Objects & Methods

Friday, 11 February 2022 23:44

1. Objects

+) Everything in Python is an object.
→ Python keeps track of everything as an object with its own memory address
→ Each value is kept as a separate object and that object has a memory address.

+) Recall: Function id() → gives memory address

+) isinstance (object , type) → returns boolean value of whether the object has that type.

```
In [5]: my_string = "Hello"
        print (isinstance(my_string, object))
        print(isinstance(my_string, str))
        print(isinstance(my_string, int))

        True
        True
        False
```

2. Method :

Is a special type of function that is associated with a particular type of object
(things an object can do)

⊛ Calling methods is like calling functions, but 1 minor difference. Since methods are applied to objects, we need to provide the object name with a "." before method name
→ Like importing module Object, and perform different methods.

object_name . method_name (arguments)

Method name	Description
`isalnum()`	Returns True if string is alphanumeric
`isalpha()`	Returns True if string contains only alphabets
`isdigit()`	Returns True if string contains only digits
`isidentifier()`	Return True if string is valid identifier
`islower()`	Returns True if string is in lowercase
`isupper()`	Returns True if string is in uppercase
`isspace()`	Returns True if string contains only whitespace

Strings

Saturday, 12 February 2022 00:03

1. Escape Sequences

Special character called an **escape character** \ (backlash)
→ When used in a string, the character following \ is treated differently from normal.

Escape sequence	Name	Example	Output
\n	newline (ASCII - line feed)	"How\nare\nyou?"	How are you?
\t	tab (ASCII - horizontal tab)	'3\t4\t5'	3 4 5
\\	backslash (\)	'\\'	\
\'	single quote (')	'don\'t'	don't
\"	double quote (")	"He says, \"Hi\"."	He says, "hi".

* e.g.

```
In [9]: str_var = "He yelled \"I WIN\" and jumped with joy."
        print(str_var)
```

```
He yelled "I WIN" and jumped with joy.
```

```
In [10]: escape = "The escape character in Python is \\. Double backslash: \\\\"
         print(escape)
```

```
The escape character in Python is \. Double backslash: \\
```

2. String Operators

Concatenation and Repetition

	Expression	Description	Example	Output
Concatenation	str1 + str2	concatenate str1 and str1	print('ab' + 'c')	abc
Repetition	str1 * int1	concatenate int1 copies of str1	print('a' * 5)	aaaaa
Repetition	int1 * str1	concatenate int1 copies of str1	print(4 * 'bc')	bcbcbcbc

All other mathmatical operators result in a Type Erro

Triple-quoted strings → Can scan multiple lines

```
In [19]: print("""How are you
         doing
         on this fine, cold morning""")
```

```
How are you
doing
on this fine, cold morning
```

Type Conversions

Saturday, 12 February 2022 01:18

+) Convert to str ⇒ Can take any value

+) Convert to int ⇒ Can take float, strings that have int value
→ CANNOT take strings with float value

```
In [27]: print(int('-99'))
-99

In [28]: print(int('99.9'))
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [28], in <module>
----> 1 print(int('99.9'))

ValueError: invalid literal for int() with base 10: '99.9'

In [25]: print(int(99.9)) #notice the difference between this and above
99
```

+) Convert to float ⇒ Can take int, float, strings that have int, float

```
In [21]: print(int(float('99.9')))
99

In [22]: print(float('-43.2'))
-43.2

In [23]: print(float('453'))
453.0
```

☐ Str Indexing and Slicing

Saturday, 12 February 2022 01:21

1. Indexing

An index is a position within the string.
+) Positive indices count from the left-hand side. Starts with 0
+) Negative indices count from the right-hand side.

0	1	2	3	4	5	6	7	8	9	10
I		**L**	**o**	**v**	**e**		**C**	**a**	**t**	**s**
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> print (x[0])        >>> print (x[6])        >>> print (x[-11])
'I'                     ' '                     'I'
                        (space)
```

2. String Slicing

Used to extract more than one character (or substring) using slicing
+) Uses syntax:

[start : finish : step]

start is the index where we start the slice → Default: beginning
finish is the index of one after where we end the slice → Default: end
step is how much we count by between each character → Default: 1

len() → length of a string (= index_final + 1)

```
>>> x[0:6:2]
'ILv'
```

If step < 0 ⇒ Index goes backwards: s[6:2:-1] → s[6] + s[5] + s[4] + s[3] ⇒ No s[2]
(6-1, 6-2)

```
>>> x[::-1]              >>> x[2:6:-1]
'staC evol I'           ''
                        (empty)
```

s[2] + s[2-1] ... but cannot reach s[6] → return empty

NOTE Weird examples.

```
>>> x[6:2]
''
(empty)
```

☐ Modifying Strings

Saturday, 12 February 2022 02:45

1. Modifying Strings

NOTE WE CANNOT MODIFY STRINGS. WE ONLY EXTRACT TO DIFFERENT STRING

→ To modify strings, we extract what we want to keep and create new strings.

```
In [14]: s = "I Love Cats"
         s_new = s[:6] + 'd' + s[6:]
         print(s)
         print(s_new)
```

```
I Love Cats
I Loved Cats
```

or (but we might mistaken)

```
In [13]: s = "I Love Cats"
         s = s[:6] +'d' + s[6:]
         print(s)
```

```
I Loved Cats
```

2. String methods

string_object . upper () → Generates new string that has all characters capitalized.

string_object . lower () → Generates new string that has all characters lowercased.

string_object . find ('substring') → returns first index where a substring is found
returns -1 if no substring exist

string_object . rfind ('substring') → returns last index where a substring is found
returns -1 if no substring exist

```
>>> name = "I'm late! I'm late! For a very important date!"
          0    4          14
>>> print(name.find('late'))
4
>>> print(name.rfind('late'))
14
```

string_object . capitalize () → Capitalize first Character (not letter)

string_object . replace (old, new, count) → Returns a copy of the string in which count occurences of old have been replaced with new
→ count default = all

```
In [22]: str1 = "How much wood would a woodchuck chuck if a woodchuck could chuck wood?"
         str2 = str1.replace("wood", "steel",3)
```

```
In [23]: print(str1)
         print(str2)
```

```
How much wood would a woodchuck chuck if a woodchuck could chuck wood?
How much steel would a steelchuck chuck if a steelchuck could chuck wood?
```