# The amateur's guide to explore machine learning

***Release 1.0.0***

**Nishant Baheti**

**Aug 19, 2022**

# CONTENTS:

CHAPTER

# ONE

# GENERATORS

## 1.1 topic 1

```
Iterable - __iter__ or __getitem__ ...
Iterator - __next__ ...
Iteration - process to iterate over iterable
```

[1]:
```
l = [1,2,4,5]
```

[7]:
```
for item in dir(l):
    if item in ['__iter__','__getitem__']:
        print(item)
```

```
__getitem__
__iter__
```

[8]:
```
for i in l:
    print(i)
```

```
1
2
4
5
```

[13]:
```
def custom_gen(n):
    for i in range(n):
        yield i

for item in dir(custom_gen(10)):
    if item in ['__next__']:
        print(item)
```

```
__next__
```

[23]:
```
g = custom_gen(5)
```

[24]:
```
next(g)
```

[24]:
```
0
```

[25]:
```
next(g)
```

[25]:
```
1
```

[26]:
```
next(g)
```

[26]:
```
2
```

[28]:
```
g = custom_gen(5)
for item in g:
    print(item)
```

```
0
1
2
3
4
```

[29]:
```
n = "Nishant"
for item in n:
    print(item)
```

```
N
i
s
h
a
n
t
```

[32]:
```
iterable = iter(n)
```

[34]:
```
for item in dir(iterable):
    if item in ['__next__']:
        print(item)
```

```
__next__
```

[35]:
```
next(iterable)
```

[35]:
```
'N'
```

[36]:
```
next(iterable)
```

[36]:
```
'i'
```

[37]:
```
next(iterable)
```

[37]:
```
's'
```

[ ]:
```
```

CHAPTER

# TWO

# ITERTOOLS

[1]:
```
from itertools import *
```

[2]:
```
x = [1,2,3,4,5,6,7]
```

[3]:
```
l = []
for iter in combinations_with_replacement(x,2):
    l.append([ iter[0], iter[1], iter[0]-iter[1] ])
l
```

[3]:
```
[[1, 1, 0],
 [1, 2, -1],
 [1, 3, -2],
 [1, 4, -3],
 [1, 5, -4],
 [1, 6, -5],
 [1, 7, -6],
 [2, 2, 0],
 [2, 3, -1],
 [2, 4, -2],
 [2, 5, -3],
 [2, 6, -4],
 [2, 7, -5],
 [3, 3, 0],
 [3, 4, -1],
 [3, 5, -2],
 [3, 6, -3],
 [3, 7, -4],
 [4, 4, 0],
 [4, 5, -1],
 [4, 6, -2],
 [4, 7, -3],
 [5, 5, 0],
 [5, 6, -1],
 [5, 7, -2],
 [6, 6, 0],
 [6, 7, -1],
 [7, 7, 0]]
```

[4]:
```
l = []
for iter in combinations(x,2):
```

(continues on next page)

(continued from previous page)

```
    l.append([ iter[0], iter[1], iter[0]-iter[1] ])
l
```

[4]:
```
[[1, 2, -1],
 [1, 3, -2],
 [1, 4, -3],
 [1, 5, -4],
 [1, 6, -5],
 [1, 7, -6],
 [2, 3, -1],
 [2, 4, -2],
 [2, 5, -3],
 [2, 6, -4],
 [2, 7, -5],
 [3, 4, -1],
 [3, 5, -2],
 [3, 6, -3],
 [3, 7, -4],
 [4, 5, -1],
 [4, 6, -2],
 [4, 7, -3],
 [5, 6, -1],
 [5, 7, -2],
 [6, 7, -1]]
```

[5]:
```
l = []
for iter in permutations(x,2):
    l.append([ iter[0], iter[1], iter[0]-iter[1] ])
l
```

[5]:
```
[[1, 2, -1],
 [1, 3, -2],
 [1, 4, -3],
 [1, 5, -4],
 [1, 6, -5],
 [1, 7, -6],
 [2, 1, 1],
 [2, 3, -1],
 [2, 4, -2],
 [2, 5, -3],
 [2, 6, -4],
 [2, 7, -5],
 [3, 1, 2],
 [3, 2, 1],
 [3, 4, -1],
 [3, 5, -2],
 [3, 6, -3],
 [3, 7, -4],
 [4, 1, 3],
 [4, 2, 2],
 [4, 3, 1],
 [4, 5, -1],
 [4, 6, -2],
```

(continues on next page)

(continued from previous page)

```
 [4, 7, -3],
 [5, 1, 4],
 [5, 2, 3],
 [5, 3, 2],
 [5, 4, 1],
 [5, 6, -1],
 [5, 7, -2],
 [6, 1, 5],
 [6, 2, 4],
 [6, 3, 3],
 [6, 4, 2],
 [6, 5, 1],
 [6, 7, -1],
 [7, 1, 6],
 [7, 2, 5],
 [7, 3, 4],
 [7, 4, 3],
 [7, 5, 2],
 [7, 6, 1]]
```

[6]:
```
l = [1,2,3,4,5,6,7]
# l = [7,6,5,4,3,2,1]
# l = [1,2,1,2,3,2,1]
# l = [2,3,5,6,7,1,2,4,6,9,7,5]
# l = [0,0,0,1,0,0,0,0,1,0]

max = 0
start_new = True
temp_list = []
for i in range(len(l)-1):
    if start_new:
        total_sum = sum(temp_list)
        if max < total_sum:
            max = total_sum
        temp_list = [l[i]]

    if l[i] < l[i+1]:
        temp_list.append(l[i+1])
        print(temp_list)
        total_sum = sum(temp_list)
        if max < total_sum:
            max = total_sum
        start_new = False

    else:
       start_new = True

print(max)
```

```
[1, 2]
[1, 2, 3]
[1, 2, 3, 4]
```

(continues on next page)

(continued from previous page)

```
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6, 7]
28
```

question 2 :

You will be given a list of stock prices for a given day and your goal is to return the maximum profit that could have been made by buying a stock at the given price and then selling the stock later on. For example if the input is: `[45, 24, 35, 31, 40, 38, 11]` then your program should return 16 because if you bought the stock at `$24` and sold it at `$40`, a profit of $16 was made and this is the largest profit that could be made. If no profit could have been made, return -1.

