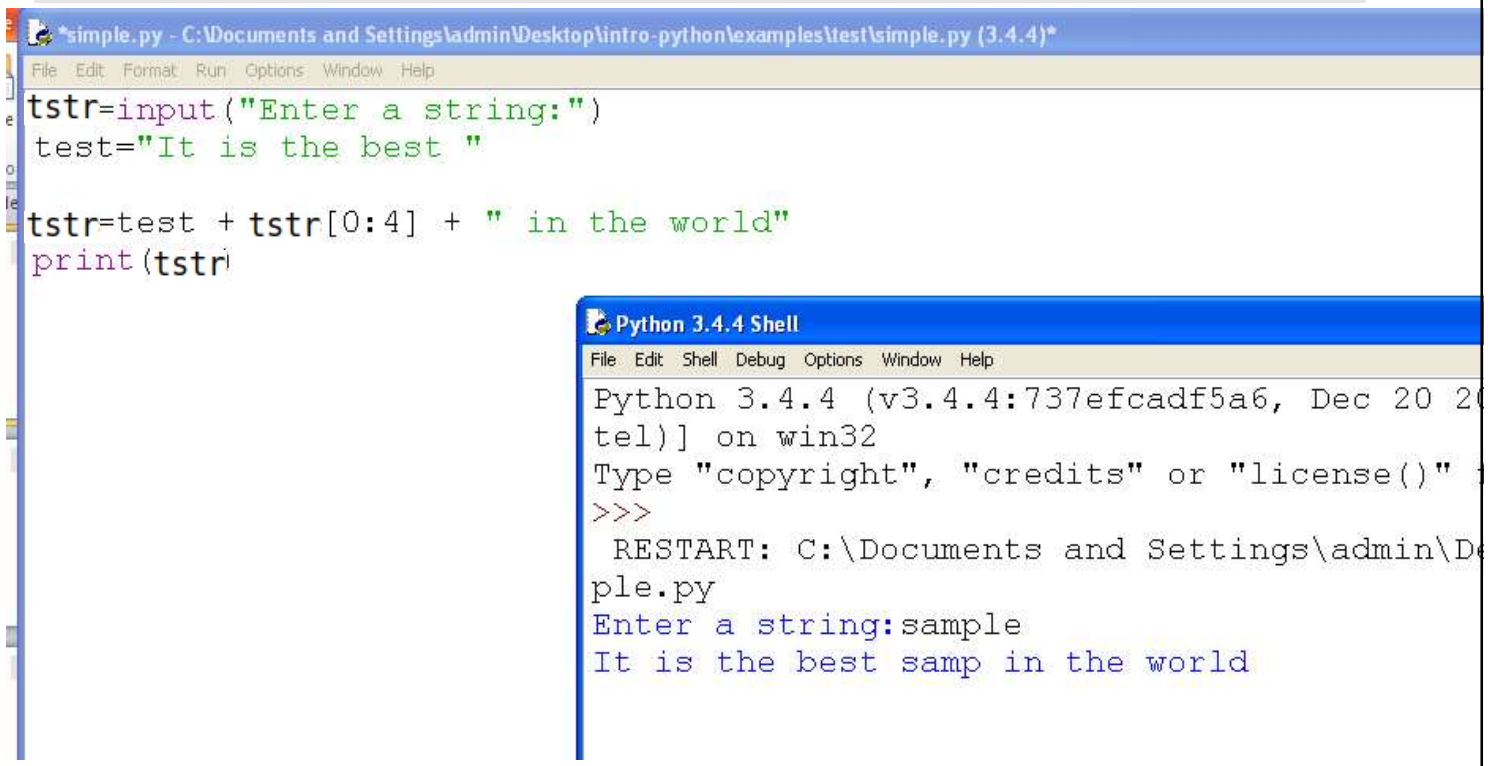


Slicing for updating

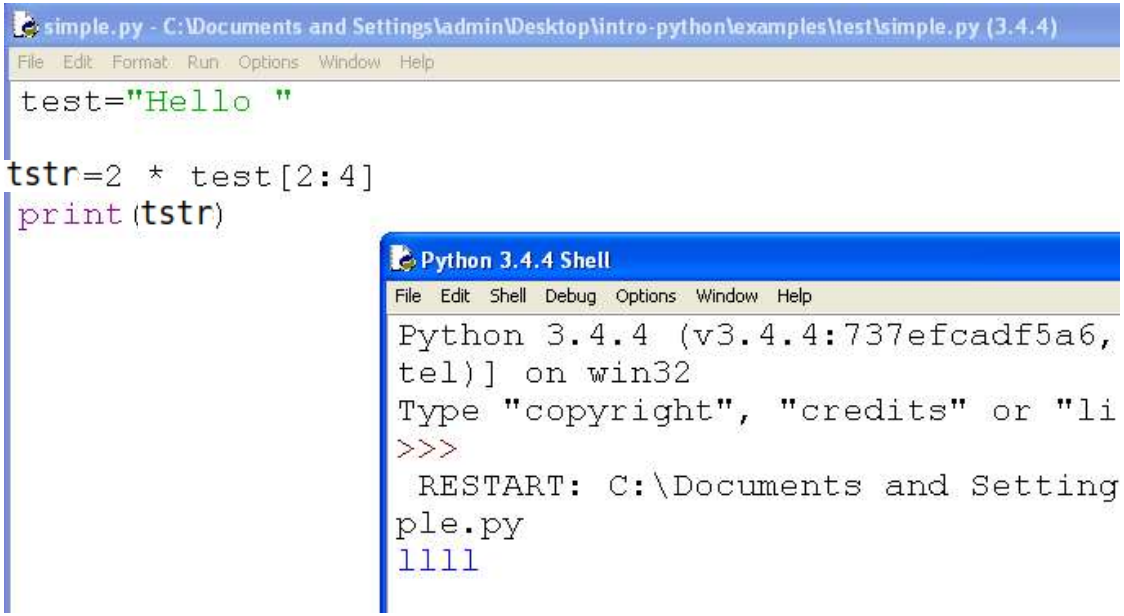



```
>>> S
'Spam'

>>> S[0] = 'z'              # Immutable objects cannot be changed
...error text omitted...
TypeError: 'str' object does not support item assignment

>>> S = 'z' + S[1:]         # But we can run expressions to make new objects
>>> S
'zpam'
```

String Special Operators



String Special Operators



Operator	Description	Example
+	Concatenation - Adds values on either side of the operator	a + b will give HelloPython
*	Repetition - Creates new strings, concatenating multiple copies of the same string	a*2 will give -HelloHello
[]	Slice - Gives the character from the given index	a[1] will give e
[:]	Range Slice - Gives the characters from the given range	a[1:4] will give ell
in	Membership - Returns true if a character exists in the given string	H in a will give 1
not in	Membership - Returns true if a character does not exist in the given string	M not in a will give 1
r/R	Raw String - Suppresses actual meaning of Escape characters. The syntax for raw strings is exactly the same as for normal strings with the exception of the raw string operator, the letter "r," which precedes the quotation marks. The "r" can be lowercase (r) or uppercase (R) and must be placed immediately preceding the first quote mark.	print r'\n' prints \n and print R'\n'prints \n
%	Format - Performs String formatting	See at next section

String Formatting Operator

- One of Python's coolest features is the string format operator %. This operator is unique to strings and makes up for the pack of having functions from C's printf() family.

