Python ka Chilla

Basics of Python

01-My First Program

In [1]:
```
print("Hello World")
print(2+3)
print("Learning with Baba Aammar")
```

```
Hello World
5
Learning with Baba Aammar
```

02-Operators

In [2]:
```
print(2+3)
print(10-8)
print(5*4)
print(6/3)
print(7//8)
print(2**4)
print(6%5)
print((2+4)-8*3/2**1)
```

```
5
2
20
2.0
0
16
1
-6.0
```

DMAS Rule Division Addition Subtraction Multiplication Left to Right Evaluation

03-Strings

In [3]:
```
print("Hello World")
print("Learning Python")
print("With Baba Aammar")
print('Test with Single Quotes')
print("Test with Double Quotes")
print('''Test with Triple Quotes''')
```

```
Hello World
Learning Python
With Baba Aammar
Test with Single Quotes
Test with Double Quotes
Test with Triple Quotes
```

04-Comments

Shortcut Key of Comment is **Ctrl+/**

In [4]:
```
print("First Comment")    #type this to comment out (Ctrl+/)
print("Hello,Second Comment")  #String Here
print(4+5-3)  #Numerical Operation
```

```
First Comment
Hello,Second Comment
6
```

05-Variables

In [5]:
```
#Enter value in a variable
a=9
a
```

Out[5]:
```
9
```

In [6]:
```
#Enter String in a Variable
b='isb'
b
```

Out[6]:
```
'isb'
```

06-Functions

In [7]:
```
def my_function():
  print("I am Learning")

my_function()
```

```
I am Learning
```

In [8]:
```
my_function()
```

```
I am Learning
```

In [9]:
```
def my_funct():
  print("Islamabad")
```

In [10]:
```
my_funct()
```

```
Islamabad
```

07-Input Variables

In [11]:
```
no=input("Enter your First Name ")
"Your First Name is",no
```

```
Enter your First Name Mahnoor
```

Out[11]:
```
('Your First Name is', 'Mahnoor')
```

In [12]:
```
no1=input("Enter your Last Name ")
```

```
"Your Last Name is",no1
```

```
Enter your Last Name Khushbakht
('Your Last Name is', 'Khushbakht')
```

Out[12]:

08-If Else

In [13]:

```
a= 36
b= 83
if a > b:
 print("a")
else:
 print("b")
```

```
b
```

In [14]:

```
a= 52
b= 52
if a > b:
 print("a is greater to b")
elif b > a:
 print("b is less to a")
else:
 print("they are equal")
```

```
they are equal
```

09_Loops

In [15]:

```
# While Loops
i = 1
while i < 6:
 print(i)
 i += 1
```

```
1
2
3
4
5
```

In [16]:

```
#For Loops
party_items = ["Snacks", "Candies", "Drinks"]
for x in party_items:
  print(x)
```

```
Snacks
Candies
Drinks
```

In [17]:

```
for i in "Chilla":
  print(i)
```

```
C
h
i
```

```
l
l
a
```

10-Import Libraries

In [18]:
```
import math
print("The value of pi is",math.pi)
import statistics
x=[10,20,30,40]
print(statistics.mean(x))
```

```
The value of pi is 3.141592653589793
25
```

In [19]:
```
import math
print("The value of pi is",math.pi)
import statistics
x=[10,20,30,40]
print(statistics.median(x))
```

```
The value of pi is 3.141592653589793
25.0
```

In [20]:
```
import statistics
x=[10,20,30,40]
print(statistics.mode(x))
```

```
10
```

11-Conditional Logics

In [21]:
```
age_at_school=10
wakas_age=input("How old is wakas now")
wakas_age=int(wakas_age)
print(type(wakas_age))
print(wakas_age==age_at_school)
```

```
How old is wakas now11
<class 'int'>
False
```

12-Type Conversions

In [22]:
```
int_no = 13
float_no = 23.7

print("Data type:",type(int_no))
print("Data type of Float before Type Casting:",type(float_no))

float_no = int(float_no)
print("Data type of Float after Type Casting:",type(float_no))

sum_no = int_no + float_no

print("Sum of int and float:",sum_no)
print("Data type of the sum:",type(sum_no))
```

```
Data type: <class 'int'>
Data type of Float before Type Casting: <class 'float'>
Data type of Float after Type Casting: <class 'int'>
Sum of int and float: 36
Data type of the sum: <class 'int'>
```

Basic Data Structures

Tuples

- Ordered collection of elements

- Enclosed in round braces/ paranthesis

- Diff types of elements can be stored

- Once your elements are stored u cannot change them(unmutatable)

In [23]:
```
tup=(1,'Sami',48,True)
tup
```

Out[23]: (1, 'Sami', 48, True)

In [24]:
```
type(tup) #to know the type
```

Out[24]: tuple

Indexing in tuple

In [25]:
```
tup[1]
```

Out[25]: 'Sami'

In [26]:
```
tup[3]
```

Out[26]: True

In [27]:
```
tup[0:4]
```

Out[27]: (1, 'Sami', 48, True)

In [28]:
```
tup[0:3] #last element is exclusive
```

Out[28]: (1, 'Sami', 48)

In [29]:

```
len(tup) #count of element in tuple
```

Out[29]: 4

In [30]:
```
tup1=(3,'China',4.5,False)
tup1
```

Out[30]: (3, 'China', 4.5, False)

In [31]:
```
tup + tup1 #Concatenation
```

Out[31]: (1, 'Sami', 48, True, 3, 'China', 4.5, False)

In [32]:
```
tup3=(10,6,20,40,5,1)
tup3
```

Out[32]: (10, 6, 20, 40, 5, 1)

In [33]:
```
min(tup3) #minimum num
```

Out[33]: 1

In [34]:
```
max(tup3) #maximum num
```

Out[34]: 40

In [35]:
```
tup3*2 #duplication
```

Out[35]: (10, 6, 20, 40, 5, 1, 10, 6, 20, 40, 5, 1)

List

- Ordered collection of elements

- Enclosed in square [] braces/ brackets

- Mutateable, can change the value

In [36]:
```
L1= ['Khushbakht',74,False]
L1
```

Out[36]: ['Khushbakht', 74, False]

In [37]:
```
type(L1)
```

Out[37]: list

In [38]:
```
len(L1)
```

Out[38]:
```
3
```

In [39]:
```
L2=[18,2.3,'Multan','Youtube',True]
L2
```

Out[39]:
```
[18, 2.3, 'Multan', 'Youtube', True]
```

In [40]:
```
L1+L2
```

Out[40]:
```
['Khushbakht', 74, False, 18, 2.3, 'Multan', 'Youtube', True]
```

In [41]:
```
L1*3
```

Out[41]:
```
['Khushbakht', 74, False, 'Khushbakht', 74, False, 'Khushbakht', 74, False]
```

In [42]:
```
L1.reverse()
L1
```

Out[42]:
```
[False, 74, 'Khushbakht']
```

In [43]:
```
L1.remove(74)
L1
```

Out[43]:
```
[False, 'Khushbakht']
```

In [44]:
```
L2.append('Sehar')
L2
```

Out[44]:
```
[18, 2.3, 'Multan', 'Youtube', True, 'Sehar']
```

In [45]:
```
L3=L2*3
L3
```

Out[45]:
```
[18,
 2.3,
 'Multan',
 'Youtube',
 True,
 'Sehar',
 18,
 2.3,
 'Multan',
 'Youtube',
 True,
 'Sehar',
 18,
```

```
2.3,
'Multan',
'Youtube',
True,
'Sehar']
```

In [46]:
```
L3.count('Sehar')
```

Out[46]:
```
3
```

In [47]:
```
L3.clear() #remove all elements from lisr
```

In [48]:
```
L3
```

Out[48]:
```
[]
```

In [49]:
```
L4=[3,6,1,56,45,12,34,58,2]
L4
```

Out[49]:
```
[3, 6, 1, 56, 45, 12, 34, 58, 2]
```

In [50]:
```
L4.sort()
```

In [51]:
```
L4
```

Out[51]:
```
[1, 2, 3, 6, 12, 34, 45, 56, 58]
```

In [52]:
```
L=[3,6]
L
```

Out[52]:
```
[3, 6]
```

In [53]:
```
l7=L.copy()
l7
```

Out[53]:
```
[3, 6]
```

In [54]:
```
L1.index(False)
```

Out[54]:
```
0
```

In [55]:
```
L4.index(12)
```

Out[55]:
```
4
```

In [56]:

```
L1.insert(2,'isb')
```

In [57]:
```
 L1
```

Out[57]:
```
[False, 'Khushbakht', 'isb']
```

In [58]:
```
L1.remove(False)
```

In [59]:
```
L1
```

Out[59]:
```
['Khushbakht', 'isb']
```

In [60]:
```
L1.extend(L2)
```

In [61]:
```
L1
```

Out[61]:
```
['Khushbakht', 'isb', 18, 2.3, 'Multan', 'Youtube', True, 'Sehar']
```

Dictionaries

- Unordered collection of elements

- Key & Value

- Curly Braces {}

- Mutateable/Change the values

In [62]:
```
party_items1 = {"Chocolates":50 , "Candies":60 , "Snacks":100 , "Cold Drinks":200}
party_items1
```

Out[62]:
```
{'Chocolates': 50, 'Candies': 60, 'Snacks': 100, 'Cold Drinks': 200}
```