[7]:
```
# l = [1,2,3,4,5,6,7]
l = [45, 24, 35, 31, 40, 38, 11]
# l = [100, 180, 260, 310, 40, 535, 695]
# l = [7,6,5,4,3,2,1]
selling_combs = []
for iter in combinations(l,2):
    selling_combs.append([iter[1]-iter[0]])

selling_combs.sort(reverse=True)
print("Max profit value of the week :",selling_combs[0])
```

```
Max profit value of the week : [16]
```

[15]:
```
for i in permutations(range(1,5),2):
    print(i)
```

```
(1, 2)
(1, 3)
(1, 4)
(2, 1)
(2, 3)
(2, 4)
(3, 1)
(3, 2)
(3, 4)
(4, 1)
(4, 2)
(4, 3)
```

CHAPTER

# THREE

# LIST COMPREHENSIONS

## 3.1 AAM ZINDGI

[1]:
```
l = []

for i in range(10):
    if i%3 == 0:
        l.append(i)
l
```

[1]:
```
[0, 3, 6, 9]
```

## 3.2 MENTOSS ZINDGI

[2]:
```
[i for i in range(10) if i%3 == 0]
```

[2]:
```
[0, 3, 6, 9]
```

[3]:
```
[i if i%3 == 0 else i%3 for i in range(10)]
```

[3]:
```
[0, 1, 2, 3, 1, 2, 6, 1, 2, 9]
```

CHAPTER

# FOUR

# DICTIONARY COMPREHENSION

[4]:
```
d = {item:f"ITEM-{item}" for item in range(10)}
d
```

[4]:
```
{0: 'ITEM-0',
 1: 'ITEM-1',
 2: 'ITEM-2',
 3: 'ITEM-3',
 4: 'ITEM-4',
 5: 'ITEM-5',
 6: 'ITEM-6',
 7: 'ITEM-7',
 8: 'ITEM-8',
 9: 'ITEM-9'}
```

[5]:
```
{value:key for key,value in d.items()}
```

[5]:
```
{'ITEM-0': 0,
 'ITEM-1': 1,
 'ITEM-2': 2,
 'ITEM-3': 3,
 'ITEM-4': 4,
 'ITEM-5': 5,
 'ITEM-6': 6,
 'ITEM-7': 7,
 'ITEM-8': 8,
 'ITEM-9': 9}
```

CHAPTER

# FIVE

# SET COMPREHENSIONS

[6]:
```
{item for item in ["nsihatn","nsihatjn","nishatn","nishant","nishant","nishant","nishant
↪"]}
```

[6]:
```
{'nishant', 'nishatn', 'nsihatjn', 'nsihatn'}
```

CHAPTER

# SIX

# LIST OPERATIONS

[1]:
```
number_of_inputs = int(input())


max = 0
while number_of_inputs > 0:
    i = float(input())
    if i > max:
        max = i
    number_of_inputs -= 1
print(max)


# list_of_inputs = []
# while number_of_inputs > 0:
#     i = float(input())
#     list_of_inputs.append(i)
#     number_of_inputs -= 1

# list_of_inputs.sort(reverse=True)
# print(list_of_inputs[0])
```

```
55.0
```

[2]:
```
l = [1,2,3,2,3,6]
l.append(7)
l
```

[2]:
```
[1, 2, 3, 2, 3, 6, 7]
```

[3]:
```
l.count(3)
```

[3]:
```
2
```

[4]:
```
l.extend([7,8,7])
l
```

[4]:
```
[1, 2, 3, 2, 3, 6, 7, 7, 8, 7]
```

[5]:
```
l.index(7) # first occurance index of the specified value
```

[5]:
```
6
```

[6]:
```
l.insert(1,10) # position, element
l
```

[6]:
```
[1, 10, 2, 3, 2, 3, 6, 7, 7, 8, 7]
```

[7]:
```
l.pop(1) # index remove element
l
```

[7]:
```
[1, 2, 3, 2, 3, 6, 7, 7, 8, 7]
```

[8]:
```
l.remove(3) # removes first occurance of specified element
l
```

[8]:
```
[1, 2, 2, 3, 6, 7, 7, 8, 7]
```

[9]:
```
l.reverse() # reverse the list
l
```

[9]:
```
[7, 8, 7, 7, 6, 3, 2, 2, 1]
```

[10]:
```
l.sort() # sort the list
```

[11]:
```
l.clear() # clean the list
```

[12]:
```
l
```

[12]:
```
[]
```

[15]:
```
l = list("Hello I am nishant".replace(" ",""))
```

[15]:
```
['H', 'e', 'l', 'l', 'o', 'I', 'a', 'm', 'n', 'i', 's', 'h', 'a', 'n', 't']
```

