dunder : double underscores on both sides.

In [ ]:

```
# attributes associated with string

dir("")
```

'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'

In [ ]:

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

In [ ]:

```
foo = "INDIA"

# positive indexing
foo = "INDIA"
      #01234    # positive index


# negative indexing (right to left)
#  I  N  D  I  A
# -5 -4 -3 -2  -1
```

indexing

string :: collection of characters

We have 2 types of indexing available for all data types that are basically a collection

- positive indexing
  - (starts with 0)
  - (starts from left to right)
- negative indexing
  - (starts with -1)
  - (starts from right to left)

index

by default `index` function tells you positive index of any character

In [ ]:

```
lang = "Python"

lang.index("P")
```

In [ ]:

```
feature = "Python is OOP OOP OOP OOP"
          #012345678910

feature.index("OOP")
```

In [ ]:

```
help("".index)
```

In [ ]:

```
 P R A S H A N T  (right to left)
```

`string` functions that starts `is`

- all these functions return boolean value

'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper'

isalnum

returns `True` if all the characters present in given string are alphanumeric

In [ ]:

```
foo = "prashant12345679"
foo.isalnum()
```

In [ ]:

```
foo = "prashantisteacher12345"
foo.isalnum()
```

In [ ]:

```
foo = "prashant@#$"
foo.isalnum()
```

isalpha

- returns True if all chars are alphabets

In [ ]:

```
foo = "velocity"
foo.isalpha()
```

In [ ]:

```
foo = "1234"
foo.isalpha()
```

isascii

- returns True if all chars are ascii

isdecimal

- returns True if chars are decimal

In [ ]:

```
foo = "10"
foo.isdecimal()
```

isdigit

returns True if chars are digits

In [ ]:

```
foo = "1234.24563"
foo.isdigit()
```

isidentifier

- variable name
- function name
- class name
- module name
- keyword name

All these things are collectively called as identifier

In [ ]:

```
help("".isidentifier)
```

In [ ]:

```
help("keywords")
```

In [ ]:

```
"prashant".isidentifier()
```

In [ ]:

```
foo = 100

"foo".isidentifier()
```

In [ ]:

```
"print()".isidentifier()
```

In [ ]:

```
"def".isidentifier()
```

Rules variable names

- variable name should not start with numerical
- variable name cannot be a keyword
- variable name should not contain special characters (except for underscore)
- variable names cannot contain space
- varibable names are case sensitive
- in short, variable names can only contain alpha-numeric chars. you can use _ (underscore) as seperator
- we should not use python built in functions as variable names (as a matter of convention)

In [ ]:

```
account_details = "1 Lakh rupees"

ACCOUNT_DETAILS
```

In [ ]:

```
foo_prashant = "prashant"
```

In [ ]:

```
True = "prashant"
```

In [ ]:

```

```

In [ ]:

```
"12foo".isidentifier()
```

In [ ]:

```
foo = ""
foo.isidentifier()
```

In [ ]:

```
"".isidentifier()
```

In [ ]:

```
"foo".isidentifier()
```

In [ ]:

```
"prashant".istitle()
```

In [ ]:

```
"Prashant".istitle()`
```

In [ ]:

```
_foo = "prash "
_foo
```

encode

- ascii, utf-8, utf-16

In [ ]:

```
help("".encode)
```

what is default encoding in Python3?

- utf-8

In [ ]:

```
foo = "my %%!name $#$% is prashant ҷҷ ҷҷҷҷҷҷҷ"
foo.encode()   # whenever a string has b prefix, we call it `bytestring`
```

expandtabs

- \t : special character used to represent tab (by default it is 8 space)

In [ ]:

```
help("".expandtabs)
```

In [ ]:

```
foo = "prashant\tvelocity"
result = foo.expandtabs(20)
print(result)
```

split & join

split

- splits existing string using a delimiter character
- argument that we pass to split function is a delimiter character
- `split` function returns a `list` object

In [ ]:

```
foo = "prashant velocity python"
foo.split(" ")
```

In [ ]:

```
bar = "virat,hardik,chahal,rahul,sachin,dhoni"
bar.split(",")
```

In [ ]:

```
states = "MH;KA;PY;PN;RS"
result = states.split(";")
result
```

In [ ]:

```
cities = "mumbai####delhi####chennai####pune"
city_list = cities.split("####")
city_list
```

join

- takes a list as input
- you invoke a join function on a string object (that is delimiter)

In [ ]:

```
city_list

"-".join(city_list)
```

In [14]:

```
languages = ["python", "java", "golang", "javascript"]
result = " \t     ".join(languages)
print(result)
```

```
python       java             golang           javascript
```

In [16]:

```
players = ["sachin", "dhoni", "dravid", "hardik"]
print(type(players))

" & ".join(players)
```

```
<class 'list'>
```

Out[16]:

```
'sachin & dhoni & dravid & hardik'
```

In [17]:

```
"ª".isdecimal()
```

Out[17]:

```
False
```

In [19]:

```
b'1234'.isdigit()
```

Out[19]:

```
True
```

Join

- we invoke join function on a string object (that will be referred as `delimiter`)

In [2]:

```
"-----".join(["prashant", "velocity"])
```

Out[2]:

```
'prashant-----velocity'
```

Split

- default delimiter is a "space" character

In [3]:

```
statement = "India is great"
statement.split()
```

Out[3]:

```
['India', 'is', 'great']
```