In [63]:
```
type(party_items1)
```

Out[63]:
```
dict
```

In [64]:
```
val=party_items1.values()
val
```

Out[64]:
```
dict_values([50, 60, 100, 200])
```

In [65]:
```
key=party_items1.keys()
key
```

Out[65]:
```
dict_keys(['Chocolates', 'Candies', 'Snacks', 'Cold Drinks'])
```

In [66]:
```
party_items1["Juices"]=130
party_items1
```

Out[66]:
```
{'Chocolates': 50,
 'Candies': 60,
 'Snacks': 100,
 'Cold Drinks': 200,
 'Juices': 130}
```

In [67]:
```
item=party_items1.pop('Snacks')
item
```

Out[67]:
```
100
```

In [68]:
```
item1={'Samosa':150,'Chicken Rolls':70}
party_items1.update(item1)
party_items1
```

Out[68]:
```
{'Chocolates': 50,
 'Candies': 60,
 'Cold Drinks': 200,
 'Juices': 130,
 'Samosa': 150,
 'Chicken Rolls': 70}
```

In [69]:
```
party_items2=party_items1.copy()
party_items2
```

Out[69]:
```
{'Chocolates': 50,
 'Candies': 60,
 'Cold Drinks': 200,
 'Juices': 130,
 'Samosa': 150,
 'Chicken Rolls': 70}
```

In [70]:
```
party_items2.clear()
party_items2
```

Out[70]:
```
{}
```

In [71]:
```
key1={'a','b','c','d','e','f'}
alphabets=dict.fromkeys(key1)
alphabets
```

Out[71]:
```
{'d': None, 'f': None, 'c': None, 'b': None, 'e': None, 'a': None}
```

In [72]:
```
value='alphabet'
alphabets=dict.fromkeys(key1,value)
alphabets
```

```
{'d': 'alphabet',
```

Out[72]:
```
 'f': 'alphabet',
 'c': 'alphabet',
 'b': 'alphabet',
 'e': 'alphabet',
 'a': 'alphabet'}
```

In [73]:
```
party_items1.get('Candies')
```

Out[73]:
```
60
```

In [74]:
```
party_items1.items()
```

Out[74]:
```
dict_items([('Chocolates', 50), ('Candies', 60), ('Cold Drinks', 200), ('Juices', 130),
('Samosa', 150), ('Chicken Rolls', 70)])
```

In [75]:
```
party=party_items1.popitem()
party
```

Out[75]:
```
('Chicken Rolls', 70)
```

In [76]:
```
party_items1
```

Out[76]:
```
{'Chocolates': 50,
 'Candies': 60,
 'Cold Drinks': 200,
 'Juices': 130,
 'Samosa': 150}
```

In [77]:
```
party=party_items1.popitem()
party
```

Out[77]:
```
('Samosa', 150)
```

In [78]:
```
party_items1
```

Out[78]:
```
{'Chocolates': 50, 'Candies': 60, 'Cold Drinks': 200, 'Juices': 130}
```

In [79]:
```
it={'Salad':78}
party_items1.update(it)
party_items1
```

Out[79]:
```
{'Chocolates': 50,
 'Candies': 60,
 'Cold Drinks': 200,
 'Juices': 130,
 'Salad': 78}
```

In [80]:
```
price=party_items1.setdefault('Salad')
price
```

Out[80]:
```
78
```

In [81]:
```
party_items1.update(alphabets)
party_items1
```

Out[81]:
```
{'Chocolates': 50,
 'Candies': 60,
 'Cold Drinks': 200,
 'Juices': 130,
 'Salad': 78,
 'd': 'alphabet',
 'f': 'alphabet',
 'c': 'alphabet',
 'b': 'alphabet',
 'e': 'alphabet',
 'a': 'alphabet'}
```

Sets

- Unordered or Unindexed

- Curly Braces

- No Brackets Allowed

In [82]:
```
s1={1,7.2,'Sehar','isb','lah',False}
s1
```

Out[82]:
```
{1, 7.2, False, 'Sehar', 'isb', 'lah'}
```

In [83]:
```
s1.add('kar')
s1
```

Out[83]:
```
{1, 7.2, False, 'Sehar', 'isb', 'kar', 'lah'}
```

In [84]:
```
s1.add('kar')
```

In [85]:
```
s1
```

Out[85]:
```
{1, 7.2, False, 'Sehar', 'isb', 'kar', 'lah'}
```

In [86]:
```
s1.remove('kar')
```

In [87]:
```
s1
```

Out[87]:
```
{1, 7.2, False, 'Sehar', 'isb', 'lah'}
```

In [88]:
```
s2={3,7.2,'Maha','rwp','Mult',True}
s2
```

Out[88]: {3, 7.2, 'Maha', 'Mult', True, 'rwp'}

In [89]:
```
s1.union(s2)
```

Out[89]: {1, 3, 7.2, False, 'Maha', 'Mult', 'Sehar', 'isb', 'lah', 'rwp'}

In [90]:
```
s1.intersection(s2)
```

Out[90]: {True, 7.2}

In [91]:
```
s2.clear()
```

In [92]:
```
s2
```

Out[92]: set()

In [93]:
```
type(s2)
```

Out[93]: set

In [94]:
```
s2={3,7.2,'Maha','rwp','Mult',True}
s2
```

Out[94]: {3, 7.2, 'Maha', 'Mult', True, 'rwp'}

In [95]:
```
s1.difference(s2)
```

Out[95]: {False, 'Sehar', 'isb', 'lah'}

In [96]:
```
s2.difference(s1)
```

Out[96]: {3, 'Maha', 'Mult', 'rwp'}

In [97]:
```
s3={1,9,5,6,3}
s3
```

Out[97]: {1, 3, 5, 6, 9}

In [98]:
```
s4={1,3,15,63,2}
s4
```

Out[98]: {1, 2, 3, 15, 63}

In [99]:
```
s3.difference_update(s4)
```

```
s3
```

Out[99]: {5, 6, 9}

In [100…
```
s5=s3.copy()
s5
```

Out[100… {5, 6, 9}

In [101…
```
s5.clear()
s5
```

Out[101… set()