CHAPTER

SEVEN

# DECORATORS

[13]:
```
def uppercase_decorator(function):
    def wrapper(*args,**kwargs):
        func = function(*args,**kwargs)
        make_uppercase = func.upper()
        return make_uppercase
    return wrapper
```

[14]:
```
def func():
    return "hello world"
```

[15]:
```
func = uppercase_decorator(func)
```

[16]:
```
func()
```

[16]:
```
'HELLO WORLD'
```

[20]:
```
@uppercase_decorator
def func1():
    return "hello world 1"
```

[21]:
```
func1()
```

[21]:
```
'HELLO WORLD 1'
```

## 7.1 timer example

[24]:
```
from time import time,sleep

def timeit(f):
    def wrapper(*args,**kwargs):
        start = time()
        val = f(*args,**kwargs)
        end = time()

        print("total time taken for this function :",f.__name__," is",end-start)
        return val
    return wrapper
```

(continues on next page)

(continued from previous page)

```
@timeit
def slow_operation():
    sleep(3)
    return "done"
```

[25]:
```
slow_operation()
```

```
total time taken for this function : slow_operation  is 3.0030782222747803
```

[25]:
```
'done'
```

CHAPTER

# EIGHT

# MAP FILTER

[1]:
```
l = ['1','2','3','4','5','6']
```

[2]:
```
list(map(int,l))
```

[2]:
```
[1, 2, 3, 4, 5, 6]
```

[3]:
```
list(map(lambda x:x**2,map(int,l)))
```

[3]:
```
[1, 4, 9, 16, 25, 36]
```

[19]:
```
def sq(x):
    return x**2

def cube(x):
    return x**3

func = [sq, cube]
l1 = list(map(int,l))

l1
```

[19]:
```
[1, 2, 3, 4, 5, 6]
```

[22]:
```
for i in l1:
    val = list(map(lambda x:x(i),func))
    print(val)
```

```
[1, 1]
[4, 8]
[9, 27]
[16, 64]
[25, 125]
[36, 216]
```

[26]:
```
d = [{
    "x" : 1,
    "y" : 2
},{
    "x" : 1,
    "y" : 4
```

(continues on next page)

(continued from previous page)

```
},{
    "x" : 2,
    "y" : 1
}]

def filter_x_1(a):
    return a["x"] == 1

list(filter(filter_x_1,d))
```

[26]:
```
[{'x': 1, 'y': 2}, {'x': 1, 'y': 4}]
```

[28]:
```
from functools import reduce


a = [1,2,4,5,6,7]

mul = reduce(lambda x,y: x+y, a)
mul
```

[28]:
```
25
```

[ ]:
```

```

CHAPTER

# NINE

# FOR ELSE OPERATION

[2]:
```
l = "hello My Name is Nishant Baheti".split(" ")
l
```

[2]:
```
['hello', 'My', 'Name', 'is', 'Nishant', 'Baheti']
```

[3]:
```
for item in l:
    print(item)
else:
    print("this loop ran successfully")
```

```
hello
My
Name
is
Nishant
Baheti
this loop ran successfully
```

- if there is a break then else will not run

[4]:
```
for item in l:
    print(item)
    break
else:
    print("this loop ran successfully")
```

```
hello
```

[6]:
```
for item in l:
    if item == "Nishant":
        print("Nishant found")
        break
else:
    print("Nishant was not found")
```

```
Nishant found
```

CHAPTER

# TEN

# BASIC OOPS AND ADVANCED CONCEPTS

[1]:
```
class Employee:
    number_of_leaves = 8

    def  __init__(self,name,salary,role):
        self.name = name
        self.salary = salary
        self.role = role

    def print_details(self):

        print(f"""
        NAME     :{self.name}
        SALARY   :{self.salary}
        ROLE     :{self.role}
        """)

    @classmethod
    def change_leaves(cls,new_leaves):
        cls.number_of_leaves = new_leaves

    @classmethod
    def instance_with_dash(cls,params_dash):
        return cls(*params_dash.split("-"))

    @staticmethod
    def print_good_string(string):
        print(f"this is a good string : {string}")



e1 = Employee("E1",45000,"t1")
e2 = Employee.instance_with_dash("E2-50000-t2")

e1.print_details()
e2.print_details()

print(e1.number_of_leaves, e2.number_of_leaves)

e1.number_of_leaves = 10
```

(continues on next page)

(continued from previous page)

```
print(e1.number_of_leaves)

Employee.number_of_leaves = 20

print(e1.number_of_leaves, e2.number_of_leaves)

e2.number_of_leaves = 10
print(Employee.number_of_leaves, e2.number_of_leaves)

e1.print_good_string("good1")
Employee.print_good_string("good2")
```

```
        NAME     :E1
        SALARY   :45000
        ROLE     :t1


        NAME     :E2
        SALARY   :50000
        ROLE     :t2

8 8
10
10 20
20 10
this is a good string : good1
this is a good string : good2
```

## 10.1 Single Inheritance

[2]:
```
class Player:
    def __init__(self,name,sports):
        self.name = name
        self.sports = sports

    def print_details(self):
        print(f"""
        NAME     :{self.name}
        SPORTS   :{self.sports}
        """)
```

### 10.1.1 without super()

[3]:
```
class Programmer(Employee,Player):

    language ="C++"

    def print_language(self):
        print(self.language)

p1 = Programmer("e1",35000,"P1")
p1.print_details()
p1.print_language()
```

```
        NAME     :e1
        SALARY   :35000
        ROLE     :P1

C++
```