String Formatting Operator

Format Symbol	Conversion
%c	character
%s	string conversion via str() prior to formatting
%i	signed decimal integer
%d	signed decimal integer
%u	unsigned decimal integer
%o	octal integer
%x	hexadecimal integer (lowercase letters)
%X	hexadecimal integer (UPPERcase letters)
%e	exponential notation (with lowercase 'e')
%E	exponential notation (with UPPERcase 'E')
%f	floating point real number
%g	the shorter of %f and %e
%G	the shorter of %f and %E

String Formatting Operator

Symbol	Functionality
*	argument specifies width or precision
-	left justification
+	display the sign
<sp>	leave a blank space before a positive number
#	add the octal leading zero ('0') or hexadecimal leading '0x' or '0X', depending on whether 'x' or 'X' were used.
0	pad from left with zeros (instead of spaces)
%	'%%' leaves you with a single literal '%'
(var)	mapping variable (dictionary arguments)
m.n.	m is the minimum total width and n is the number of digits to display after the decimal point (if appl.)

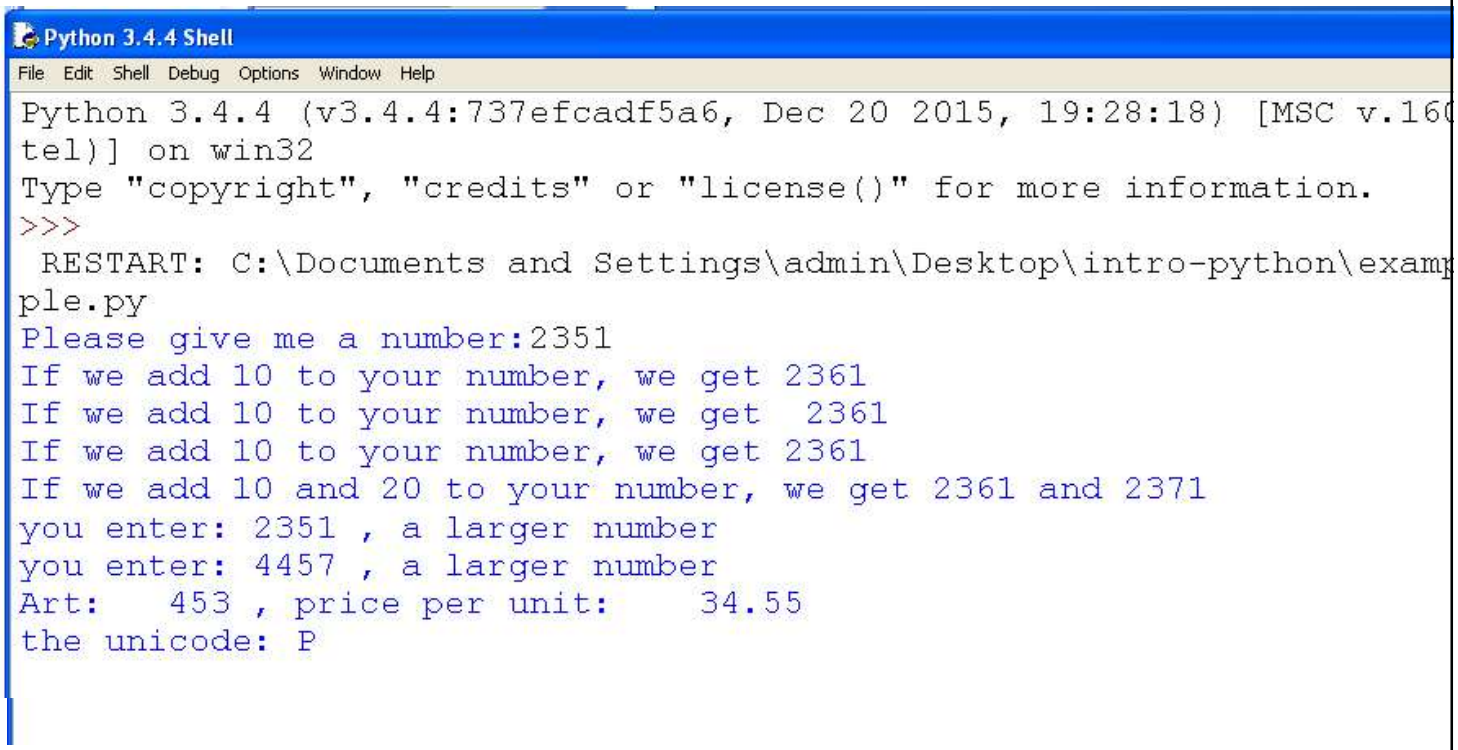
String Formatting Operator

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
response = input("Please give me a number:")

number = int(response)
plusTen = number + 10

print ("If we add 10 to your number, we get " + str(plusTen))
print ("If we add 10 to your number, we get ", plusTen)
print ("If we add 10 to your number, we get %s" % plusTen)
plusTwenty = number + 20
print ("If we add 10 and 20 to your number, we get %s and %s" % (plusTen, plusTwenty))
print ("you enter: %d , a larger number" %number)
print ("you enter: %o , a larger number" %number)
print ("Art: %5d , price per unit: %8.2f" %(453, 34.5472))
print (u"the unicode: \u0420")
```

String Formatting Operator



String comparing Operator

- The standard comparisons (<, <=, >, >=, ==, !=) apply to strings. These comparisons use the standard character-by-character comparison rules for ASCII or Unicode.

String comparing Operator

*simpletuple.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/19

File Edit Format Run Options Window Help

```
s1='this is a book'
s2='this si a book'
if(s1<s2):
    print('s1 is less than s2')

#output
        s1 is less than s2
```

built-in functions for strings

- bin(*x*) Convert an integer number to a binary string. The result is a valid Python expression
- hex(*x*) Convert an integer number to a lowercase hexadecimal string prefixed with “0x”.
- oct(*x*) Convert an integer number to an octal string. The result is a valid Python expression. If *x* is not a Python int object, it has to define an __index__() method that returns an integer.
- chr(*i*) Return the string representing a character whose Unicode code point is the integer *i*. For example, chr(97) returns the string 'a'. This is the inverse of ord(). The valid range for the argument is from 0 through 1,114,111 (0x10FFFF in base 16). ValueError will be raised if *i* is outside that range.
- ord(*c*) Given a string representing one Unicode character, return an integer representing the Unicode code point of that character. For example, ord('a') returns the integer 97 and ord('\u2020') returns 8224. This is the inverse of chr().

built-in functions for strings

- len(*s*) Return the length (the number of items) of an object. The argument may be a sequence (such as a string, bytes, tuple, list, or range) or a collection (such as a dictionary, set, or frozen set).
- repr(*object*) Return a string containing a printable representation of an object. For many types, this function makes an attempt to return a string that would yield an object with the same value when passed to eval(), otherwise the representation is a string enclosed in angle brackets that contains the name of the type of the object together with additional information often including the name and address of the object. A class can control what this function returns for its instances by defining a __repr__() method.
- str(object) Return a str version of object
- Id(object) Return the **identity** of the object

Len Method

Python String len() Method

Advertisements

Previous Page Next Page

Description

The method **len()** returns the length of the string.

Syntax

Following is the syntax for **len()** method –