In [102…
```
s4.remove(63)
s4
```

Out[102… {1, 2, 3, 15}

In [103…
```
s4.discard(15)
s4
```

Out[103… {1, 2, 3}

In [104…
```
s3.add(3)
s3
```

Out[104… {3, 5, 6, 9}

In [105…
```
s4.intersection_update(s3)
s4
```

Out[105… {3}

In [106…
```
s4={1,3,15,63,2}
s4
```

Out[106… {1, 2, 3, 15, 63}

In [107…
```
s3.isdisjoint(s4)
```

Out[107… False

In [108…
```
a={1,2,3}
a
```

{1, 2, 3}

Out[108…

In [109…
```
b={1,2,3,4,5}
b
```

Out[109… {1, 2, 3, 4, 5}

In [110…
```
a.issubset(b)
```

Out[110… True

In [111…
```
b.issubset(a)
```

Out[111… False

In [112…
```
a.pop()
```

Out[112… 1

In [113…
```
a.pop()
```

Out[113… 2

In [114…
```
a
```

Out[114… {3}

In [115…
```
a.symmetric_difference(b)
```

Out[115… {1, 2, 4, 5}

In [116…
```
a.symmetric_difference_update(b)
```

In [117…
```
a
```

Out[117… {1, 2, 4, 5}

In [118…
```
b={'a','b'}
b
```

Out[118… {'a', 'b'}

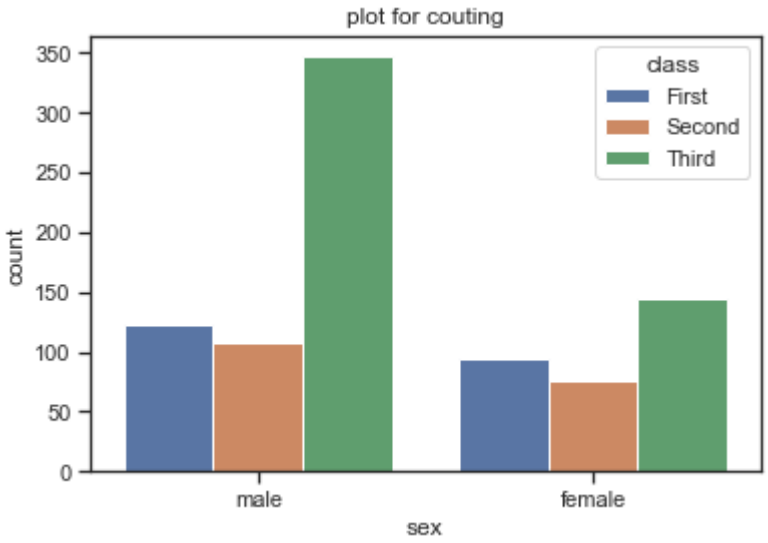
In [119…
```
a.update(b)
a
```

Out[119… {1, 2, 4, 5, 'a', 'b'}

Graphs

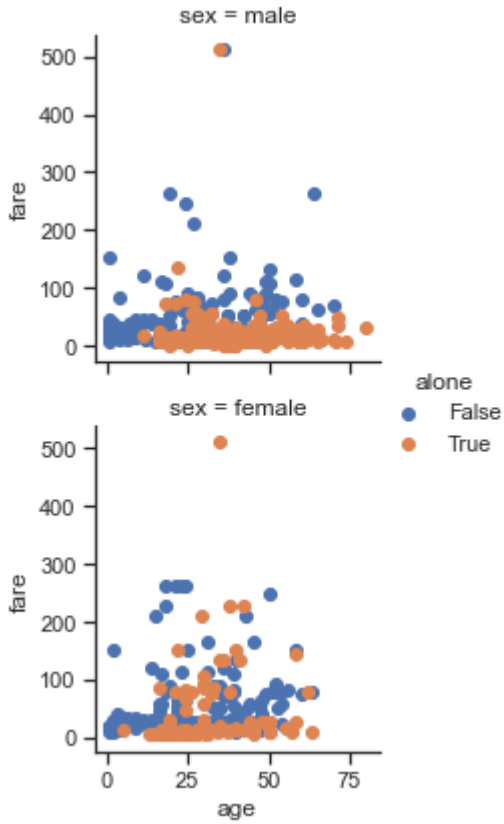
In [120…

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="ticks", color_codes=True)
titanic=sns.load_dataset("titanic")
pl=sns.countplot(x='sex', data=titanic, hue='class')
pl.set_title("plot for couting")
plt.show()
```



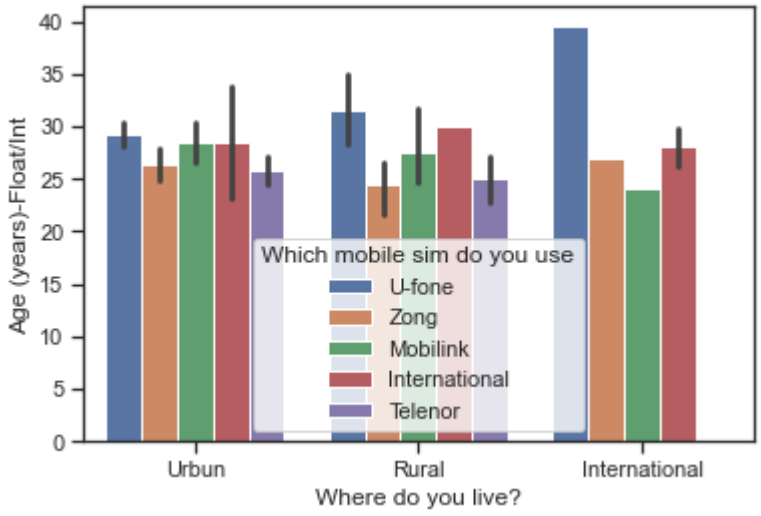
In [121…

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="ticks", color_codes=True)
titanic=sns.load_dataset("titanic")
g=sns.FacetGrid(titanic, row='sex', hue='alone')
g=(g.map(plt.scatter,"age","fare").add_legend())
plt.show()
```



In [122…

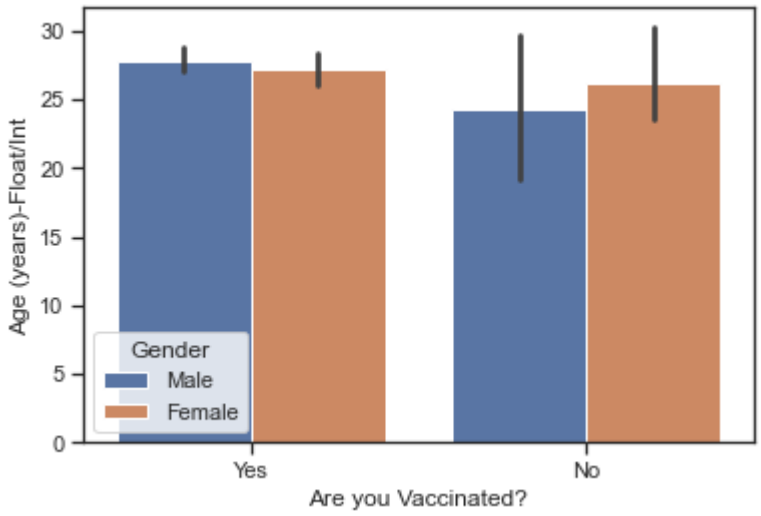
```
# import library
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#import data from file
chilla = pd.read_csv("chilla.csv")
sns.set_theme(style="ticks", color_codes=True)

p= sns.barplot(x="Where do you live?",y="Age (years)-Float/Int", hue="Which mobile sim
plt.show()
```



In [123…

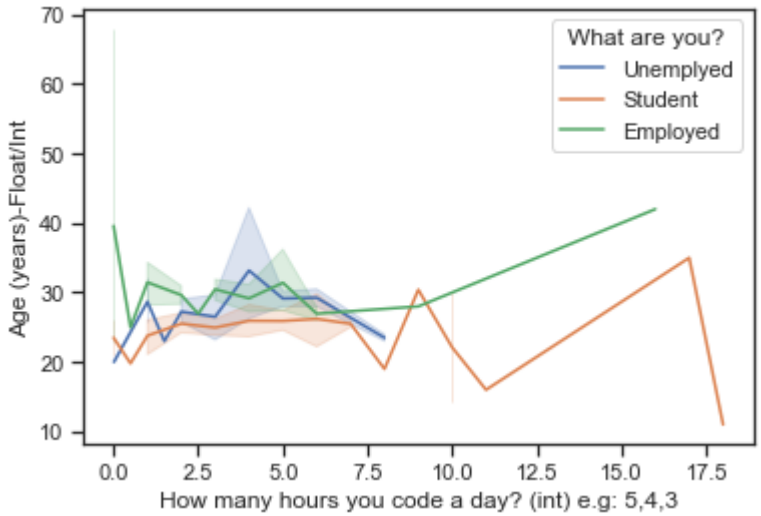
```
# import library
import pandas as pd
import seaborn as sns
```

```
import matplotlib.pyplot as plt
#import data from file
chilla = pd.read_csv("chilla.csv")
sns.set_theme(style="ticks", color_codes=True)

p= sns.barplot(x="Are you Vaccinated?",y="Age (years)-Float/Int", hue="Gender",  data=c
plt.show()
```



In [124…

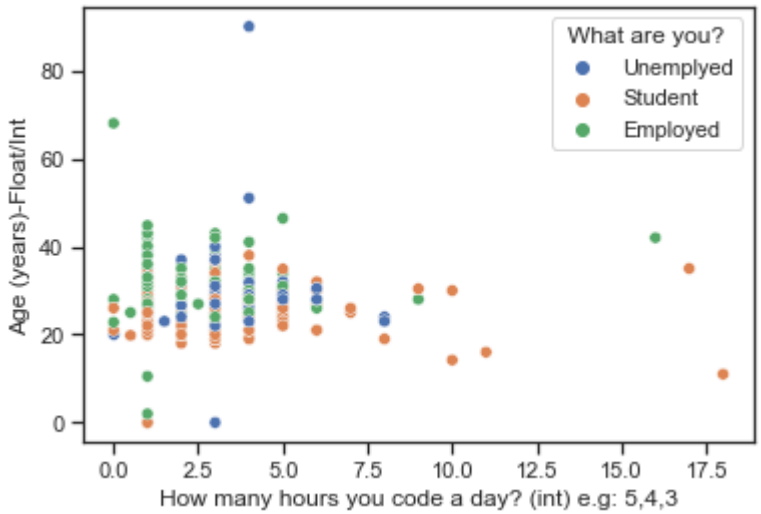
```
# import library
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#import data from file
chilla = pd.read_csv("chilla.csv")
sns.set_theme(style="ticks", color_codes=True)

p= sns.lineplot(x="How many hours you code a day? (int) e.g: 5,4,3",y="Age (years)-Floa
plt.show()
```



In [125…

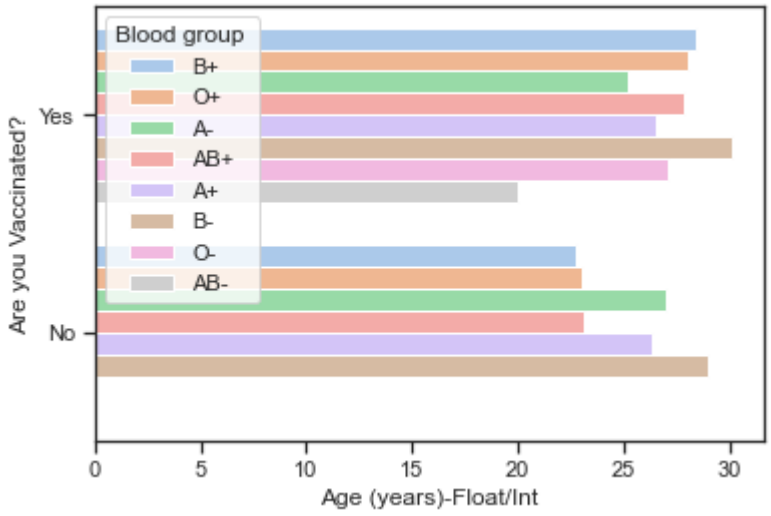
```
# import library
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#import data from file
chilla = pd.read_csv("chilla.csv")
sns.set_theme(style="ticks", color_codes=True)

p= sns.scatterplot(x="How many hours you code a day? (int) e.g: 5,4,3",y="Age (years)-F
plt.show()
```