[4]:
```
# sequence matters

## this will throw error
class Programmer(Player,Employee):

    language ="C++"

    def print_language(self):
        print(self.language)

p1 = Programmer("e1",35000,"P1")
p1.print_details()
p1.print_language()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-a25debaa34a6> in <module>
      9         print(self.language)
     10
---> 11 p1 = Programmer("e1",35000,"P1")
     12 p1.print_details()
     13 p1.print_language()

TypeError: __init__() takes 3 positional arguments but 4 were given
```

[5]:
```
class Programmer(Player,Employee):

    language ="C++"

    def print_language(self):
        print(self.language)

p1 = Programmer("e1",["tennis","cricket"])
```

(continues on next page)

(continued from previous page)

```
p1.print_details()
p1.print_language()
```

```
        NAME     :e1
        SPORTS   :['tennis', 'cricket']

C++
```

## 10.2 Heirarchy in descending order

instance variable child > child class variable > parent instance variable > parent class variable

[6]:
```
class A:
    classvar1 = "this is a class vairable in class A"

    def __init__(self):
        self.classvar1 = "this is a instance variable in class A"

class B(A):
    classvar1 = "this is a class variable in class B"

    def __init__(self):
        self.classvar1 = "this is a instance variable in class B"



b = B()

b.classvar1
```

[6]:
```
'this is a instance variable in class B'
```

[7]:
```
class A:
    classvar1 = "this is a class vairable in class A"

    def __init__(self):
        self.classvar1 = "this is a instance variable in class A"

class B(A):
    classvar1 = "this is a class variable in class B"

#     def __init__(self):
#         self.classvar1 = "this is a instance variable in class B"



b = B()
```

(continues on next page)

(continued from previous page)

```
b.classvar1
```

[7]:
```
'this is a instance variable in class A'
```

[8]:
```
class A:
    classvar1 = "this is a class vairable in class A"

#     def __init__(self):
#         self.classvar1 = "this is a instance variable in class A"

class B(A):
    classvar1 = "this is a class variable in class B"

#     def __init__(self):
#         self.classvar1 = "this is a instance variable in class B"



b = B()

b.classvar1
```

[8]:
```
'this is a class variable in class B'
```

[9]:
```
class A:
    classvar1 = "this is a class vairable in class A"

#     def __init__(self):
#         self.classvar1 = "this is a instance variable in class A"

class B(A):
    pass
#     classvar1 = "this is a class variable in class B"

#     def __init__(self):
#         self.classvar1 = "this is a instance variable in class B"



b = B()

b.classvar1
```

[9]:
```
'this is a class vairable in class A'
```

## 10.3 super()

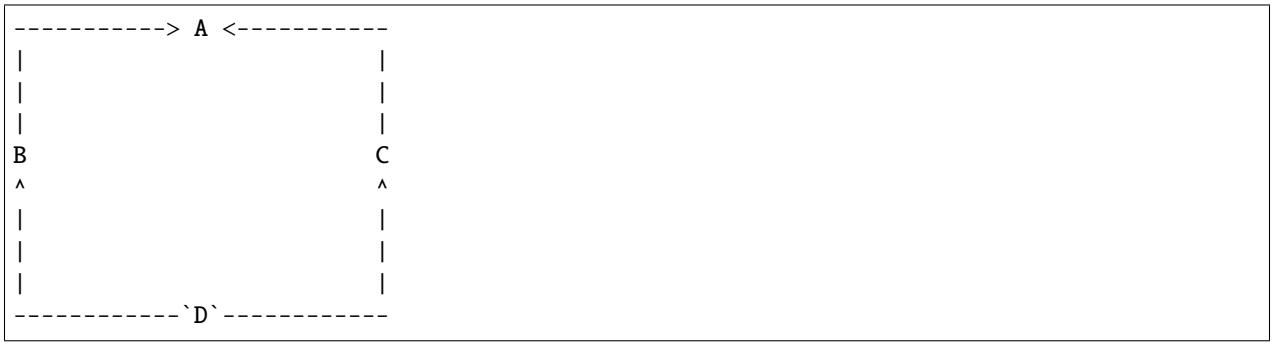
[10]:
```
class A:
    classvar1 = "this is a class vairable in class A"

    def __init__(self):
        self.var1 = "this is a instance variable in class A"

class B(A):
    classvar1 = "this is a class variable in class B"

    def __init__(self):
        super().__init__()
        self.var1 = "this is a instance variable in class B" ## overridden by child



b = B()

b.var1
```

[10]:
```
'this is a instance variable in class B'
```

[11]:
```
class A:
    classvar1 = "this is a class vairable in class A"

    def __init__(self):
        self.var1 = "this is a instance variable in class A"

class B(A):
    classvar1 = "this is a class variable in class B"

    def __init__(self):
        self.var1 = "this is a instance variable in class B"
        super().__init__() ## overridden by parent


b = B()

b.var1
```

[11]:
```
'this is a instance variable in class A'
```

## 10.4 Diamond shape problem




[12]:
```
class A:
    def func(self):
        print("CLASS A")
class B(A):
    def func(self):
        print("CLASS B")
class C(A):
    def func(self):
        print("CLASS C")
class D(B,C):
    pass

a = A()
b = B()
c = C()
d = D()

d.func()
```

```
CLASS B
```

[13]:
```

class A:
    def func(self):
        print("CLASS A")
class B(A):
    def func(self):
        print("CLASS B")
class C(A):
    def func(self):
        print("CLASS C")
class D(C,B):
    pass

a = A()
b = B()
c = C()
d = D()

d.func()
```

```
CLASS C
```