```
len( str )
```

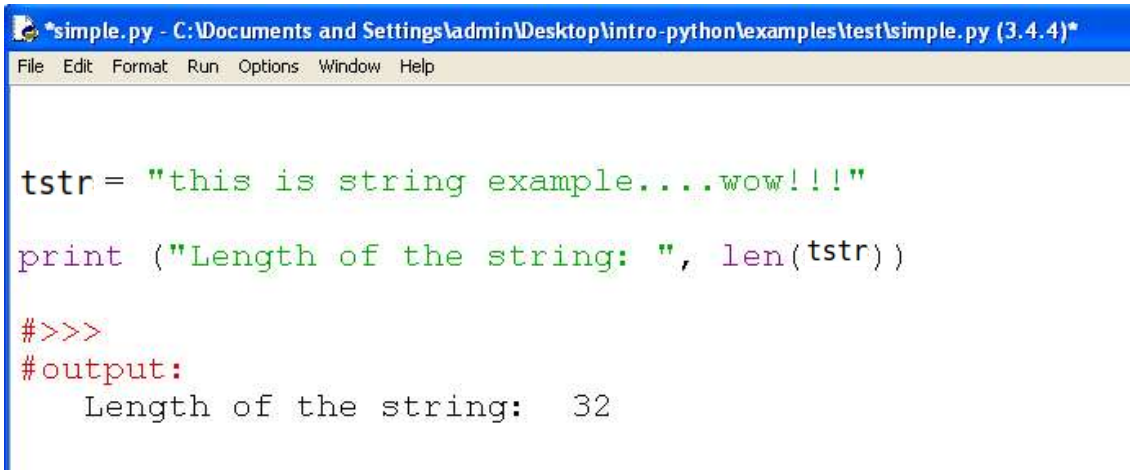
Parameters

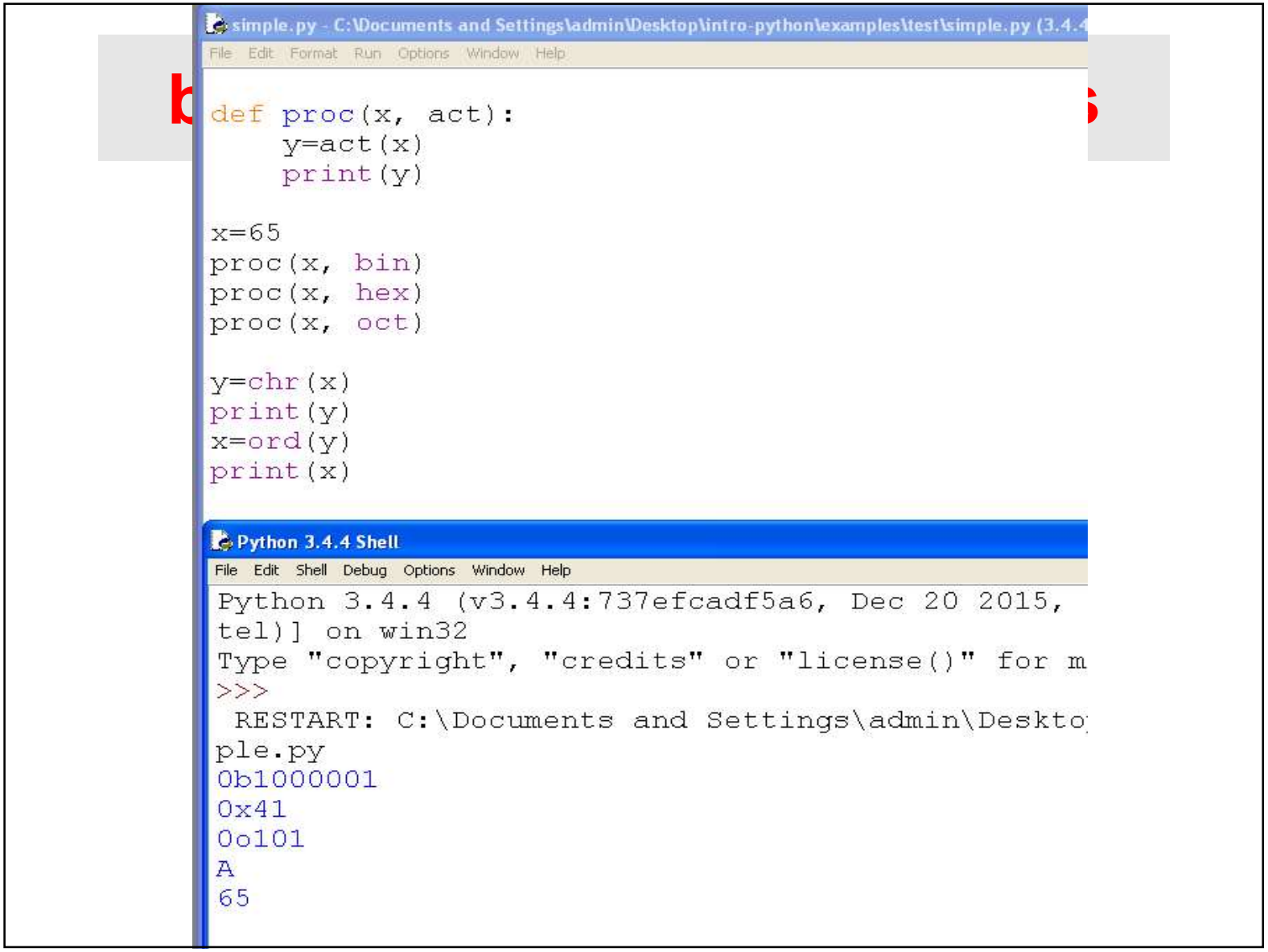
- NA

Return Value

This method returns the length of the string.

Len Methods


simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)
File Edit Format Run Options Window Help
tstr = "this is string example....wow!!!"
print ("Length of the string: ", len(tstr))
#>>>
#output:
Length of the string: 32


simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4
File Edit Format Run Options Window Help
def proc(x, act):
y=act(x)
print(y)
x=65
proc(x, bin)
proc(x, hex)
proc(x, oct)
y=chr(x)
print(y)
x=ord(y)
print(x)
Python 3.4.4 Shell
File Edit Shell Debug Options Window Help
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015,
tel)] on win32
Type "copyright", "credits" or "license()" for m
>>>
RESTART: C:\Documents and Settings\admin\Deskto
ple.py
0b1000001
0x41
0o101
A
65

String Type Conversion

- str(x) : convert x to a string
- int(x) : convert string x to a int number
- float(x) : convert string x to a float number

Type specific Methods

- **Python includes the following Type specific methods to manipulate strings**

1	**capitalize()** Capitalizes first letter of string
2	**center(width, fillchar)** Returns a space-padded string with the original string centered to a total of width columns.
3	**count(str, beg= 0,end=len(string))** Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given.
4	**decode(encoding='UTF-8',errors='strict')** Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding.
5	**encode(encoding='UTF-8',errors='strict')** Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'.
6	**endswith(suffix, beg=0, end=len(string))** Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise.
7	**expandtabs(tabsize=8)** Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided.
8	**find(str, beg=0 end=len(string))** Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise.
9	**index(str, beg=0, end=len(string))**

9	**index(str, beg=0, end=len(string))** Same as find(), but raises an exception if str not found.
10	**isalnum()** Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.
11	**isalpha()** Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.
12	**isdigit()** Returns true if string contains only digits and false otherwise.
13	**islower()** Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.
14	**isnumeric()** Returns true if a unicode string contains only numeric characters and false otherwise.

15	**isspace()** Returns true if string contains only whitespace characters and false otherwise.
16	**istitle()** Returns true if string is properly "titlecased" and false otherwise.
17	**isupper()** Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.
18	**join(seq)** Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.
19	**len(string)** Returns the length of the string
20	**ljust(width[, fillchar])** Returns a space-padded string with the original string left-justified to a total of width columns.

21	**lower()** Converts all uppercase letters in string to lowercase.
22	**lstrip()** Removes all leading whitespace in string.
23	**maketrans()** Returns a translation table to be used in translate function.
24	**max(str)** Returns the max alphabetical character from the string str.
25	**min(str)** Returns the min alphabetical character from the string str.
26	**replace(old, new [, max])** Replaces all occurrences of old in string with new or at most max occurrences if max given.
27	**rfind(str, beg=0,end=len(string))** Same as find(), but search backwards in string.

28	**rindex(str, beg=0, end=len(string))** Same as index(), but search backwards in string.
29	**rjust(width,[, fillchar])** Returns a space-padded string with the original string right-justified to a total of width columns.
30	**rstrip()** Removes all trailing whitespace of string.
31	**split(str="", num=string.count(str))** Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given.
32	**splitlines(num=string.count('\n'))** Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed.
33	**startswith(str, beg=0,end=len(string))** Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise.
34	**strip([chars])** Performs both lstrip() and rstrip() on string

35	**swapcase()** Inverts case for all letters in string.
36	**title()** Returns "titlecased" version of string, that is, all words begin with uppercase and the rest are lowercase.
37	**translate(table, deletechars="")** Translates string according to translation table str(256 chars), removing those in the del string.
38	**upper()** Converts lowercase letters in string to uppercase.
39	**zfill (width)** Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero).
40	**isdecimal()** Returns true if a unicode string contains only decimal characters and false otherwise.

Python String center() Method

Advertisements

Previous Page Next Page

The method center() returns centered in a string of length width. Padding is done using the specified fillchar. Default filler is a space.

Syntax