In [126…

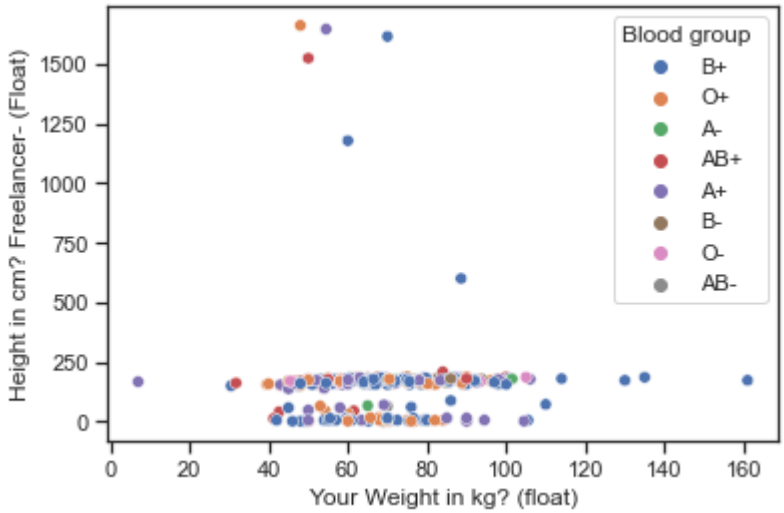
```
# import library
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#import data from file
chilla = pd.read_csv("chilla.csv")
sns.set_theme(style="ticks", color_codes=True)

p= sns.barplot(x="Age (years)-Float/Int",y="Are you Vaccinated?", hue="Blood group ",
plt.show()
```



In [127…

```
# import library
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#import data from file
```

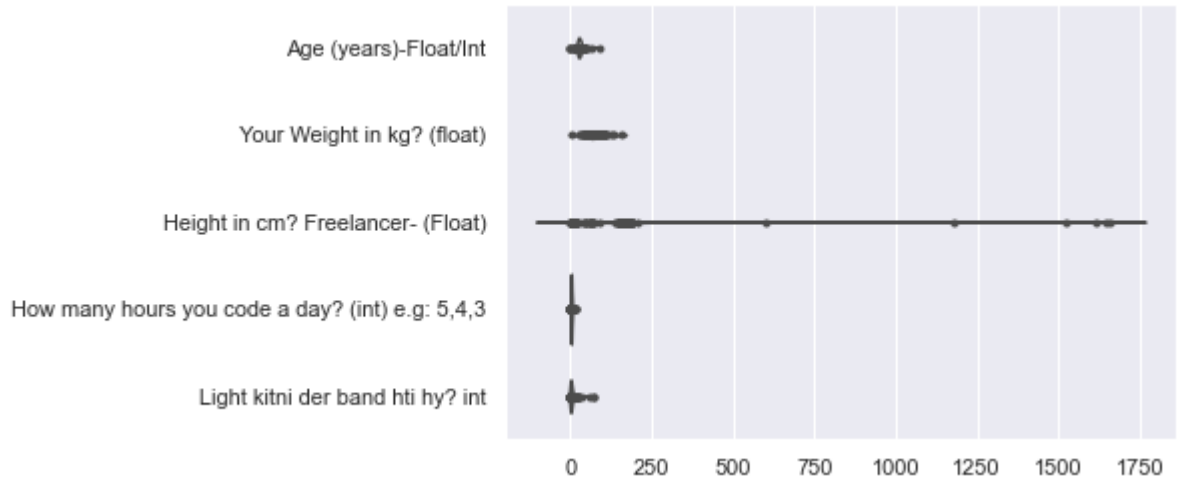
```
chilla = pd.read_csv("chilla.csv")
sns.set_theme(style="ticks", color_codes=True)

p= sns.scatterplot(x="Your Weight in kg? (float)",y="Height in cm? Freelancer- (Float)"
plt.show()
```



In [128…

```
import numpy as np
import seaborn as sns

sns.set_theme()
chilla = pd.read_csv("chilla.csv")

# Show each distribution with both violins and points
sns.violinplot(data=chilla, palette="light:g", inner="points", orient="h")
```

Out[128… <AxesSubplot:>



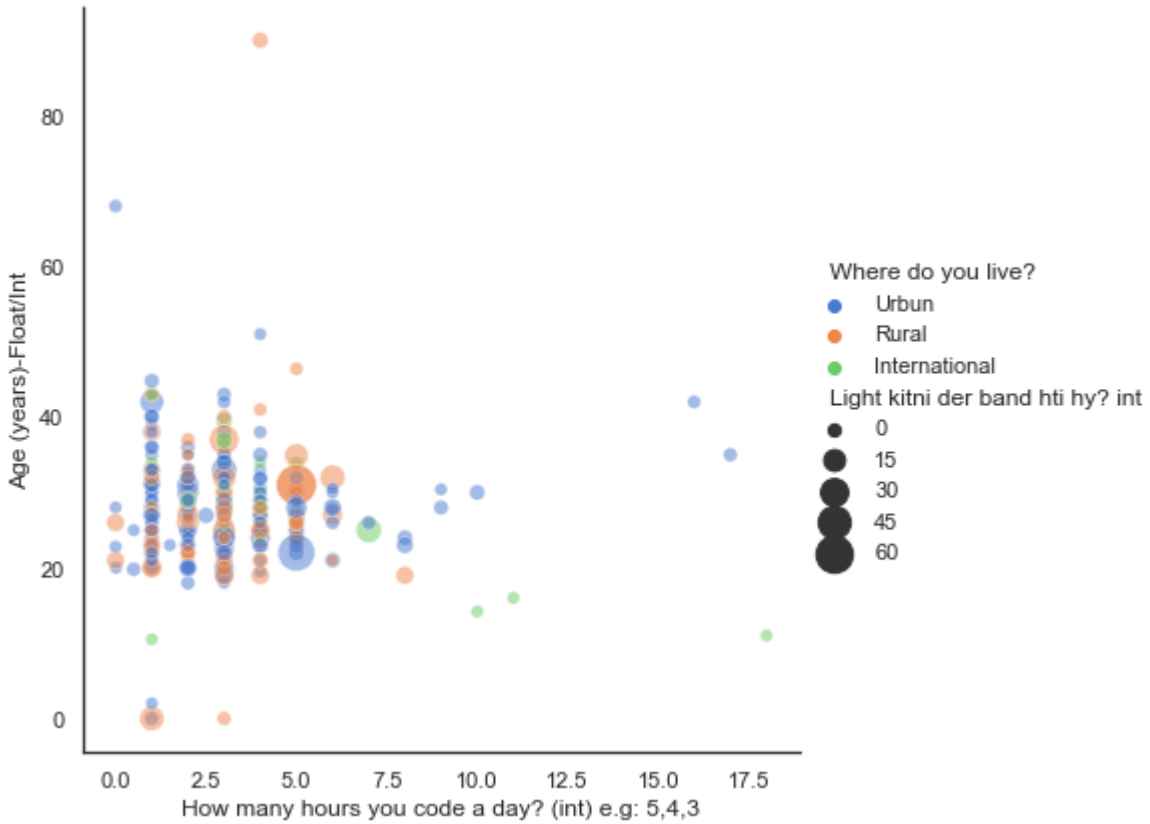
In [129…

```
import seaborn as sns
import pandas as pd
import numpy as np
sns.set_theme(style="white")

# Load the example mpg dataset
chilla = pd.read_csv("chilla.csv")
```

```
# Plot miles per gallon against horsepower with other semantics
sns.relplot(x="How many hours you code a day? (int) e.g: 5,4,3", y="Age (years)-Float/I
            sizes=(40, 400), alpha=.5, palette="muted",
            height=6, data=chilla)
```

Out[129…

```
<seaborn.axisgrid.FacetGrid at 0x2c3be158640>
```

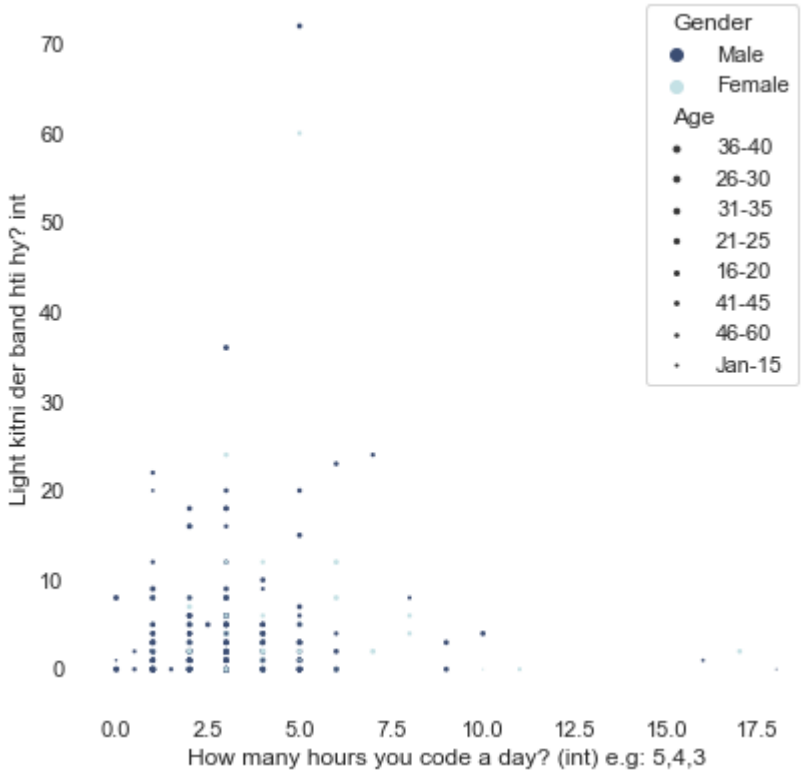


In [130…

```
import seaborn as sns
import pandas as pd
import numpy as np
sns.set_theme(style="white")

# Load the example mpg dataset
chilla = pd.read_csv("chilla.csv")

# Draw a scatter plot while assigning point colors and sizes to different
# variables in the dataset
f, ax = plt.subplots(figsize=(6.5, 6.5))
sns.despine(f, left=True, bottom=True)

sns.scatterplot(x="How many hours you code a day? (int) e.g: 5,4,3", y="Light kitni der
                hue="Gender", size="Age",
                palette="ch:r=-.2,d=.3_r",

                sizes=(1, 8), linewidth=0,
                data=chilla, ax=ax)
```