## 10.5 Operator overloading and dunder methods(underscore methods)

https://docs.python.org/3/library/operator.html

[14]:
```
class Employee:
    number_of_leaves = 8

    def  __init__(self,name,salary,role):
        self.name = name
        self.salary = salary
        self.role = role

    def print_details(self):

        print(f"""
        NAME     :{self.name}
        SALARY   :{self.salary}
        ROLE     :{self.role}
        """)

    @classmethod
    def change_leaves(cls,new_leaves):
        cls.number_of_leaves = new_leaves

    @classmethod
    def instance_with_dash(cls,params_dash):
        return cls(*params_dash.split("-"))

    @staticmethod
    def print_good_string(string):
        print(f"this is a good string : {string}")

e1 = Employee("E1",45000,"SDE1")
e2 = Employee("E2",40000,"SDE1")

print(e1+e2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-42bd4dc47e1e> in <module>
     30 e2 = Employee("E2",40000,"SDE1")
     31
---> 32 print(e1+e2)

TypeError: unsupported operand type(s) for +: 'Employee' and 'Employee'
```

[15]:
```
class Employee:
    number_of_leaves = 8

    def  __init__(self,name,salary,role):
        self.name = name
        self.salary = salary
        self.role = role

    def print_details(self):

        print(f"""
        NAME     :{self.name}
        SALARY   :{self.salary}
        ROLE     :{self.role}
        """)

    @classmethod
    def change_leaves(cls,new_leaves):
        cls.number_of_leaves = new_leaves

    @classmethod
    def instance_with_dash(cls,params_dash):
        return cls(*params_dash.split("-"))

    @staticmethod
    def print_good_string(string):
        print(f"this is a good string : {string}")

    ## create a dunder method to work on add operation

    def __add__(self,other):

        return self.salary + other.salary

    def __truediv__(self,other):

        return self.salary / other.salary

e1 = Employee("E1",45000,"SDE1")
e2 = Employee("E2",40000,"SDE1")

print(e1+e2)
print(e1/e2)
```

```
85000
1.125
```

[16]:
```
dir(Employee)
```

[16]:
```
['__add__',
 '__class__',
 '__delattr__',
 '__dict__',
```

(continues on next page)

(continued from previous page)

```
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__truediv__',
 '__weakref__',
 'change_leaves',
 'instance_with_dash',
 'number_of_leaves',
 'print_details',
 'print_good_string']
```

[17]:
```
e1 ## here it is showing default
```

[17]:
```
<__main__.Employee at 0x7f747cba7370>
```

[18]:
```
class Employee:
    number_of_leaves = 8

    def  __init__(self,name,salary,role):
        self.name = name
        self.salary = salary
        self.role = role

    def print_details(self):

        print(f"""
        NAME     :{self.name}
        SALARY   :{self.salary}
        ROLE     :{self.role}
        """)

    @classmethod
    def change_leaves(cls,new_leaves):
```

(continues on next page)

(continued from previous page)

```
        cls.number_of_leaves = new_leaves

    @classmethod
    def instance_with_dash(cls,params_dash):
        return cls(*params_dash.split("-"))

    @staticmethod
    def print_good_string(string):
        print(f"this is a good string : {string}")

    ## create a dunder method to work on add operation

    def __add__(self,other):

        return self.salary + other.salary

    def __truediv__(self,other):

        return self.salary / other.salary

    def __repr__(self):
        return f"""Employee('{self.name}',{self.salary},'{self.role}')"""

e1 = Employee("E1",45000,"SDE1")
e2 = Employee("E2",40000,"SDE1")

print(e1)
print(repr(e1)) ##overridden repr
```

```
Employee('E1',45000,'SDE1')
Employee('E1',45000,'SDE1')
```

[19]:
```
class Employee:
    number_of_leaves = 8

    def  __init__(self,name,salary,role):
        self.name = name
        self.salary = salary
        self.role = role

    def print_details(self):

        print(f"""
        NAME     :{self.name}
        SALARY   :{self.salary}
        ROLE     :{self.role}
        """)

    @classmethod
    def change_leaves(cls,new_leaves):
        cls.number_of_leaves = new_leaves
```

(continues on next page)

(continued from previous page)

```
    @classmethod
    def instance_with_dash(cls,params_dash):
        return cls(*params_dash.split("-"))

    @staticmethod
    def print_good_string(string):
        print(f"this is a good string : {string}")

    ## create a dunder method to work on add operation

    def __add__(self,other):

        return self.salary + other.salary

    def __truediv__(self,other):

        return self.salary / other.salary

    def __repr__(self):
        return f"""Employee('{self.name}',{self.salary},'{self.role}')"""

    def __str__(self):
        return f"""
        NAME     :{self.name}
        SALARY   :{self.salary}
        ROLE     :{self.role}
        """

e1 = Employee("E1",45000,"SDE1")
e2 = Employee("E2",40000,"SDE1")

print(e1) ##overridden repr with str
print(str(e1))
print(repr(e1))
```

```
        NAME     :E1
        SALARY   :45000
        ROLE     :SDE1


        NAME     :E1
        SALARY   :45000
        ROLE     :SDE1

Employee('E1',45000,'SDE1')
```

## 10.6 Abstract Base Class

[20]:
```
from abc import ABC,abstractmethod

class Shape(ABC):

    @abstractmethod
    def print_area(self):
        pass
```

[21]:
```
class Square(Shape):

    def __init__(self,side):
        self.side = side


s = Square(10)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-21-1c588ec6d471> in <module>
      5
      6
----> 7 s = Square(10)

TypeError: Can't instantiate abstract class Square with abstract methods print_area
```