```
str.center(width[, fillchar])
```

Parameters

- **width** -- This is the total width of the string.
- **fillchar** -- This is the filler character.

Return Value

This method returns centered in a string of length width.

Center Method

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
tstr = "this is string example....wow!!!";

print ("str.center(40, 'a') : ", tstr.center(40, 'a'))
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MSC v.1
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\intro-python\exar
ple.py
str.center(40, 'a') :  aaaathis is string example....wow!!!aaaa
```

Python String count() Method

Advertisements

Previous Page Next Page

Description

The method **count()** returns the number of occurrences of substring sub in the range [start, end]. Optional arguments start and end are interpreted as in slice notation.

Syntax

```
str.count(sub, start= 0,end=len(string))
```

Parameters

- **sub** -- This is the substring to be searched.
- **start** -- Search starts from this index. First character starts from 0 index. By default search starts from 0 index.
- **end** -- Search ends from this index. First character starts from 0 index. By default search ends at the last index.

Return Value

Centered in a string of length width.

Count method

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
tstr = "this is string example....wow!!!";

sub = "i";
print ("str.count(sub, 4, 40) : ", tstr.count(sub, 4, 40))
sub = "wow";
print ("str.count(sub) : ", tstr.count(sub))
```

```
 RESTART: C:\Documents and Settings\admin\Desktop\i
ple.py
str.count(sub, 4, 40) :  2
str.count(sub) :  1
```

Python String encode() Method

Advertisements

Previous Page Next Page

Description

The method **encode()** returns an encoded version of the string. Default encoding is the current default string encoding. The errors may be given to set a different error handling scheme.

Syntax

```
str.encode(encoding='UTF-8',errors='strict')
```

Parameters

- **encoding** -- This is the encodings to be used. For a list of all encoding schemes please visit: Standard Encodings.
- **errors** -- This may be given to set a different error handling scheme. The default for errors is 'strict', meaning that encoding errors raise a UnicodeError. Other possible values are 'ignore', 'replace', 'xmlcharrefreplace', 'backslashreplace' and any other name registered via codecs.register_error( ).

Return Value

Decoded string.

Python String decode() Method

Advertisements

Previous Page Next Page

Description

The method **decode()** decodes the string using the codec registered for *encoding*. It defaults to the default string encoding.

Syntax

```
Str.decode(encoding='UTF-8',errors='strict')
```

Parameters

- **encoding** -- This is the encodings to be used. For a list of all encoding schemes please visit: Standard Encodings.
- **errors** -- This may be given to set a different error handling scheme. The default for errors is 'strict', meaning that encoding errors raise a UnicodeError. Other possible values are 'ignore', 'replace', 'xmlcharrefreplace', 'backslashreplace' and any other name registered via codecs.register_error()..

Return Value

Decoded string.

Encode Method

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

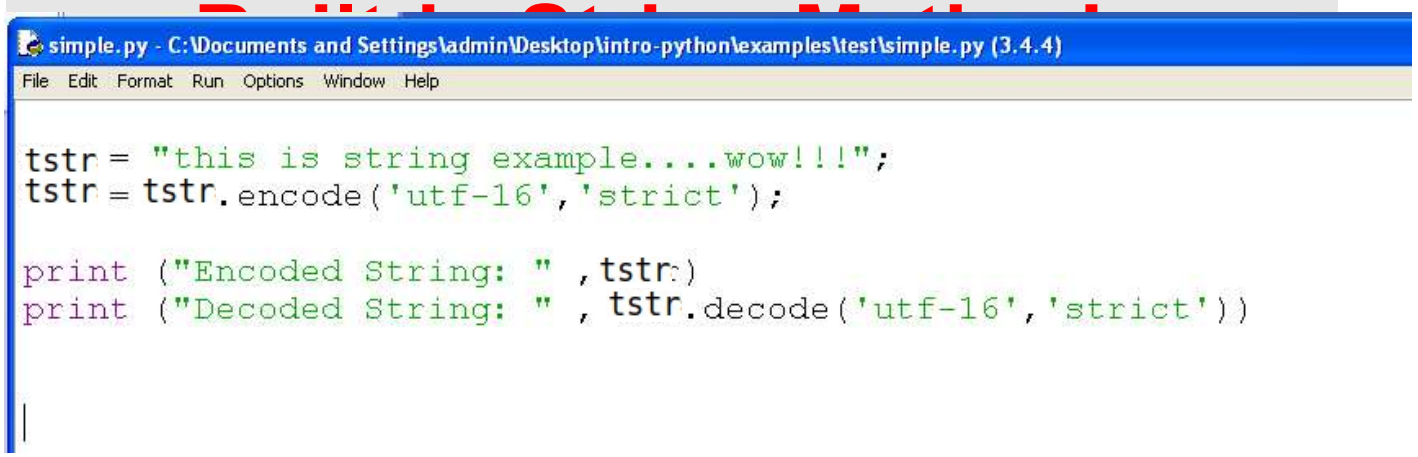
File Edit Format Run Options Window Help

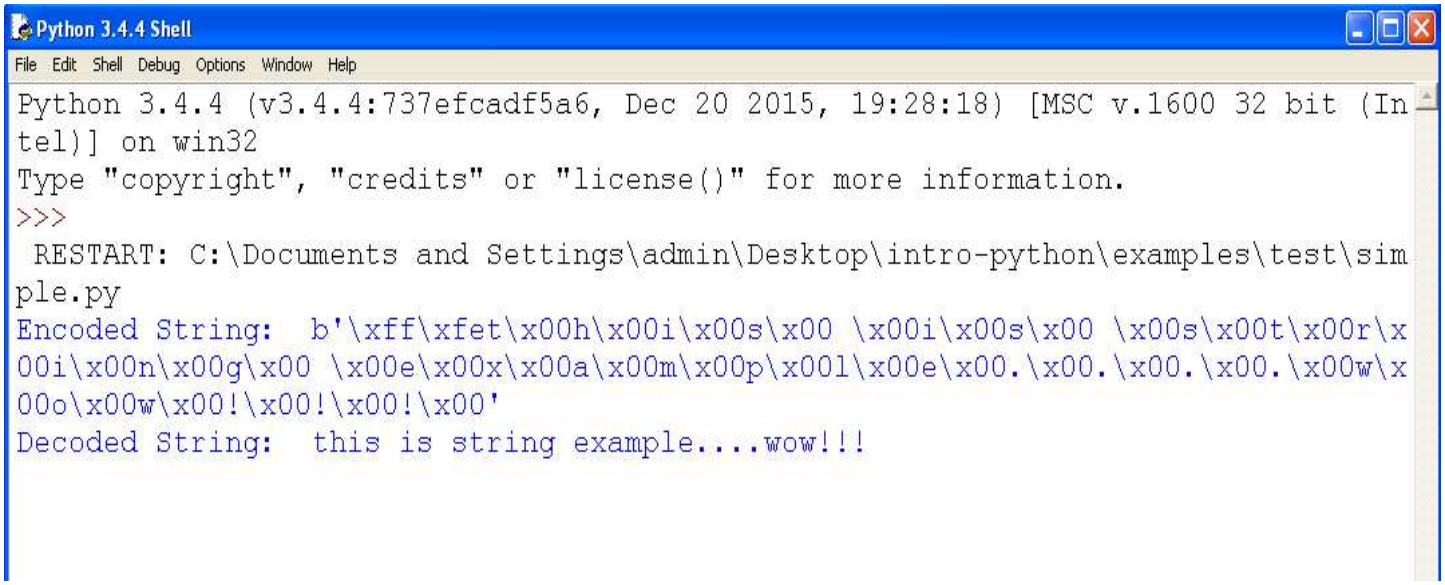
```
original = '27岁少妇生孩子后变老'