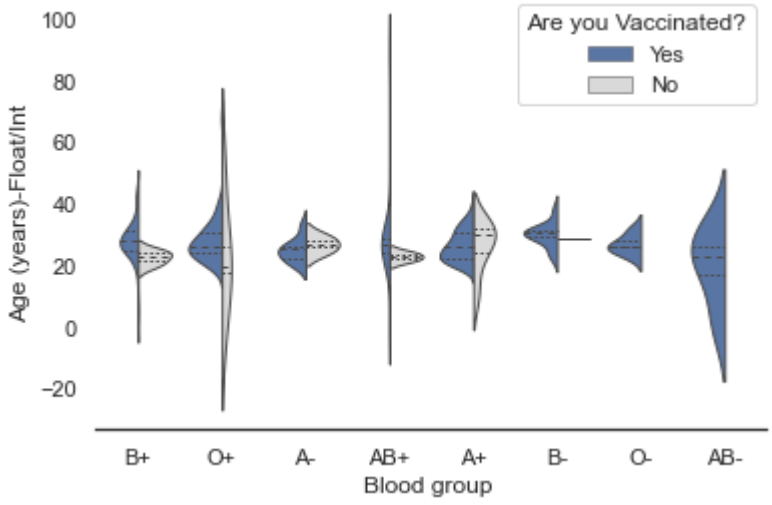
Out[130…

```
<AxesSubplot:xlabel='How many hours you code a day? (int) e.g: 5,4,3', ylabel='Light kit
ni der band hti hy? int'>
```



In [131…

```
import seaborn as sns
import pandas as pd
import numpy as np
sns.set_theme(style="white")

# Load the example mpg dataset
chilla = pd.read_csv("chilla.csv")

# Draw a nested violinplot and split the violins for easier comparison
sns.violinplot(data=chilla, x="Blood group ", y="Age (years)-Float/Int", hue="Are you V
               split=True, inner="quart", linewidth=1,
               palette={"Yes": "b", "No": ".85"})
sns.despine(left=True)
```



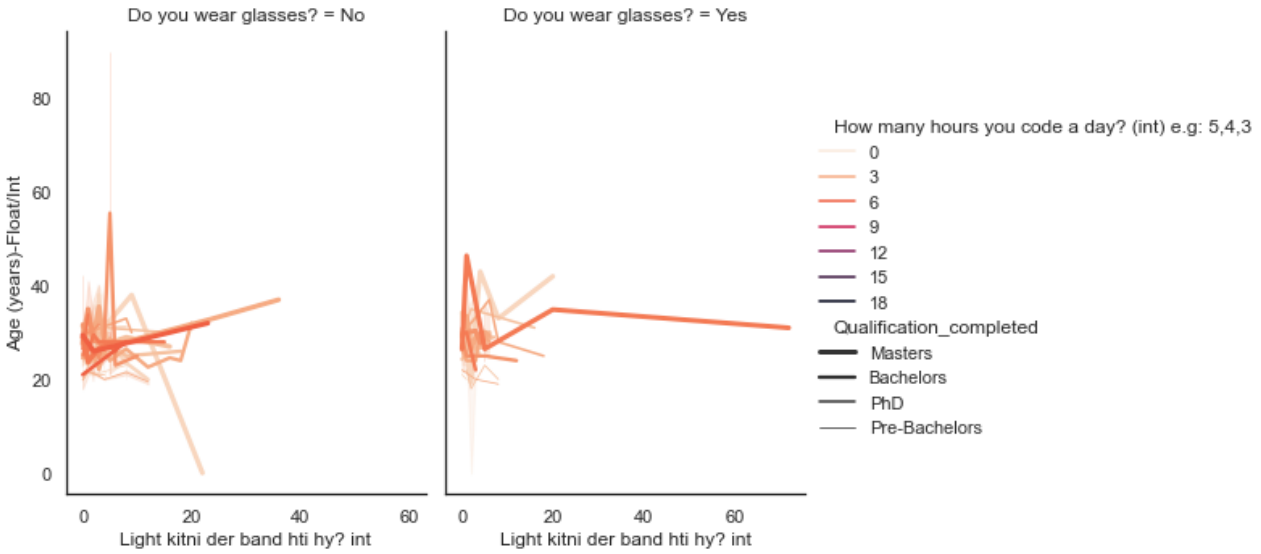
In [132…

```
import seaborn as sns
```

```
import pandas as pd
import numpy as np
sns.set_theme(style="white")

# Load the example mpg dataset
chilla = pd.read_csv("chilla.csv")

# Define the palette as a list to specify exact values
# palette = sns.color_palette("rocket_r")

sns.relplot(
    data=chilla,
    x="Light kitni der band hti hy? int", y="Age (years)-Float/Int",
    hue="How many hours you code a day? (int) e.g: 5,4,3", size="Qualification_complete
    kind="line", palette="rocket_r",
    height=5, aspect=.75, facet_kws=dict(sharex=False),
)
```

Out[132…

```
<seaborn.axisgrid.FacetGrid at 0x2c3bcdf61f0>
```

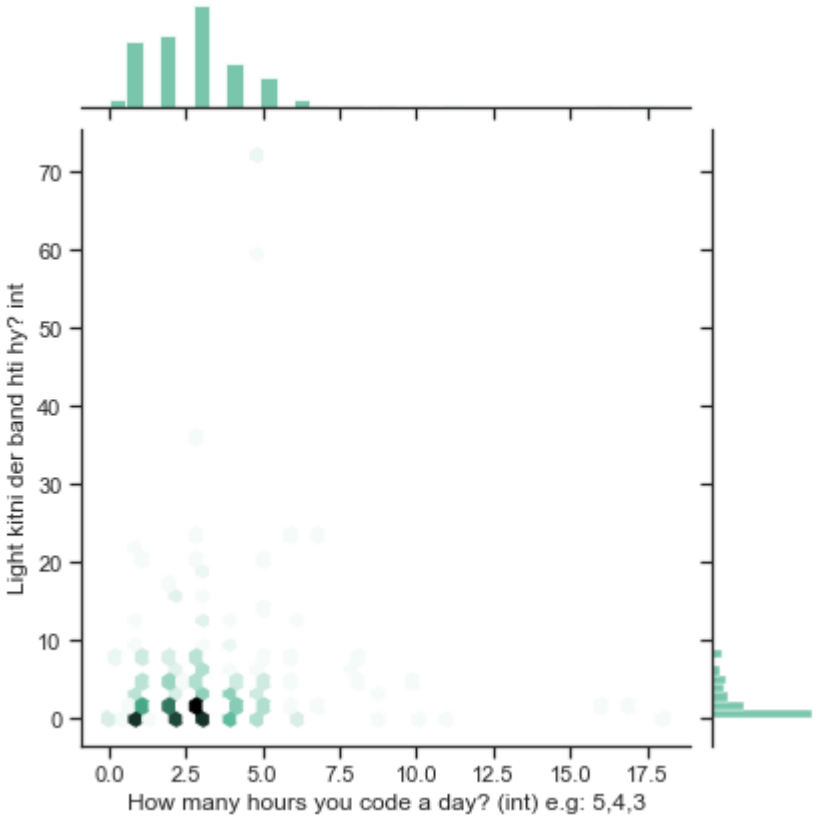


In [133…

```
import seaborn as sns
import pandas as pd
import numpy as np
sns.set_theme(style="white")

# Load the example mpg dataset
chilla = pd.read_csv("chilla.csv")

sns.set_theme(style="ticks")



sns.jointplot(data=chilla, x="How many hours you code a day? (int) e.g: 5,4,3", y="Ligh
```

