در این داکیومنت به معرفی تعدادی از Magic Methodeها و یا Dubble Under Methode ها پرداخته شده است. Dubble under Methodeها متدهای از پیش ساخته شده توسط پایتون هستند که کاربر می تواند با بازنویسی کردن آنها قابلیتهایشان را تغییر بدهد.

**\_\_new\_\_ و \_\_init\_\_** : این متد برای ساخت یک شی از کلاس استفاده می شود. هنگام ساخت یک شی، این متد از کلاس مربوطه صدا زده می شود سپس برای آغاز و مقداردهی آن شی، متد \_\_init\_\_ صدا زده می شود.

class A(object):

def \_\_new\_\_(cls):

print("Creating instance")

return super(A, cls).\_\_new\_\_(cls)

def \_\_init\_\_(self):

print("Init is called")

**\_\_str\_\_** : این متد برای هنگامی که می خواهیم از یک شی پرینت بگیریم و اطلاعات مورد نظر ما را نشان دهد، استفاده می شود.

در حالت عادی اگر تابع ()print را به همراه نام شی صدا کنیم، نام کلاس مربوطه به همراه آدرس حافظه ی شی چاپ می گردد. برای تغییر در این نحوه ی چاپ، می توان متد \_\_str\_\_ را تعریف کرد. رشته ای که توسط این متد برگشت داده می شود هنگام پرینت شدن شی نشان داده می شود.

def \_\_str\_\_(self):

s='this is \_\_str\_\_ printing'

return s

**\_\_del\_\_** : وقتی رفرنسهای اشاره کننده به یک شی همگی حذف می شوندُُ این متد صدا زده می شود. یعنی هنگامیکه برنامه ی حاوی شی تمام می شود و یا شی حذف می گردد این متد مربوط به شی صدا زده می شود.

def \_\_del\_\_(self):

print("Destructor called")

**\_\_repr\_\_** : وقتی بخواهیم یک شی را به صور رشته نمایش دهیم از این متد استفاده می کنیم. این کار بیشتر به منظور دیباگ کردن استفاده می شود. در حالت عادی با صدا زدن متد ()repr به همراه نام شی، نام کلاس مربوطه به همراه آدرس حافظه ی شی نشان داده می شود. برای تغییر این نمایش، متد \_\_repr\_\_ را بازنویسی می کنیم.

**\_\_hash\_\_**: تابع ()hash برای هش کردن یک شی استفاده می شود و خروجی آن از نوع int هست. برای بازنویسی کردن آن از متد \_\_hash\_\_ در کلاس آن شی استفاده می کنیم.

**\_\_bool\_\_**: خروجی این متد برای هرنوع از متغیرها و اشیا، True یا False می باشد. اگر در کلاس مربوط به شی ای، این متد بازنویسی نشده باشد، خروجی این متد برای آن شی True می باشد. در غیر اینصورت بستگی به تعریف آن در کلاس دارد.

class custom():

val = 0

def \_\_init\_\_(self, num):

self.val = num

def \_\_bool\_\_(self):

return bool(self.val)

# custom objects

x = custom(0)

y = custom(52)

print(bool(x)) # False

print(bool(y)) # True

**\_\_call\_\_**: اگر بخواهیم به یک شی قابلیت call شدن بدهید ، باید این متد را در کلاس مربوط به آن شی تعریف کنید.

**\_\_len\_\_** : می توان این متد را در بدنه یک کلاس بازنویسی کرد تا بتوان طول atribute های شی آن کلاس را به دست آورد.

class foo(object):

def \_\_init\_\_(self,obs=[])

self.data = obs

def \_\_len\_\_(self):

return len(self.data)

**\_\_iter\_\_** : از این متد برای iterable کردن شی استفاده می شود.

class Counter:

def \_\_init\_\_(self, start, end):

self.num = start

self.end = end

def \_\_iter\_\_(self):

return self

def \_\_next\_\_(self):

if self.num > self.end:

raise StopIteration

else:

self.num += 1

return self.num - 1