[22]:
```
class Square(Shape):

    def __init__(self,side):
        self.side = side

    def print_area(self):

        print(self.side**2)


s = Square(10)
s.print_area()
```

```
100
```

[23]:
```
Shape() # cant create
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-23-c85743fd6946> in <module>
----> 1 Shape() # cant create

TypeError: Can't instantiate abstract class Shape with abstract methods print_area
```

## 10.7 setter and property decorator

[24]:
```
class Subscriber:

    def __init__(self,fname,lname):
        self.fname = fname
        self.lname = lname
        self._email = f"{self.fname}.{self.lname}@faloola.com"

    def name(self):
        return f"{self.fname} {self.lname}"

    def email(self):
        return self._email
s = Subscriber("Nishant","Maheshwari")
print(s.name())
print(s.email())

s.lname = "Baheti"
print(s.name())
print(s.email()) ## email will not change
```

```
Nishant Maheshwari
Nishant.Maheshwari@faloola.com
Nishant Baheti
Nishant.Maheshwari@faloola.com
```

[25]:
```
class Subscriber:

    def __init__(self,fname,lname):
        self.fname = fname
        self.lname = lname
#         self.email = f"{self.fname}.{self.lname}@faloola.com"

    def name(self):
        return f"{self.fname} {self.lname}"

    @property
    def email(self):
        return f"{self.fname}.{self.lname}@faloola.com"

s = Subscriber("Nishant","Maheshwari")
print(s.name())
print(s.email())

s.lname = "Baheti"
print(s.name())
print(s.email()) ## email changed but set as property which is not callable
```

```
Nishant Maheshwari
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-25-8f6b2bf645ba> in <module>
```

(continues on next page)

(continued from previous page)

```
     15 s = Subscriber("Nishant","Maheshwari")
     16 print(s.name())
---> 17 print(s.email())
     18
     19 s.lname = "Baheti"

TypeError: 'str' object is not callable
```

[26]:
```
class Subscriber:

    def __init__(self,fname,lname):
        self.fname = fname
        self.lname = lname
#         self.email = f"{self.fname}.{self.lname}@faloola.com"

    def name(self):
        return f"{self.fname} {self.lname}"

    @property
    def email(self):
        return f"{self.fname}.{self.lname}@faloola.com"

s = Subscriber("Nishant","Maheshwari")
print(s.name())
print(s.email)

s.lname = "Baheti"
print(s.name())
print(s.email) ## email changed
```

```
Nishant Maheshwari
Nishant.Maheshwari@faloola.com
Nishant Baheti
Nishant.Baheti@faloola.com
```

[27]:
```
class Subscriber:

    def __init__(self,fname,lname):
        self.fname = fname
        self.lname = lname
#         self.email = f"{self.fname}.{self.lname}@faloola.com"

    def name(self):
        return f"{self.fname} {self.lname}"

    @property
    def email(self):
        return f"{self.fname}.{self.lname}@faloola.com"

s = Subscriber("Nishant","Maheshwari")
print(s.name())
print(s.email)
```

(continues on next page)

(continued from previous page)

```
s.lname = "Baheti"
print(s.name())
print(s.email) ## email changed

s.email = "foo.bar@faloola.com"
```

```
Nishant Maheshwari
Nishant.Maheshwari@faloola.com
Nishant Baheti
Nishant.Baheti@faloola.com
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-27-efecc42b7d81> in <module>
     21 print(s.email) ## email changed
     22
---> 23 s.email = "foo.bar@faloola.com"

AttributeError: can't set attribute
```

[28]:
```
class Subscriber:

    def __init__(self,fname,lname):
        self.fname = fname
        self.lname = lname

    def name(self):
        return f"{self.fname} {self.lname}"

    @property
    def email(self):
        return f"{self.fname}.{self.lname}@faloola.com"

    @email.setter
    def email(self,new_email):
        name = new_email.split("@")[0]
        name_l = name.split(".")
        self.fname =  name_l[0]
        self.lname = name_l[1]



s = Subscriber("Nishant","Maheshwari")
print(s.name())
print(s.email)

s.lname = "Baheti"
print(s.name())
print(s.email) ## email changed


s.email = "foo.bar@faloola.com"
```

(continues on next page)

(continued from previous page)

```
print(s.name())
print(s.email)
```

```
Nishant Maheshwari
Nishant.Maheshwari@faloola.com
Nishant Baheti
Nishant.Baheti@faloola.com
foo bar
foo.bar@faloola.com
```

[29]:
```
class Subscriber:

    def __init__(self,fname,lname):
        self.fname = fname
        self.lname = lname

    def name(self):
        return f"{self.fname} {self.lname}"

    @property
    def email(self):
        if self.fname is None or self.lname is None:
            return "Email is not set"
        return f"{self.fname}.{self.lname}@faloola.com"

    @email.setter
    def email(self,new_email):
        name = new_email.split("@")[0]
        name_l = name.split(".")
        self.fname =  name_l[0]
        self.lname = name_l[1]

    @email.deleter
    def email(self):
        self.fname = None
        self.lname = None

s = Subscriber("Nishant","Maheshwari")
print(s.name())
print(s.email)

s.lname = "Baheti"
print(s.name())
print(s.email) ## email changed


s.email = "foo.bar@faloola.com"
print(s.name())
print(s.email)

del s.email
print(s.email)
```

(continues on next page)

