In [1]:

```
foo = "prashant"
type(foo)

# `foo` is a variable
# `foo` is an object of class `str`
```

Out[1]:

```
str
```

'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 [100]:

```
para = """OOP & procedure oriented
         #01234
easy to code
easy to read
robust lib support
open source
dynamically typed lang
extensible
portable
supports GUI
interpreted
"""

para.find("&")
```

Out[100]:

```
4
```

In [101]:

```
para = """
OOP & procedure oriented
12345
easy to code
easy to read
robust lib support
open source
dynamically typed lang
extensible
portable
supports GUI
interpreted
"""

para.find("&")
```

Out[101]:

```
5
```

In [ ]:

```
## function call
## function invoking

<function-name>(<arguments>)
```

format

- function is used to pass dynamic values in a given string
- basically you add placeholders to add values dynamically at later stage
- placeholder can be added with {} and you can specifiy numbers

In [97]:

```
student = "{0} has scored {1} in today's exam"

result = student.format("hardik", "300")
print(result)


# prashant = student.format("prashant", 90)
# virat = student.format("virat", 100)
# print(prashant)
# print(virat)
```

```
hardik has scored 300 in today's exam
```

In [ ]:

```

```

In [31]:

```
help("".format)
```

```
Help on built-in function format:

format(...) method of builtins.str instance
    S.format(*args, **kwargs) -> str

    Return a formatted version of S, using substitutions from args and kwarg
s.
    The substitutions are identified by braces ('{' and '}').
```

In [25]:

```
example = "python has features {0}, {1}, {2}".format("OOP", "interpreted", "portable")
example
```

Out[25]:

```
'python has features OOP, interpreted, portable'
```

In [24]:

```
statement = "{country} is great".format(country="india")
statement
```

Out[24]:

```
'india is great'
```

In [23]:

```
details_of_prashant = "{profession} - {skill} - {insti}".format(profession="engineer", skil
details_of_prashant
```

Out[23]:

```
'engineer - python - velocity'
```

In [30]:

```
magic_string = "{0}, {1}"
result = magic_string.format("prashant", "virat")
result
```

Out[30]:

```
'prashant, virat'
```

In [58]:

```
match_score = """
{0} team has won the match. Man of the match is {1}.
Today match was played in Eden Garden between so and soo....
"""
```

In [59]:

```
result = match_score.format("Indian", "Virat")
print(result)
```

```
Indian team has won the match. Man of the match is Virat.
Today match was played in Eden Garden between so and soo....
```

In [43]:

```
match_score = "{0} team has won the match. Man of the match is {1}. Today match was played
match_score.format("Austrelia", "XYZ")
```

Out[43]:

```
'Austrelia team has won the match. Man of the match is XYZ. Today match was
played in Eden Garden between so and soo....'
```

Indexing in Python

- Python indexing always starts with 0
- So when I say string is collection of characters

Example

```
"prashant"
 01234567
```

In [98]:

```
name = "prashant"
       #01234567

name.find("shant")
```

Out[98]:

```
3
```

In [99]:

```
feature = "python is interpreted"
          #0123456789
feature.find("is")
```

Out[99]:

```
7
```