1 May 1

Python Basics

e) Class competition:

```
raise_amount = 1.04
def __init__(self, name, price):
    self.name = name
    self.price = price.
def raise_price(self):
    self.price = self.price + competition.raise_amount.
```

→ obj → creates instance

→ instance variable.

2 or `self.raise_amount`.

Class variable.

if not declared in the instance you need to use class.

e) Every instance has in built special attribute.

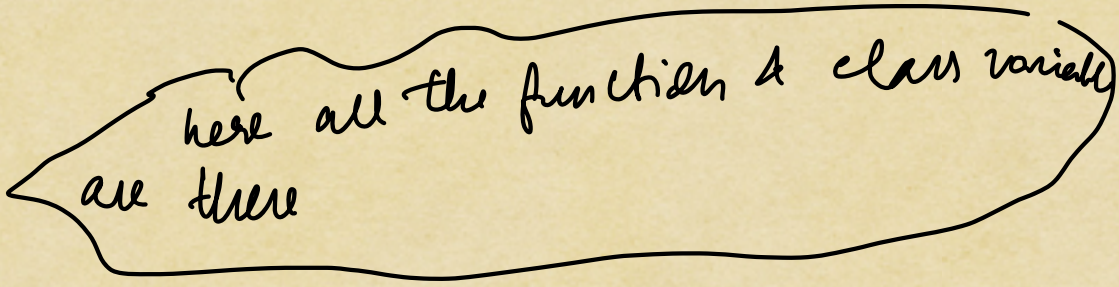
```
simulation = Competition('simulation', 100)
simulation.__dict__
```

special attribute

⇒ {'name': 'simulation', 'price': 104.0}

Class variable & function not present in dictionary of instance

a) Class also has special dict : Complition.__dict__



a) Change made to class is also change all instance.

a) Change made to instance will only change that instance & not that class.

Q) In Python class variable are Public, they can be accessed outside class which is generally not true in JAVA data encapsulation (where data inside class is not available outside class)

simulation.name = "smd"

↓

accessed outside & changed outside.

↓ So that no one can change outside we have to make it private.

```
def __init__(self, name, prize):
    self.__name = name
    self.__prize = prize
```

→ now it can read from outside but cannot be changed

↓

just private hack

```
s1 = Competition('Valo tour', 25)
s1.name = 'Lol tour'
s1.__dict__
```

⇒ `{'_competition__name': 'Valo tour', '_competition__prize': 25, '__name': 'lol tour'}`

⇒ Now `s1._competition__prize = 300`

↓

this will change.

↓

but not good practice

↓

↓

Use getters & setter.

```
def get_name(self):
    return self.__name

def set_name(self):
    return.__name = name
```

a) `def __repr__(self):`
`return f"({self.__name}, {self.__size})"`

b) print ⟶ invokes string representation of the object.

c) `def __str__(self):`
`return f"{self.__name} & {self.__size}"`

↳ another way to represent

↳ if you use print(debate).

↳ if you use only debate.

a) 1+2 is actually int.__add__(1, 2)

b) s1 = saving(30)
s2 = saving(20)

```
class Savings:
    def __init__(self, amount):
        self.__amount = amount

    def __add__(self, other):
        return self.__amount +
               other.__amount

    def __sub__(self, other):
        return self.__amount - other.
                                __amount

    def __mul__(self, othe):
```

otherwise coerce ↑

- not * or +

so have to make it so

__mul__ is only for float (*)

float.

then __floordiv__ (//) - integer

__mod__ (%) - integer

__pow__ (**)

Built in class methods

-> def __len__(self):
    return len(self.__participants)

-> def __iter__(self)
    self.__index = 0
    return self.

next(obj)

learn more about __iter__ & next

-> name = property(get_name, set_name, del_name).

→ Decorator

better way

```
@property
def name(self):
    return self.__name
```

```
@name.setter
def name(self, value):
    self.__name = value
```

```
@name.deleter
def name(self):
    del self.__name
```

→ New getter & setter is for instance for class it is called class Method.

Ex

→ Class Competition:

```
__raise_amount = 1.04

@classmethod
def get_raise_amount(cls):
    return cls.__raise_amount.
```

→ class

getter for class variable

```
@classmethod.
def set_raise_amount(cls):
    return cls.__raise_amount.
```

setter for class variable

↓

better way

Now

```
class Rectangle:
    @staticmethod
    def area(x,y):
        return x*y
```

0) Abstract class

↓

Classes with no methods

An Base & derived class??

Abstract Base class??

2 phases → Optimize models that are developed, faster to be practical, optimizing querying, query

SQL & Python → Production Query.
Pyspark. basic. & code.

MLOPs → still building their stack.
↳ Basic questions.

Collections Models, Recovery Models, Basics

Vivek asked - 2 models
↳ run parallel 36 model recovery, how to optimize.

training
↓
is easy.

model inference.
↓
is fast

a) if you nearest neighbours problem id of the index on the tables.

Q) Say

→ Problem statement.

→ Data → size, level, id, target variable, definition.

→ preparation → cleaning → outliers, missing.

→ Analysis → Univariate, Bivariate
↓ ↘
Binning, correlation
information value

→ Model → Benchmark Model, Best Model.

→ Metrics →

→ Interpretation →

→ Notebooks →

→ Production. →