encoded = original.encode('utf-8','strict')
print("Encoded String: ", encoded)
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\intro-python\examples\test\sim
ple.py
Encoded String:  b'27\xe5\xb2\x81\xe5\xb0\x91\xe5\xa6\x87\xe7\x94\x9f\xe5\xad\xa
9\xe5\xad\x90\xe5\x90\x8e\xe5\x8f\x98\xe8\x80\x81'
```


simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)
File Edit Format Run Options Window Help
tstr = "this is string example....wow!!!";
tstr = tstr.encode('utf-16','strict');
print ("Encoded String: " ,tstr)
print ("Decoded String: " , tstr.decode('utf-16','strict'))


Python 3.4.4 Shell
File Edit Shell Debug Options Window Help
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py
Encoded String: b'\xff\xfet\x00h\x00i\x00s\x00 \x00i\x00s\x00 \x00s\x00t\x00r\x00i\x00n\x00g\x00 \x00e\x00x\x00a\x00m\x00p\x00l\x00e\x00.\x00.\x00.\x00.\x00w\x00o\x00w\x00!\x00!\x00!\x00'
Decoded String: this is string example....wow!!!

Python String find() Method

Advertisements

Previous Page Next Page

Description

It determines if string *str* occurs in string, or in a substring of string if starting index *beg* and ending index *end* are given.

Syntax

```
str.find(str, beg=0 end=len(string))
```

Parameters

- **str** -- This specifies the string to be searched.
- **beg** -- This is the starting index, by default its 0.
- **end** -- This is the ending index, by default its equal to the lenght of the string.

Return Value

Index if found and -1 otherwise.

Find Method

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
str1 = "this is string example....wow!!!";
str2 = "exam";

print (str1.find(str2))
print (str1.find(str2, 10))
print (str1.find(str2, 40))



>>>
    output:
        15
        15
        -1
```

Python String isalnum() Method

Advertisements

Previous Page Next Page

Description

The method **isalnum()** checks whether the string consists of alphanumeric characters.

Syntax

Following is the syntax for **isalnum()** method:

```
str.isalnum()
```

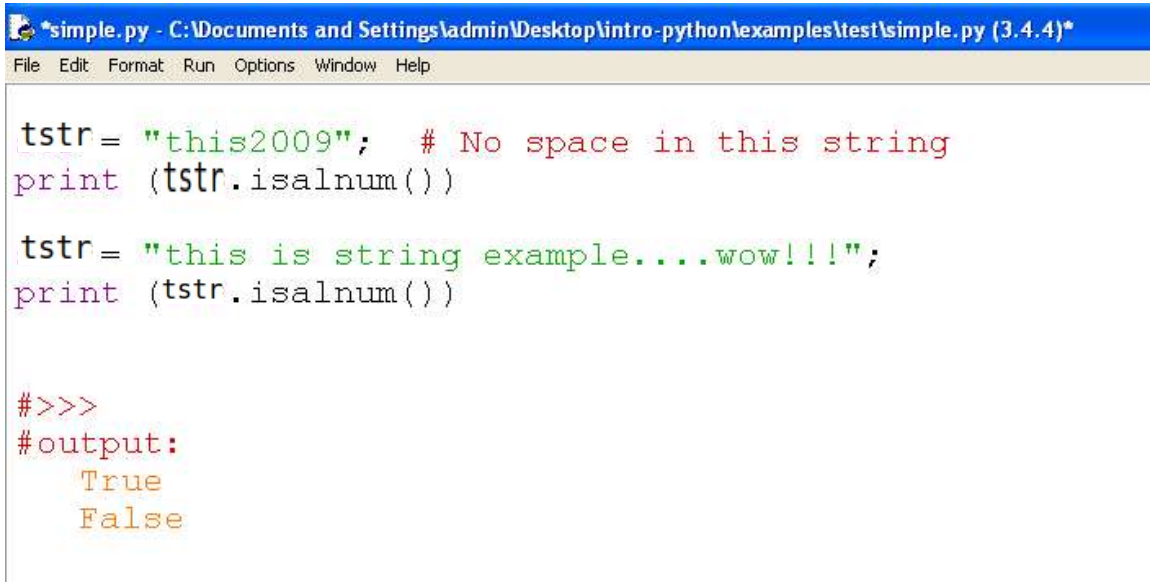
Parameters

- NA

Return Value

This method returns true if all characters in the string are alphanumeric and there is at least one character, false otherwise.

isalnum Method



Python String replace() Method

Advertisements

Previous Page Next Page

Description

The method **replace()** returns a copy of the string in which the occurrences of *old* have been replaced with *new*, optionally restricting the number of replacements to *max*.

Syntax

Following is the syntax for **replace()** method –

```
str.replace(old, new[, max])
```

Parameters

- **old** -- This is old substring to be replaced.
- **new** -- This is new substring, which would replace old substring.
- **max** -- If this optional argument max is given, only the first count occurrences are replaced.

Return Value

This method returns a copy of the string with all occurrences of substring old replaced by new. If the optional argument max is given, only the first count occurrences are replaced.

Replace Method

replace.py - /home/nowzari/Desktop/python/examples/13-strings/replace.py (3.6.6)

File Edit Format Run Options Window Help

```
tstr = "this is string example....wow!!! this is really string";
print (tstr.replace("is", "was"))
print (tstr.replace("is", "was", 3))


#>>>
#output:

thwas was string example....wow!!! thwas was really string
thwas was string example....wow!!! thwas is really string
```

Replace and find

replace.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/14-strings/replace.py (3.4.4)

File Edit Format Run Options Window Help

```
S = 'xxxxSPAMxxxxSPAMxxxx'
print(S)
where = S.find('SPAM')              # Search for position
S = S[:where] + 'EGGS' + S[(where+4):]
print(S)
S = 'xxxxSPAMxxxxSPAMxxxx'
S=S.replace('SPAM', 'EGGS')         # Replace all
print(S)
S = 'xxxxSPAMxxxxSPAMxxxx'
S=S.replace('SPAM', 'EGGS', 1)      # Replace one
print(S)

#output
xxxxSPAMxxxxSPAMxxxx
xxxxEGGSxxxxSPAMxxxx
xxxxEGGSxxxxEGGSxxxx
xxxxEGGSxxxxSPAMxxxx
```

Python String maketrans() Method

Advertisements

Previous Page Next Page

Description

The method **maketrans()** returns a translation table that maps each character in the *intabstring* into the character at the same position in the *outtab* string. Then this table is passed to the translate( ) function.

Note: Both intab and outtab must have the same length.

Syntax

Following is the syntax for **maketrans()** method –

```
str.maketrans(intab, outtab)
```

Parameters

- **intab** -- This is the string having actual characters.
- **outtab** -- This is the string having corresponding mapping character.

Return Value

This method returns a translate table to be used translate( ) function.

Python String translate() Method

Previous Page Next Page

Description

The method **translate()** returns a copy of the string in which all characters have been translated using *table* (constructed with the maketrans() function in the string module), optionally deleting all characters found in the string *deletechars*.