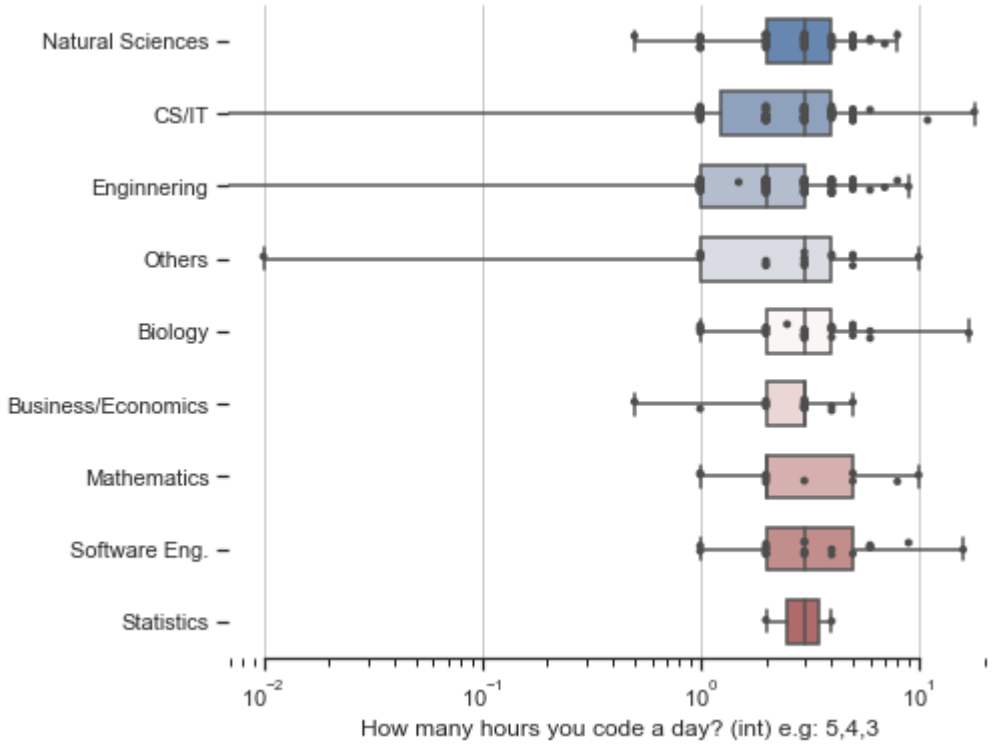
Out[133…

```
<seaborn.axisgrid.JointGrid at 0x2c3be1a6eb0>
```



In [134…

```
import seaborn as sns
import pandas as pd
import numpy as np
sns.set_theme(style="white")

# Load the example mpg dataset
chilla = pd.read_csv("chilla.csv")

sns.set_theme(style="ticks")

# Initialize the figure with a logarithmic x axis
f, ax = plt.subplots(figsize=(7, 6))
ax.set_xscale("log")



# Plot the orbital period with horizontal boxes
sns.boxplot(x="How many hours you code a day? (int) e.g: 5,4,3", y="field_of_study", da
            whis=[0, 100], width=.6, palette="vlag")

# Add in points to show each observation
sns.stripplot(x="How many hours you code a day? (int) e.g: 5,4,3", y="field_of_study",
              size=4, color=".3", linewidth=0)

# Tweak the visual presentation
ax.xaxis.grid(True)
ax.set(ylabel="")
sns.despine(trim=True, left=True)
```

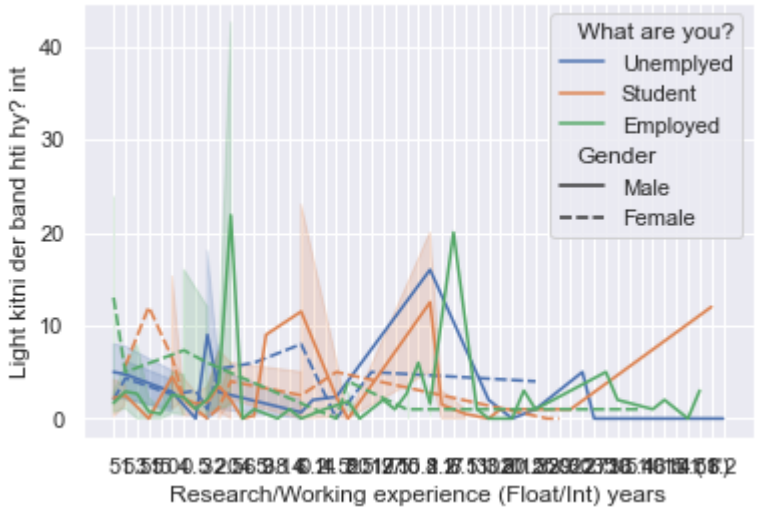


In [135…

```
import seaborn as sns
import pandas as pd
import numpy as np
sns.set_theme(style="darkgrid")
# Load the example mpg dataset
chilla = pd.read_csv("chilla.csv")
# Plot the responses for different events and regions
sns.lineplot(x="Research/Working experience (Float/Int) years", y="Light kitni der band
             hue="What are you?", style="Gender",
             data=chilla)
```

Out[135…

```
<AxesSubplot:xlabel='Research/Working experience (Float/Int) years', ylabel='Light kitni
der band hti hy? int'>
```



In [136…

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="whitegrid")
```

```
chilla = pd.read_csv("chilla.csv")
f, ax = plt.subplots(figsize=(11, 6))
sns.violinplot(data=chilla, palette="Set3", bw=.2, cut=1, linewidth=1)

# Finalize the figure
ax.set(ylim=(-.7, 1.05))
sns.despine(left=True, bottom=True)
```

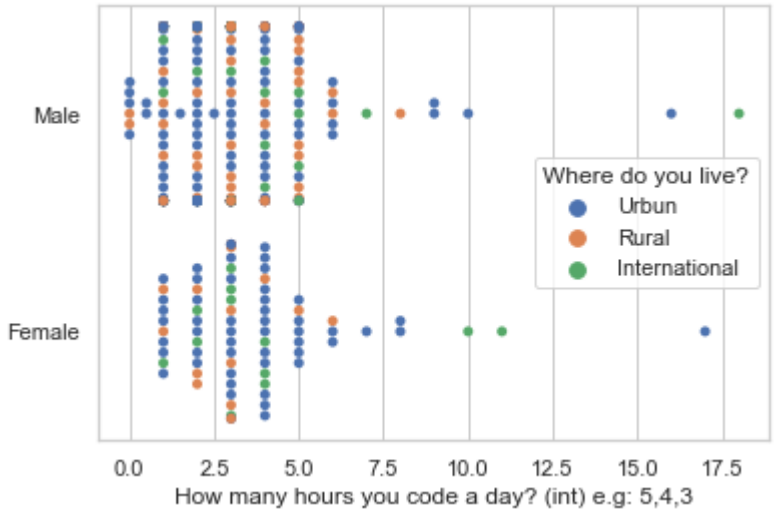
In [137…

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="whitegrid")

# Load the example dataset of brain network correlations
chilla = pd.read_csv("chilla.csv")

# Draw a categorical scatterplot to show each observation
ax = sns.swarmplot(data=chilla, x="How many hours you code a day? (int) e.g: 5,4,3", y=
ax.set(ylabel="")
```

```
C:\Users\786\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 63.8%
of the points cannot be placed; you may want to decrease the size of the markers or use
stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\786\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 6.5%
of the points cannot be placed; you may want to decrease the size of the markers or use
stripplot.
  warnings.warn(msg, UserWarning)
```

Out[137…

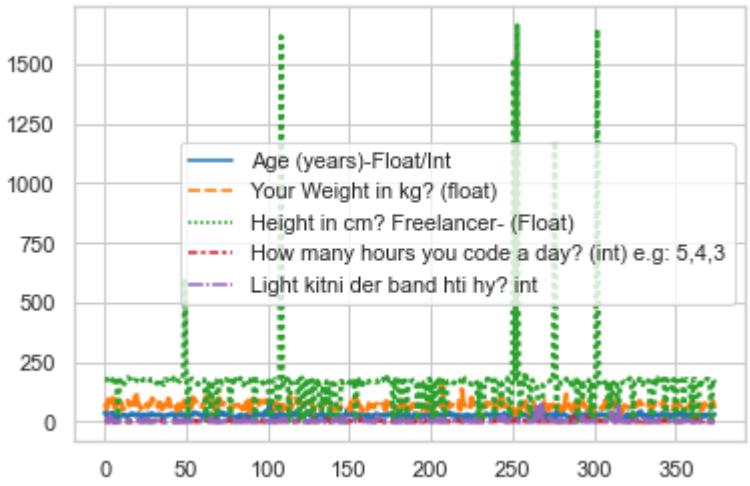
```
[Text(0, 0.5, '')]
```



In [138…

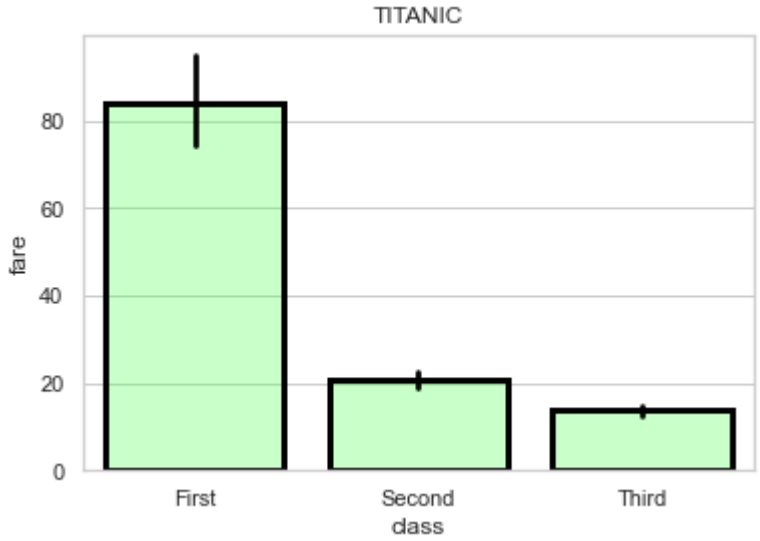
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="whitegrid")