(continued from previous page)

```
Nishant Maheshwari
Nishant.Maheshwari@faloola.com
Nishant Baheti
Nishant.Baheti@faloola.com
foo bar
foo.bar@faloola.com
Email is not set
```

## 10.8 Object instrospection

[30]:
```
type("hello")
```

[30]:
```
str
```

[32]:
```
id("hello")
```

[32]:
```
140138285539376
```

[33]:
```
dir("hello")
```

[33]:
```
['__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__',
```

(continues on next page)

(continued from previous page)

```
 '__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',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

[35]:
```
import inspect
```

(continues on next page)

(continued from previous page)

```
print(inspect.getmembers("Hello"))
```

```
[('__add__', <method-wrapper '__add__' of str object at 0x7f747d454270>), ('__class__',
↪<class 'str'>), ('__contains__', <method-wrapper '__contains__' of str object at␣
↪0x7f747d454270>), ('__delattr__', <method-wrapper '__delattr__' of str object at␣
↪0x7f747d454270>), ('__dir__', <built-in method __dir__ of str object at 0x7f747d454270>
↪), ('__doc__', "str(object='') -> str\nstr(bytes_or_buffer[, encoding[, errors]]) ->␣
↪str\n\nCreate a new string object from the given object. If encoding or\nerrors is␣
↪specified, then the object must expose a data buffer\nthat will be decoded using the␣
↪given encoding and error handler.\nOtherwise, returns the result of object.__str__()␣
↪(if defined)\nor repr(object).\nencoding defaults to sys.getdefaultencoding().\nerrors␣
↪defaults to 'strict'."), ('__eq__', <method-wrapper '__eq__' of str object at␣
↪0x7f747d454270>), ('__format__', <built-in method __format__ of str object at␣
↪0x7f747d454270>), ('__ge__', <method-wrapper '__ge__' of str object at 0x7f747d454270>
↪), ('__getattribute__', <method-wrapper '__getattribute__' of str object at␣
↪0x7f747d454270>), ('__getitem__', <method-wrapper '__getitem__' of str object at␣
↪0x7f747d454270>), ('__getnewargs__', <built-in method __getnewargs__ of str object at␣
↪0x7f747d454270>), ('__gt__', <method-wrapper '__gt__' of str object at 0x7f747d454270>
↪), ('__hash__', <method-wrapper '__hash__' of str object at 0x7f747d454270>), ('__init_
↪_', <method-wrapper '__init__' of str object at 0x7f747d454270>), ('__init_subclass__',
↪ <built-in method __init_subclass__ of type object at 0x557127ed5780>), ('__iter__',
↪<method-wrapper '__iter__' of str object at 0x7f747d454270>), ('__le__', <method-
↪wrapper '__le__' of str object at 0x7f747d454270>), ('__len__', <method-wrapper '__len_
↪_' of str object at 0x7f747d454270>), ('__lt__', <method-wrapper '__lt__' of str␣
↪object at 0x7f747d454270>), ('__mod__', <method-wrapper '__mod__' of str object at␣
↪0x7f747d454270>), ('__mul__', <method-wrapper '__mul__' of str object at␣
↪0x7f747d454270>), ('__ne__', <method-wrapper '__ne__' of str object at 0x7f747d454270>
↪), ('__new__', <built-in method __new__ of type object at 0x557127ed5780>), ('__reduce_
↪_', <built-in method __reduce__ of str object at 0x7f747d454270>), ('__reduce_ex__',
↪<built-in method __reduce_ex__ of str object at 0x7f747d454270>), ('__repr__', <method-
↪wrapper '__repr__' of str object at 0x7f747d454270>), ('__rmod__', <method-wrapper '__
↪rmod__' of str object at 0x7f747d454270>), ('__rmul__', <method-wrapper '__rmul__' of␣
↪str object at 0x7f747d454270>), ('__setattr__', <method-wrapper '__setattr__' of str␣
↪object at 0x7f747d454270>), ('__sizeof__', <built-in method __sizeof__ of str object␣
↪at 0x7f747d454270>), ('__str__', <method-wrapper '__str__' of str object at␣
↪0x7f747d454270>), ('__subclasshook__', <built-in method __subclasshook__ of type␣
↪object at 0x557127ed5780>), ('capitalize', <built-in method capitalize of str object␣
↪at 0x7f747d454270>), ('casefold', <built-in method casefold of str object at␣
↪0x7f747d454270>), ('center', <built-in method center of str object at 0x7f747d454270>),
↪ ('count', <built-in method count of str object at 0x7f747d454270>), ('encode', <built-
↪in method encode of str object at 0x7f747d454270>), ('endswith', <built-in method␣
↪endswith of str object at 0x7f747d454270>), ('expandtabs', <built-in method expandtabs␣
↪of str object at 0x7f747d454270>), ('find', <built-in method find of str object at␣
↪0x7f747d454270>), ('format', <built-in method format of str object at 0x7f747d454270>),
↪ ('format_map', <built-in method format_map of str object at 0x7f747d454270>), ('index
↪', <built-in method index of str object at 0x7f747d454270>), ('isalnum', <built-in␣
↪method isalnum of str object at 0x7f747d454270>), ('isalpha', <built-in method isalpha␣
↪of str object at 0x7f747d454270>), ('isascii', <built-in method isascii of str object␣
↪at 0x7f747d454270>), ('isdecimal', <built-in method isdecimal of str object at␣
↪0x7f747d454270>), ('isdigit', <built-in method isdigit of str object at 0x7f747d454270>
↪), ('isidentifier', <built-in method isidentifier of str object at 0x7f747d454270>), (
↪'islower', <built-in method islower of str object at 0x7f747d454270>), ('isnumeric',
↪<built-in method isnumeric of str object at 0x7f747d454270>), ('isprintable', <built-
↪in method isprintable of str object at 0x7f747d454270>), ('isspace', <built-in method␣
↪isspace of str object at 0x7f747d454270>), ('istitle', <built-in method istitle of str␣
↪object at 0x7f747d454270>), ('isupper', <built-in method isupper of str object at␣
↪0x7f747d454270>), ('join', <built-in method join of str object at 0x7f747d454270>), (
↪'ljust', <built-in method ljust of str object at 0x7f747d454270>), ('lower', <built-in␣
↪method lower of str object at 0x7f747d454270>), ('lstrip', <built-in method lstrip of␣
```

(continues on next page)

(continued from previous page)