Syntax

Following is the syntax for **translate()** method –

```
str.translate(table);
```

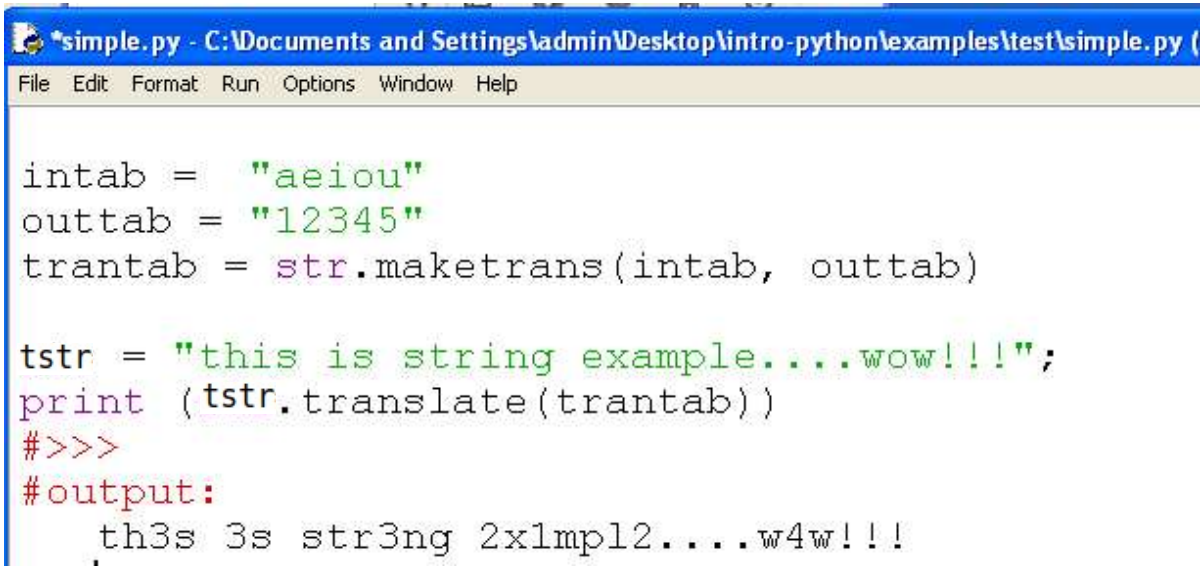
Parameters

- **table** -- You can use the maketrans() helper function in the string module to create a translation table.

Return Value

This method returns a translated copy of the string.

Maketrans Method



Rjust Method

Description

The method **rjust()** returns the string right justified in a string of length *width*. Padding is done using the specified *fillchar* (default is a space). The original string is returned if width is less than len(s).

Syntax

Following is the syntax for **rjust()** method –

```
str.rjust(width[, fillchar])
```

Parameters

- **width** -- This is the string length in total after padding.
- **fillchar** -- This is the filler character, default is a space.

Return Value

This method returns the string right justified in a string of length width. Padding is done using the specified fillchar (default is a space). The original string is returned if width is less than len(s).

Rjust Method

*simple.py - /home/nowzari/Desktop/python/python-my/python/exa

File Edit Format Run Options Window Help

```
str1 = "this is string example";
m=str1.rjust(40,'2')
print(m)

>> output

2222222222222222this is string example
```

Rjust Method

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
x=1
while(x<11) :
      print(repr(x).rjust(2), repr(x*x).rjust(3), end=' ')
      # Note use of 'end' on previous line
      print(repr(x*x*x).rjust(4))
      x=x+1
```

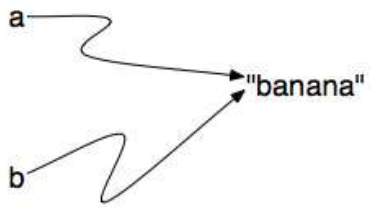
Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [
tel)] on win32
Type "copyright", "credits" or "license()" for more informa
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\intro-pyt
ple.py
 1   1    1
 2   4    8
 3   9   27
 4  16   64
 5  25  125
 6  36  216
 7  49  343
 8  64  512
 9  81  729
10 100 1000
```

Shared References

```
a = "banana"
b = "banana"
```

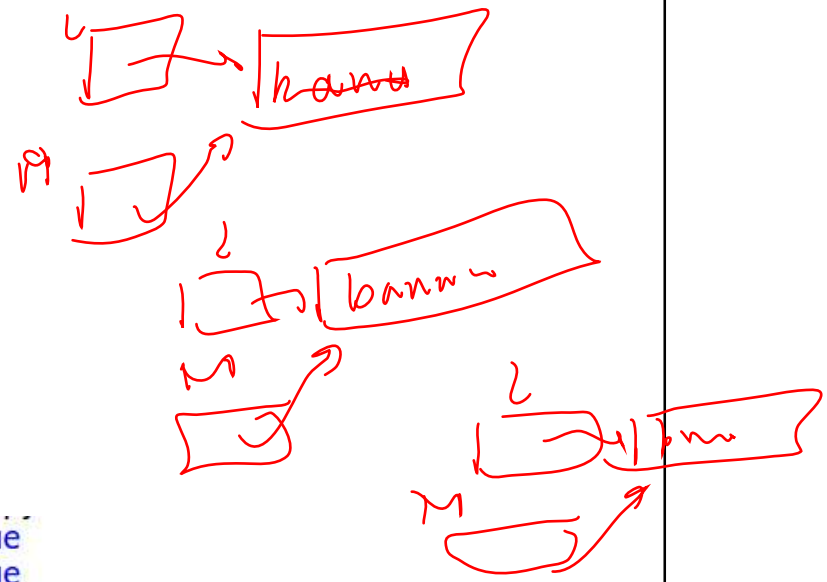


Shared References

simple.py - /home/nowzari/Desktop/python/python

File Edit Format Run Options Window Help

```
L='banana'
M='banana'
print(L==M)
print(L is M)
print(id(L), id(M))    # Same values

L='banana'
M=L
print(L==M)
print(L is M)
print(id(L), id(M))    # Same values

L='banana'
M=L[:]
print(L==M)
print(L is M)
print(id(L), id(M))     # Same values
```



```
True
True
140495238079072 140495238079072
True
True
140495238079072 140495238079072
True
True
140495238079072 140495238079072
```

Parameter passing

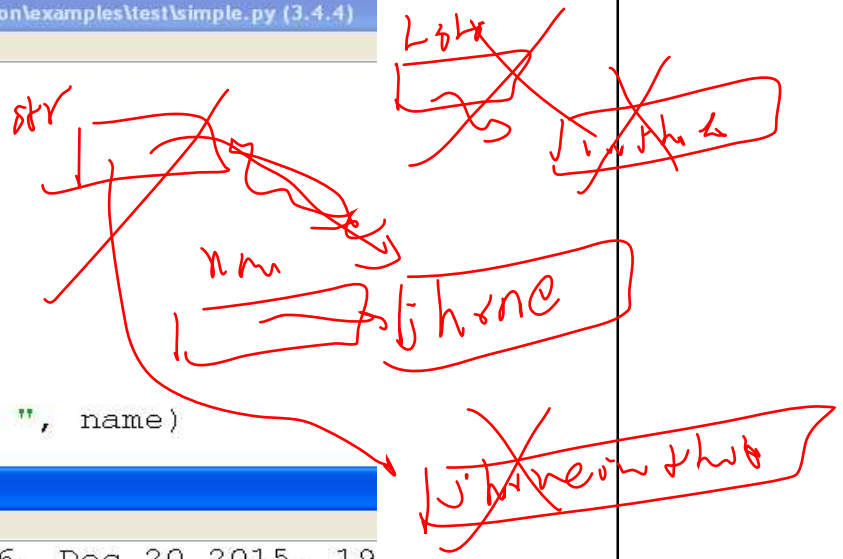
If you pass immutable arguments like integers, strings or tuples to a function, the passing acts like call-by-value. The object reference is passed to the function parameters. They can't be changed within the function, because they can't be changed at all, i.e. they are immutable.