chilla = pd.read_csv("chilla.csv")
sns.lineplot(data=chilla, palette="tab10", linewidth=2.5)
```

Out[138… <AxesSubplot:>



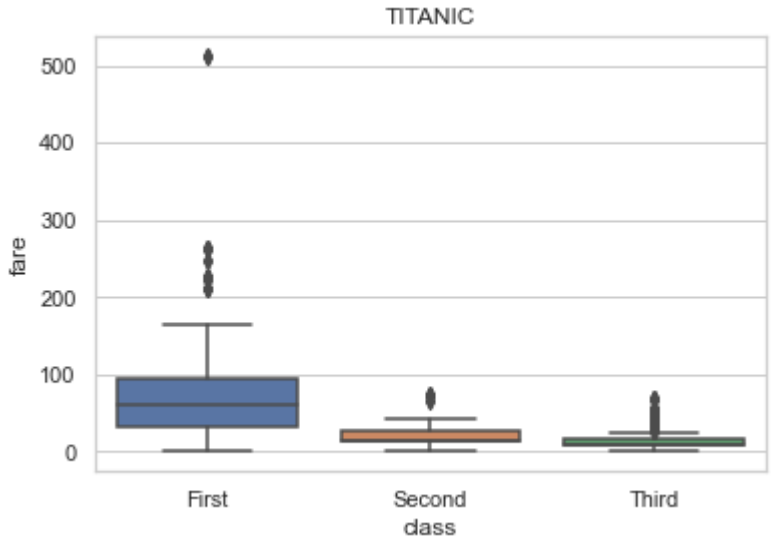
In [139…

```
import seaborn as sns
import matplotlib.pyplot as plt
from numpy import mean

kashti= sns.load_dataset("titanic")

sns.barplot(x='class',y='fare',data=kashti,linewidth=3,facecolor=(0.3 ,1 ,0.3 , 0.3),er

plt.title('TITANIC')
plt.show()
```



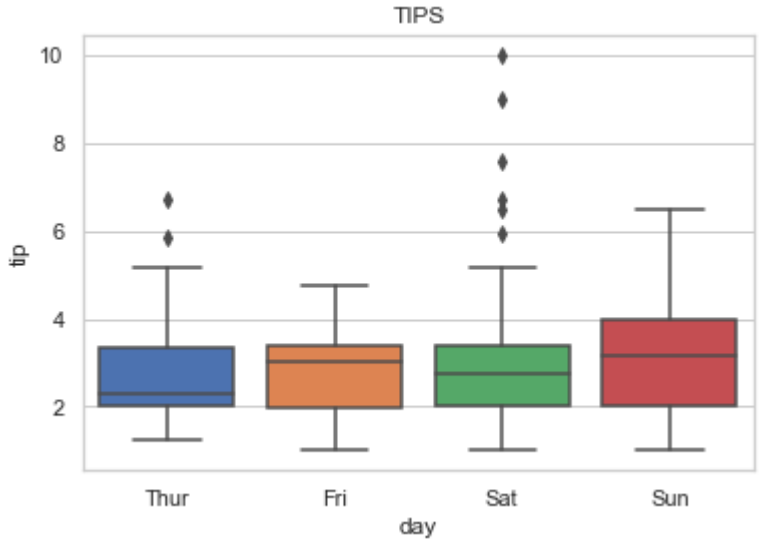
In [140…

```
import seaborn as sns
import matplotlib.pyplot as plt
kashti= sns.load_dataset("titanic")
sns.boxplot(x='class',y='fare',data=kashti)
plt.title('TITANIC')
plt.show()
```



In [141…

```
import seaborn as sns
import matplotlib.pyplot as plt
tip= sns.load_dataset("tips")
sns.boxplot(x='day',y='tip',data=tip,saturation=1)
plt.title('TIPS')
plt.show()
```



In [142…

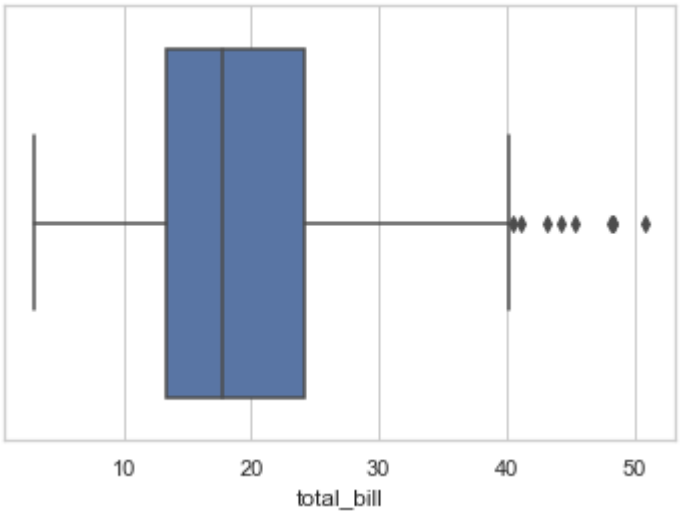
```
import seaborn as sns
import matplotlib.pyplot as plt
tip= sns.load_dataset("tips")
tip.describe()
```

Out[142…

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

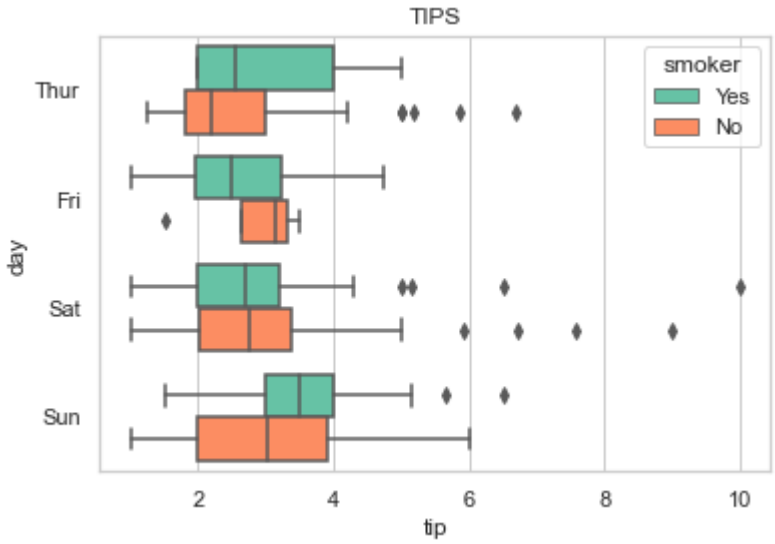
In [143…

```
import seaborn as sns
import matplotlib.pyplot as plt
tip= sns.load_dataset("tips")
sns.boxplot(x=tip['total_bill'])
plt.show()
```



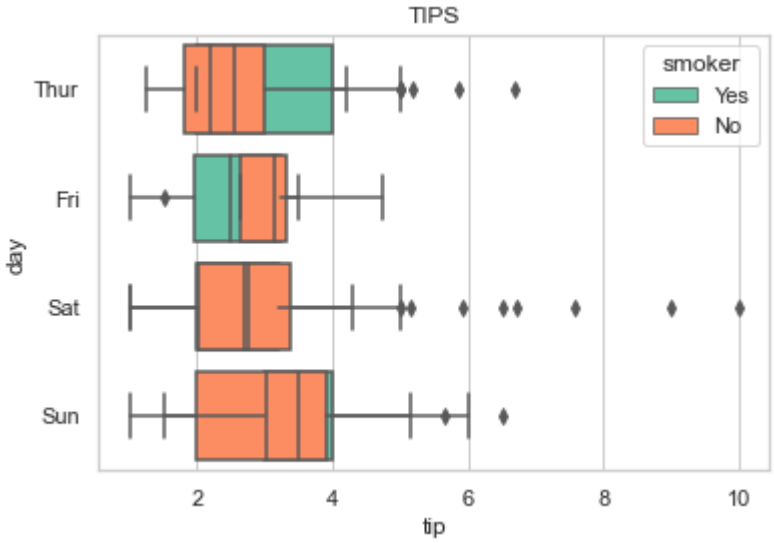
In [144…

```
import seaborn as sns
import matplotlib.pyplot as plt
tip= sns.load_dataset("tips")
sns.boxplot(x='tip',y='day',hue="smoker",palette="Set2",dodge=True,data=tip,saturation=
plt.title('TIPS')
plt.show()
```



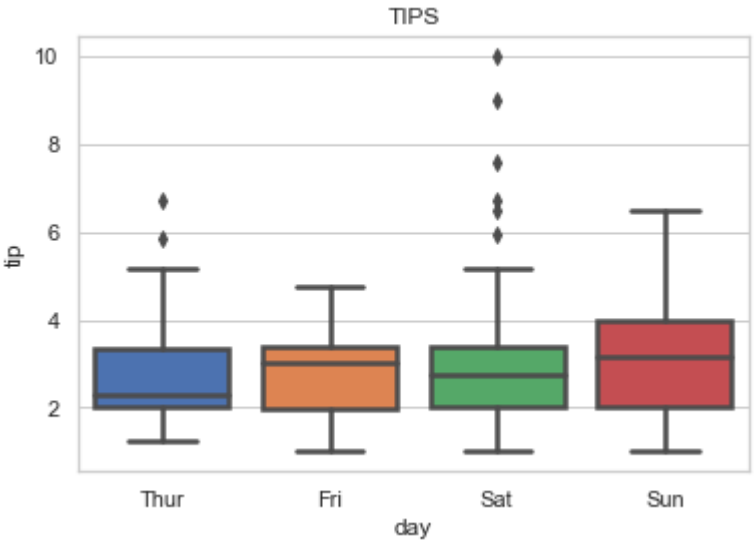
In [145…