```
[ ]:
```

CHAPTER

# ELEVEN

# FUNCTION CACHING

[1]:
```
import time
```

[3]:
```
def some_func(n):

    time.sleep(n)
    return n
```

[4]:
```
some_func(5)
```

[4]:
```
5
```

- caching / memoization

[22]:
```
from functools import lru_cache
import datetime
@lru_cache(maxsize=3) ## latest 3 calls are saved
def some_func(n):

    time.sleep(n)
    return n
```

[23]:
```
start = datetime.datetime.now()

some_func(10)

end = datetime.datetime.now()

print("time taken at first ",end-start)
```

```
time taken at first  0:00:10.001500
```

[24]:
```
start = datetime.datetime.now()

some_func(10)

end = datetime.datetime.now()

print("time taken at second call ",end-start)
```

```
time taken at second call  0:00:00.000255
```

```
[ ]:
```

CHAPTER

# TWELVE

# STRING OPERATIONS

```
[1]: "heLlo World".capitalize() # first letter uppercase all other lower
[1]: 'Hello world'

[2]: "hello World".casefold() # lowercase
[2]: 'hello world'

[3]: "Hello World".center(50) # put the string in center
[3]: '                    Hello World                     '

[4]: " Hello World ".center(50,"x") #put the string in center and fill padding with x
[4]: 'xxxxxxxxxxxxxxxxx Hello World xxxxxxxxxxxxxxxxxx'

[5]: "Hello World Hello".count("Hello") # count freq
[5]: 2

[6]: "Hello World".endswith("d")
[6]: True

[7]: "Hello\tWorld".expandtabs(4) # expand \t tabs with 4 spaces
[7]: 'Hello   World'

[8]: "Hello World".find("World") # returns index from which word's first occurance is␣
     ↪starting otherwise -1
[8]: 6

[9]: "Hello World".index("World")
[9]: 6

[10]: "Hello World".index("World",6,10) # index and find is similar only difference is that␣
      ↪this will throw value error
      ---------------------------------------------------------------------------
      ValueError                                Traceback (most recent call last)
```

(continues on next page)

(continued from previous page)

```
<ipython-input-10-a7a86f317d1d> in <module>
----> 1 "Hello World".index("World",6,10) # index and find is similar only difference is␣
↪that this will throw value error

ValueError: substring not found
```

[20]:
```
"Hello World".isalnum(),"123".isalnum() #alphnumeric
```
[20]:
```
(False, True)
```

[16]:
```
"HelloWorld".isalpha(),"Hello World".isalpha()
```
[16]:
```
(True, False)
```

[22]:
```
"12".isdecimal(),"Hello World".isdecimal()
```
[22]:
```
(True, False)
```

[24]:
```
"12".isdigit(),"Hello World".isdigit()
```
[24]:
```
(True, False)
```

[25]:
```
"12".isnumeric(),"Hello World".isnumeric()
```
[25]:
```
(True, False)
```

[27]:
```
"Hello World".isspace(), "             ".isspace() # if string contains only spaces
```
[27]:
```
(False, True)
```

[35]:
```
" ".join(("hello","world","!","Nishant")) # join by separator
```
[35]:
```
'hello world ! Nishant'
```

[37]:
```
x = {"fname":"Nishant","lname":"Baheti"}
" ".join(x.values())
```
[37]:
```
'Nishant Baheti'
```

[39]:
```
"Hello World".lower()
```
[39]:
```
'hello world'
```

[40]:
```
"Hello World".upper()
```
[40]:
```
'HELLO WORLD'
```

[46]:
```
x = "Jello world".maketrans("J","H")
"Jello world".translate(x)
```
[46]:
```
'Hello world'
```

[48]:
```
"I could eat bananas all day, bananas bananas are my favorite fruit".rpartition("bananas
↪") # starts from last occurance
```

```
[48]: ('I could eat bananas all day, bananas ', 'bananas', ' are my favorite fruit')

[49]: "Hello, world, Nishant".split(",")
[49]: ['Hello', ' world', ' Nishant']

[51]: "    Hello World".lstrip()
[51]: 'Hello World'

[52]: "Hello World   ".rstrip()
[52]: 'Hello World'

[53]: "        Hello World      ".strip()
[53]: 'Hello World'

[54]: "Hello World".startswith("H")
[54]: True

[55]: "Ello There maTe".swapcase()
[55]: 'eLLO tHERE MAtE'

[56]: "my name is nishant and i am exploring string methods in python".title()
[56]: 'My Name Is Nishant And I Am Exploring String Methods In Python'

[62]: "500".zfill(20)
[62]: '00000000000000000500'
```

CHAPTER

THIRTEEN

# INDICES AND TABLES

- genindex
- modindex
- search