Terminology

- attributes:
- objects and classes:
- dunder: word used for representing double underscores on both the sides of attribute name

In [ ]:

```
country = "India"
type(country)  # `country` is an object of class `str`
```

In [6]:

```
foo = None
type(foo)  # `foo` is an object of class `NoneType`

# what is `foo` and `bar`?
# These are popular variable names used in Python world.
```

Out[6]:

```
NoneType
```

Survival functions

- type
- help
- dir

`help()` function

Help functions returns description (documentation) associated with function

In [7]:

```
country = "India"  # `country` object of class `str`
dir(country)
```

Out[7]:

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmod__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'capitalize',
 'casefold',
 'center',
 'count',
 'encode',
 'endswith',
 'expandtabs',
 'find',
 'format',
 'format_map',
 'index',
 'isalnum',
 'isalpha',
 'isascii',
 'isdecimal',
 'isdigit',
 'isidentifier',
 'islower',
 'isnumeric',
 'isprintable',
 'isspace',
```

```
 'istitle',
 'isupper',
 'join',
 'ljust',
 'lower',
 'lstrip',
 'maketrans',
 'partition',
 'removeprefix',
 'removesuffix',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

Functions of `string` object (total 47)

'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill'

capitalize

- capitalizes first character of a given string

In [14]:

```
country = "india is great"
result = country.capitalize()
result
```

Out[14]:

```
'India is great'
```

In [13]:

```
help(country.capitalize)
```

```
Help on built-in function capitalize:

capitalize() method of builtins.str instance
    Return a capitalized version of the string.

    More specifically, make the first character have upper case and the rest
lower
    case.
```

casefold

- used for converting upper case string into lower case string

In [11]:

```
country = "India is BEST country"
country.casefold()
```

Out[11]:

```
'india is best country'
```

In [15]:

```
help(country.casefold)
```

```
Help on built-in function casefold:

casefold() method of builtins.str instance
    Return a version of the string suitable for caseless comparisons.
```

use case of `casefold`

- caseless comparision

In [18]:

```
country = "India"
motherland = "INDIA"

country.casefold() == motherland.casefold()
```

Out[18]:

```
True
```

Center

- it centers a given string in the middle of given width argument
- it has option to use whitespace character as fill character (by default)

- we can optionally change fill character to any character we want

In [26]:

```
help("".center)
```

```
Help on built-in function center:

center(width, fillchar=' ', /) method of builtins.str instance
    Return a centered string of length width.

    Padding is done using the specified fill character (default is a space).
```

In [25]:

```
# passing single argument as `width`

char = "Introduction"
char.center(80)
```

Out[25]:

```
'                                  Introduction
'
```

In [33]:

```
# passing two arguments. first for `width` and second for `fill character`

char = "introduction"
char.center(80, "$")
```

Out[33]:

```
'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$introduction
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$'
```

In [29]:

```
char = "p"
char.center(4)
```

Out[29]:

```
' p  '
```

count

- returns number of occurrences of input string in a given string

In [34]:

```
help("".count)
```

```
Help on built-in function count:

count(...) method of builtins.str instance
    S.count(sub[, start[, end]]) -> int

    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end].  Optional arguments start and end are
    interpreted as in slice notation.
```

In [43]:

```
sample = """ India is great country.
India is developing country.
India will become new super-power.
power
new-power
"""

sample.count("power")
```

Out[43]:

```
3
```

endswith

Return True if S ends with the specified suffix, False otherwise.

In [48]:

```
player = "virat kohli"
player.endswith("kohli")
```

Out[48]:

```
True
```

In [49]:

```
player = "virat kohli"
player.endswith("Dhoni")
```

Out[49]:

```
False
```

In [51]:

```
country = "india"
country.endswith("b")
```

Out[51]:

```
False
```

In [52]:

```
help("".endswith)
```

```
Help on built-in function endswith:

endswith(...) method of builtins.str instance
    S.endswith(suffix[, start[, end]]) -> bool

    Return True if S ends with the specified suffix, False otherwise.
    With optional start, test S beginning at that position.
    With optional end, stop comparing S at that position.
    suffix can also be a tuple of strings to try.
```

startswith

In [54]:

```
lang = "python is nice"
lang.startswith("python")
```

Out[54]:

```
True
```

In [56]:

```
state = "maharashtra"
state.startswith("maha")
```

Out[56]:

```
True
```

In [57]:

```
river = "godavari"
river.startswith("")
```

Out[57]:

```
True
```

Variables are nothing but just references to an object

In [ ]:

```
foo = "prashant"

    # `foo` is an object of type `str`
    # `foo` is an object of class `str`
    # `foo` is a variable pointing to / referring to object of type `str`
```

In [ ]:

```
- int (1, 2, 3, 4, 5, 6)
- float
- str
- bool
- None
```

`complex` data type in python

real + imaginary

10 + 7i

In [61]:

```
foo = complex(10, 8)
```

In [63]:

```
dir(foo)
```

Out[63]:

```
['__abs__',
 '__add__',
 '__bool__',
 '__class__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__neg__',
 '__new__',
 '__pos__',
 '__pow__',
 '__radd__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmul__',
 '__rpow__',
 '__rsub__',
 '__rtruediv__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__truediv__',
 'conjugate',
 'imag',
 'real']
```

In [64]:

```
foo.real
```

Out[64]:

```
10.0
```

In [66]:

```
foo.imag
```

Out[66]:

```
8.0
```