Day_2 Python

Topics Covered:

 Functions :

Lambda Function

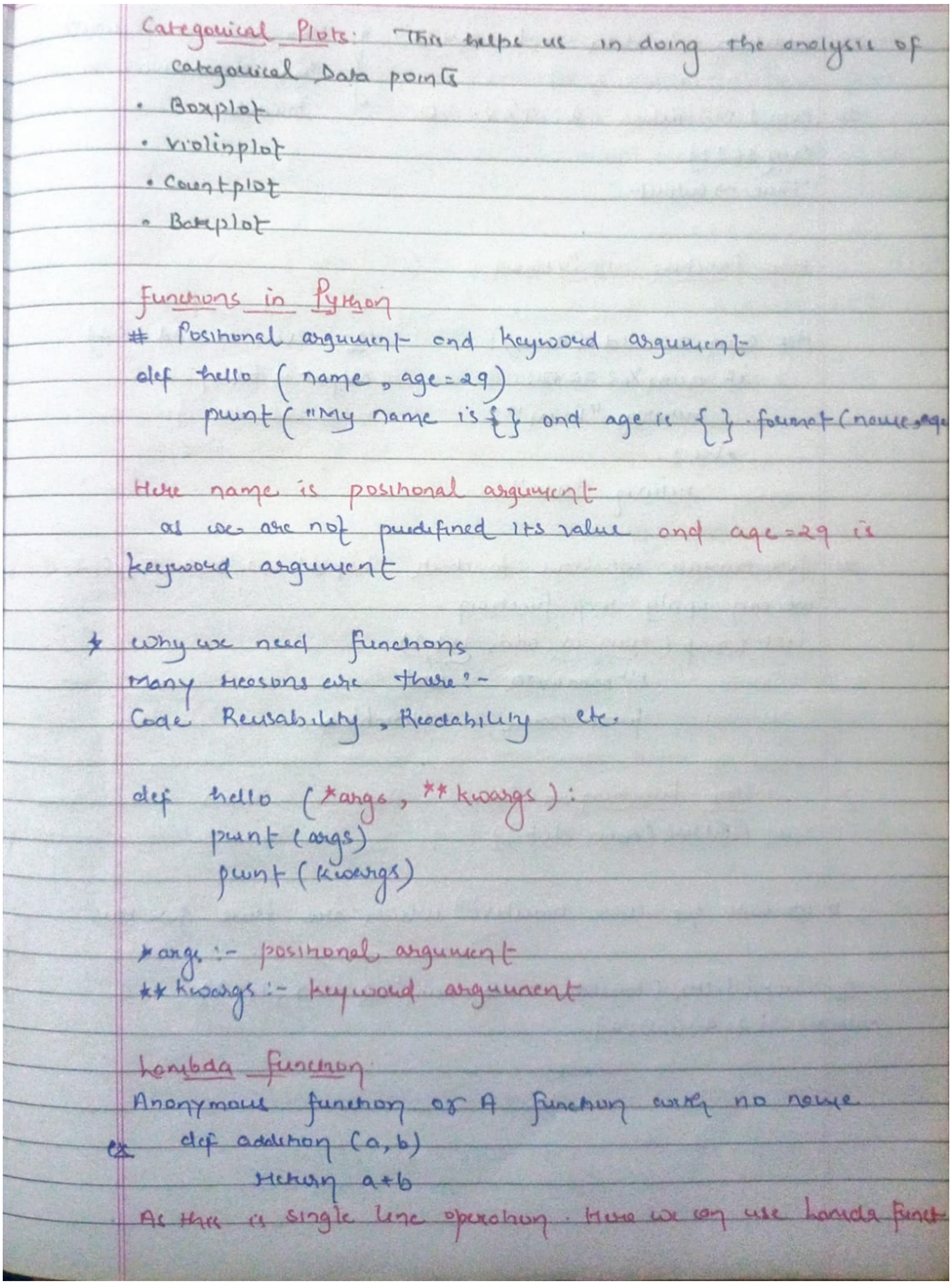
Map function

Filter Function

List Comprehension

String Formatting

Python list Iterables vs Iterators



Categorical Plots: This helps us in doing the analysis of categorical Data points

- Boxplot
- violinplot
- Countplot
- Barplot

Functions in Python

```
# Positional argument and keyword argument
def hello (name, age=29)
      print ("My name is {} and age is {}.format(name,age
```

Here name is positional argument as we are not predefined its value and age=29 is keyword argument

* why we need functions
Many reasons are there :-
Code Reusability, Readability etc.

```
def hello (*args, **kwargs):
      print (args)
      print (kwargs)
```

*args :- positional argument
**kwargs :- keyword argument

Lambda function
Anonymous function or A function with no name

```
ex   def addition (a,b)
          return a+b
```

As this is single line operation. Here we can use lamda funct

addition = lambda a,b : a+b } Here we cannot write two expressions in lambda.

even 1 = lambda a: a%2==0
even 1(12)
True → output

Map function in Python

```
def even_or_odd (num):
    if num%2==0:
        return "Even"
    else:
        return "Odd"
```

ex: even_or_odd (24)
output: Even.

For example we have to check list=[1,2,3,4,5,6,7,8]
we can apply map function

$$\text{list (map (} \underbrace{\text{even\_or\_odd}}_{\substack{\text{1st parameter} \\ \text{funct- name}}}, \underbrace{\text{lst}}_{\substack{\text{2nd} \\ \text{iterable}}} \text{))}$$

* Filter function

list (filter (even, lst))

we will get those numbers which are true for this

ex list (filter (lambda num: num%2==0, lst))
output. [2,4,6,8,0]

Date
Page

List Comprehension

It provides a concise way to create lists. It consist of brackets containing an expression followed by a for clause, then zero or more for or if clause. The expression can be anything, meaning you can put in all kinds of object in.

```
lst1=[]
def lst_square(lst):                    ex: lst_square([1,2,3,4,5])
    for i in lst:                       output: [1,4,9,16,25]
        lst1.append(i*i)
    return lst1
```

Convert same code in list comprehension:

```
lst = []
[i*i For i in lst]   } one line of code
                       we will get same output
```

String formatting in Python

```
def greeting(name):
    return "Hello {}. Welcome to the community ".format(name)
                  placeholder

def welcome_email(name, age):
    return "Welcome {name}. Your age is {age}".format(
                                              name,age)
```

→ In this case format can have parameters in any order as we have already assigned temporary variable

→ If we haven't assigned temporary variable then there should be in order in format parameters

for eg. Welcome Aniket. Your age is 19.
→ Welcome 19. Your age is Aniket } if we dont assign and can misprint.

Python List Iterables vs Iterators

List is Iterable

```
lst = [1,2,3,4,5,6,7]
for i in lst:
    print(i)
```

output:
```
1
2
3
4
5
6
7
```

{ All values will be allocated in memory }

* iter (lst)
< list iterator at 0x2080128 ... > } In this list is stored in particular address

Here memory is not initialized for element in list.

* It will be initialized one by one if we add next inbuilt function

```
lst1 = iter(lst)
next(lst1)
```

After last it will give error

Suppose we have millions of element in list. Here it is very useful to use Iterator

Pyforest - lazy - import of all Python Data Science libraries
with the help of this all popular libraries are there when you need them. If you don't use a library, it won't be imported. When you are done with your script you can export the Python code for the import statement

ex
```
df = pd.read_csv(" ..... ")
```
it will run but we haven't imported pandas library

```
df.head()
```

output: we will see result :

⇒