String as a parameter

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
def proc(str):
    lstr=" in the function"
    str = str + lstr
    print (str)
    return


name="jhone"
proc(name)
print("name after function call: ", name)
```



Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19
tel)] on win32
Type "copyright", "credits" or "license()" for mor
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\
ple.py
jhone in the function
name after function call:  jhone
```

Examples

string to number

strNum.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\9b-1\strNum.py (3.4.4)

File Edit Format Run Options Window Help

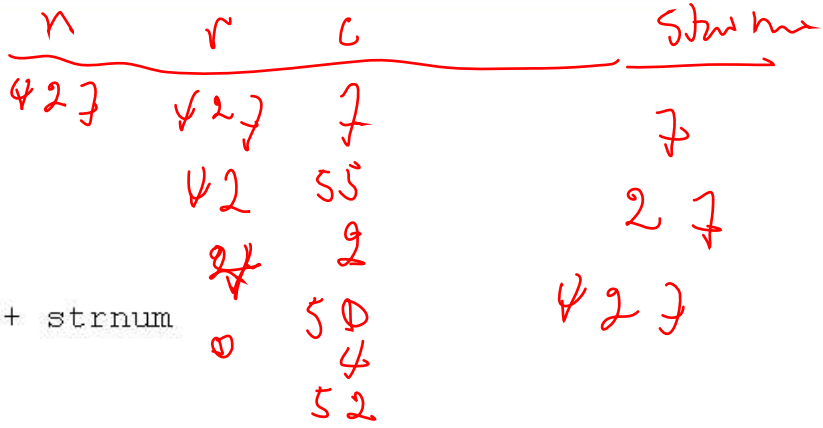
```
def str2num(strnum):
    c, r, index = 0, 0 , 0
    c = ord(strnum[index])
    r = c -48
    index=index + 1
    while (index < len(strnum)):
        c = ord(strnum[index])
        c = c -48
        r = r * 10 + c
        index=index + 1
    return r


nstr = input("Please Enter A Number String :")
num=str2num(nstr)
print("the corresponding number is: " , num)
```

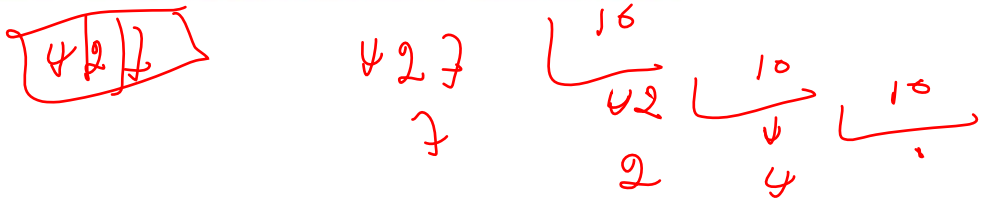
number to string

strNum.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\9b-1\strNum.py (3.4.4)

File Edit Format Run Options Window Help

```
def num2str(n):
    strnum=""
    r=n
    while (r>0):
        c = r % 10
        r = r // 10
        c = c + 48
        strnum = chr(c) + strnum
    return strnum


n = int(input("Please Enter A Number String :"))
nstr=num2str(n)
print("the corresponding string is: " , nstr)
```



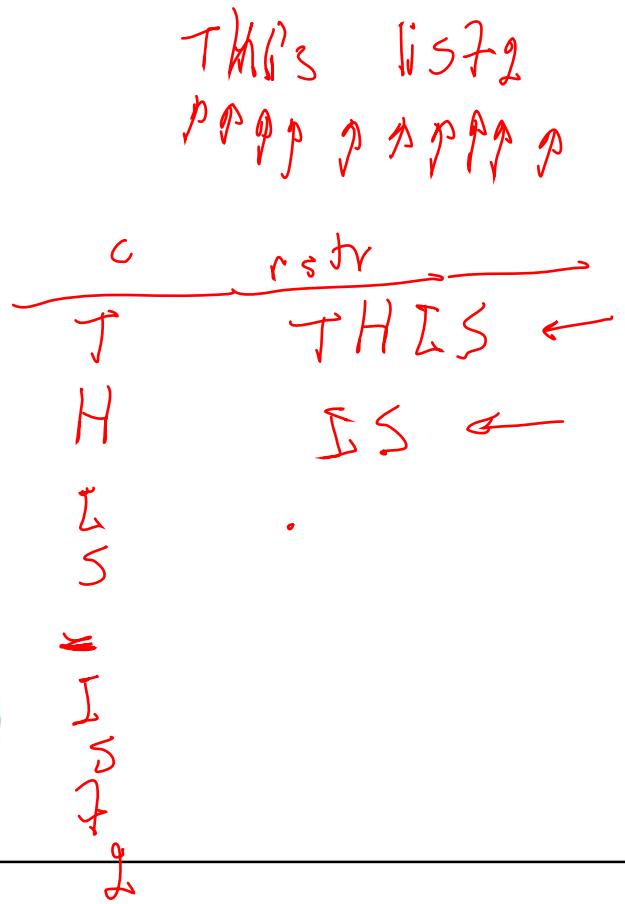


String tokenize

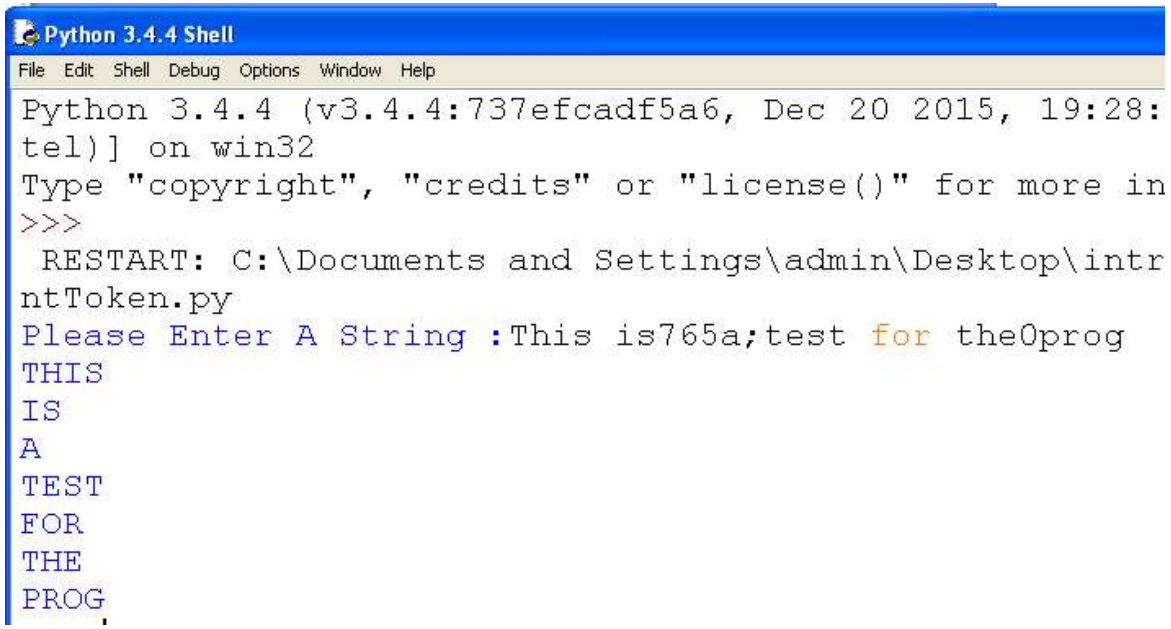
printToken.py - /home/nowzari/Desktop/python/examples/13-strings/printToken.py (3.6.6)

File Edit Format Run Options Window Help

```
def strToken(tstr):
    rStr=""
    counter = 0
    tstr=tstr.upper()
    while (counter < len(tstr)):
        c = tstr[counter]
        if (('A' <= c) and (c <='Z')):
            rStr=rStr + c
        else:
            if(rStr!="") :
                print(rStr)
                rStr=""
        counter=counter + 1
    print(rStr)
    return



nstr = input("Please Enter A String :")
strToken(nstr)
```



String tokenize



btostr.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/15/btostr.py (3.4.4)

File Edit Format Run Options Window Help

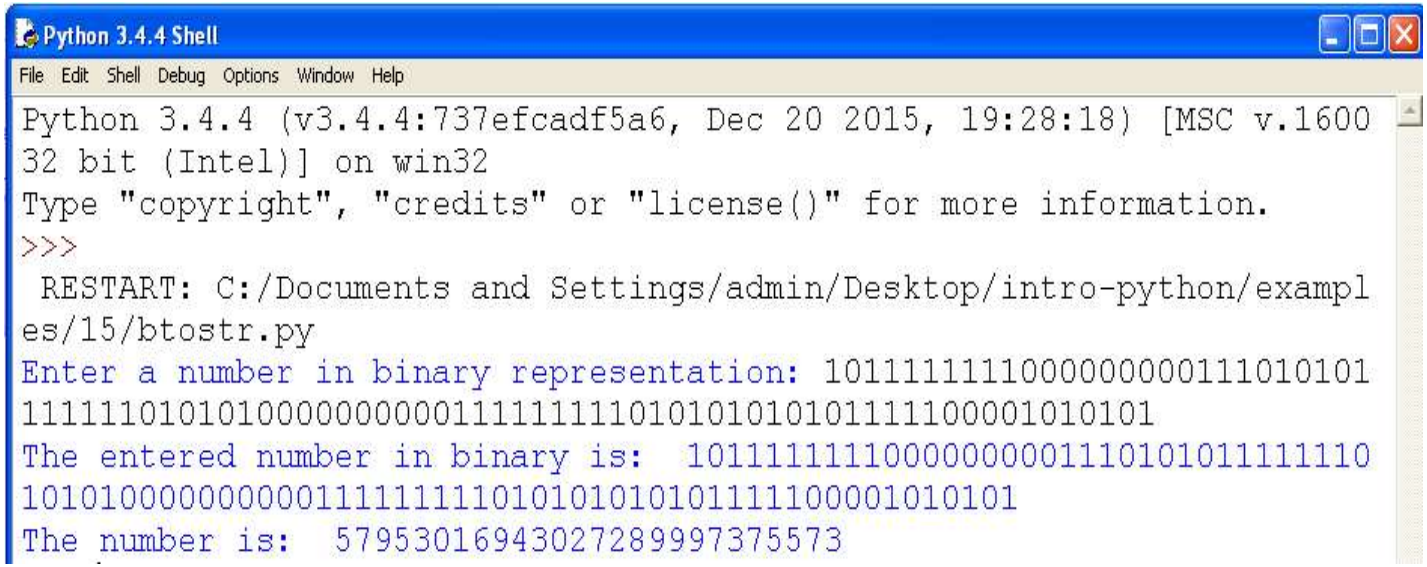
```
# Convert binary digits to integer with ord
def bin2int(B):

    I = 0
    while (B != ''):
        I = I * 2 + (ord(B[0]) - ord('0'))
        B = B[1:]
    return I

num=input("Enter a number in binary representation: ")
print("The entered number in binary is: ", num)
n=bin2int(num)
print("The number is: ", n)
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19
tel)] on win32
Type "copyright", "credits" or "license()" for mor
>>>
 RESTART: C:/Documents and Settings/admin/Desktop/
r.py
Enter a number in binary representation: 1101
The entered number in binary is:  1101
The number is:  13
```

Binary to integer



Password Program

Sounds like you need to consult *Nigerian Security Services, Ltd.* Here is their program for generating random passwords:

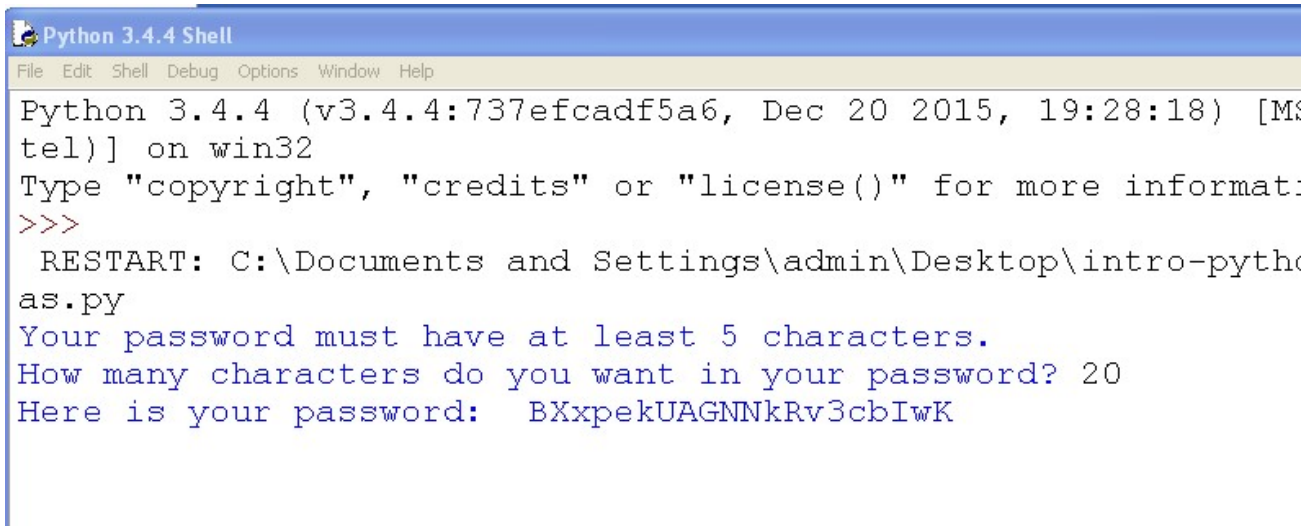
Password

disas.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\disas.py (3.4.4)

File Edit Format Run Options Window Help

```
from random import *

digits = 0;
while ( digits < 5 ) :
    print("Your password must have at least 5 characters.")
    digits = int(input("How many characters do you want in your password? "))

choices = "abcdefghijklmnopqrstuvwxyz"
choices = choices + choices.upper()
choices = choices + "1234567890"

password = "";
j = 0;
while ( j<digits ):
    password = password + choices[randint(0,len(choices)-1)];
    j = j + 1;

print("Here is your password: " , password );
```

Password


Python 3.4.4 Shell
File Edit Shell Debug Options Window Help
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MS
tel)] on win32
Type "copyright", "credits" or "license()" for more informat
>>>
RESTART: C:\Documents and Settings\admin\Desktop\intro-pytho
as.py
Your password must have at least 5 characters.
How many characters do you want in your password? 20
Here is your password: BXxpekUAGNNkRv3cbIwK

Password

randomPass.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\14-strings\randomPass.py (3.4.4)

File Edit Format Run Options Window Help

```
digits = 0;
while ( digits < 5 ) :
    print("Your password must have at least 5 characters.")
    digits = int(input("How many characters do you want in your password? "))

choices = "abcdefghijklmnopqrstuvwxyz"
choices = choices + choices.upper()
choices = choices + "1234567890"

password = "";
j = 0;
while ( j<digits ):
    password = password + choice(choices)
    j = j + 1;

print("Here is your password: " , password );
```

End