```
import seaborn as sns
import matplotlib.pyplot as plt
tip= sns.load_dataset("tips")
sns.boxplot(x='tip',y='day',hue="smoker",palette="Set2",dodge=False,data=tip,saturation
plt.title('TIPS')
plt.show()
```



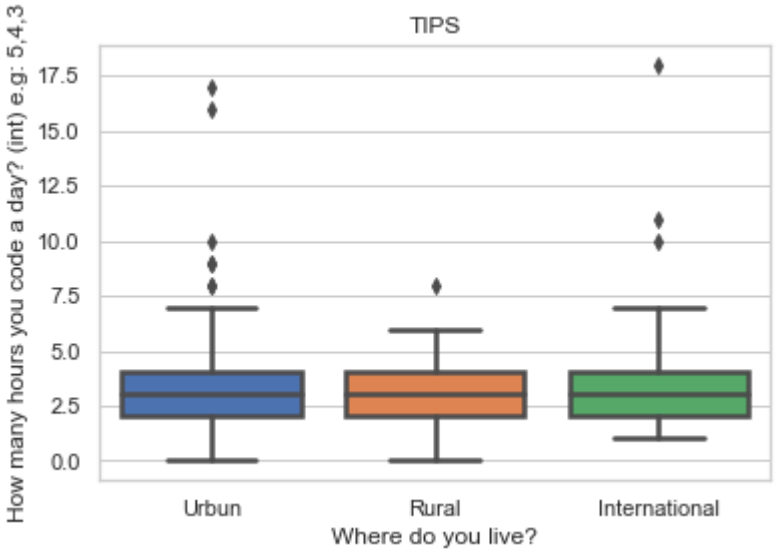
In [146…

```
import seaborn as sns
import matplotlib.pyplot as plt
tip= sns.load_dataset("tips")
sns.boxplot(x='day',y='tip',data=tip,saturation=1, linewidth=2.5)
plt.title('TIPS')
plt.show()
```



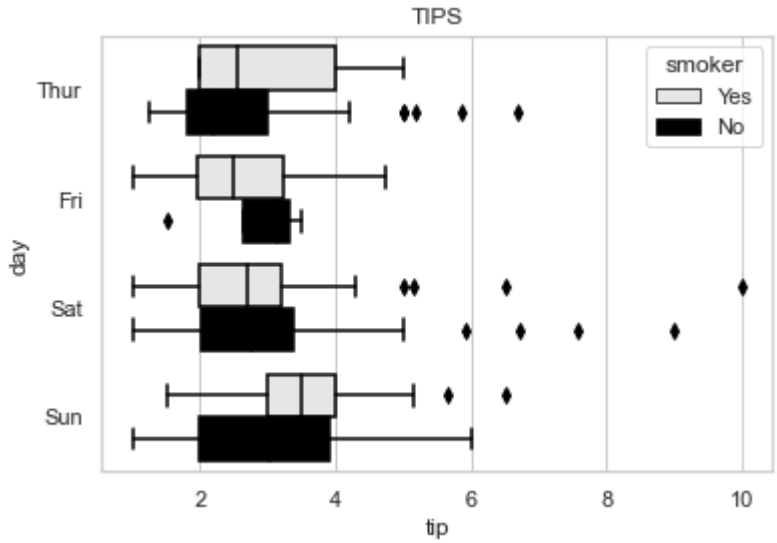
In [147…

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#import data from file
chilla = pd.read_csv("chilla.csv")
sns.boxplot(x='Where do you live?',y='How many hours you code a day? (int) e.g: 5,4,3',
plt.title('TIPS')
plt.show()
```



In [148…

```
import seaborn as sns
import matplotlib.pyplot as plt
tip= sns.load_dataset("tips")
sns.boxplot(x="tip", y="day", saturation= 1, data=tip, orient="h", hue="smoker",
            palette ={"Yes": "0.9", "No":"0"})

plt.title('TIPS')
plt.show()
```



In [149…

```
import seaborn as sns
import matplotlib.pyplot as plt
kashti= sns.load_dataset("titanic")
kashti.head()
```

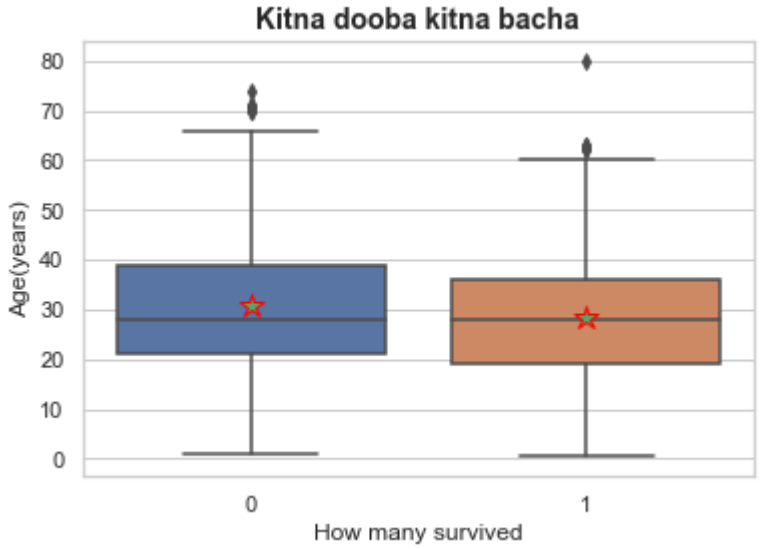
Out[149…

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

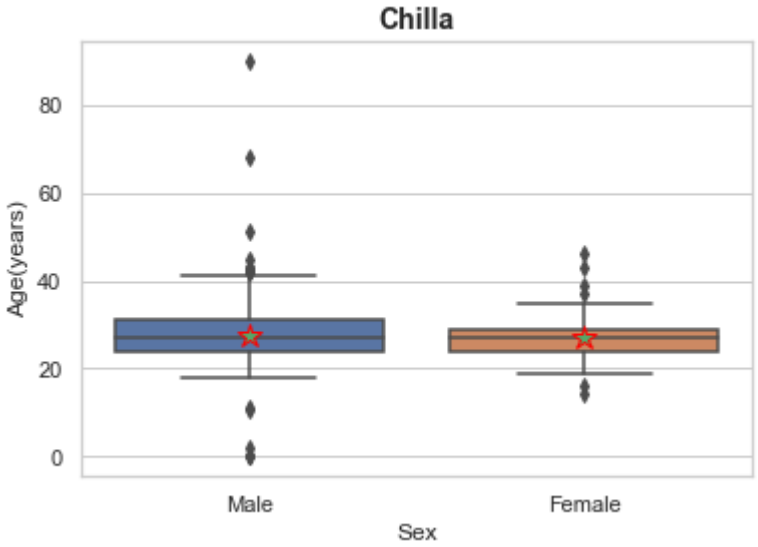
In [150…

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
kashti= sns.load_dataset("titanic")
sns.boxplot(x='survived',y='age',showmeans=True,meanprops=
            {"marker":"*","markersize":"12","markeredgecolor":"red"},data=kashti)
plt.xlabel("How many survived",size=12)
plt.ylabel("Age(years)",size=12)
plt.title("Kitna dooba kitna bacha",size=14,weight='bold')
plt.show()
```



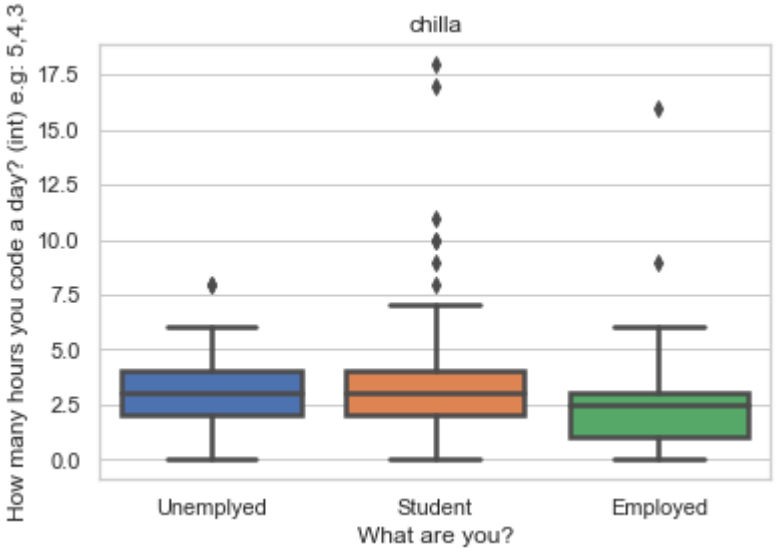
In [151…

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
chilla= pd.read_csv("chilla.csv")
sns.boxplot(x='Gender',y='Age (years)-Float/Int',showmeans=True,meanprops=
            {"marker":"*","markersize":"12","markeredgecolor":"red"},data=chilla)
plt.xlabel("Sex",size=12)
plt.ylabel("Age(years)",size=12)
plt.title("Chilla",size=14,weight='bold')
plt.show()
```



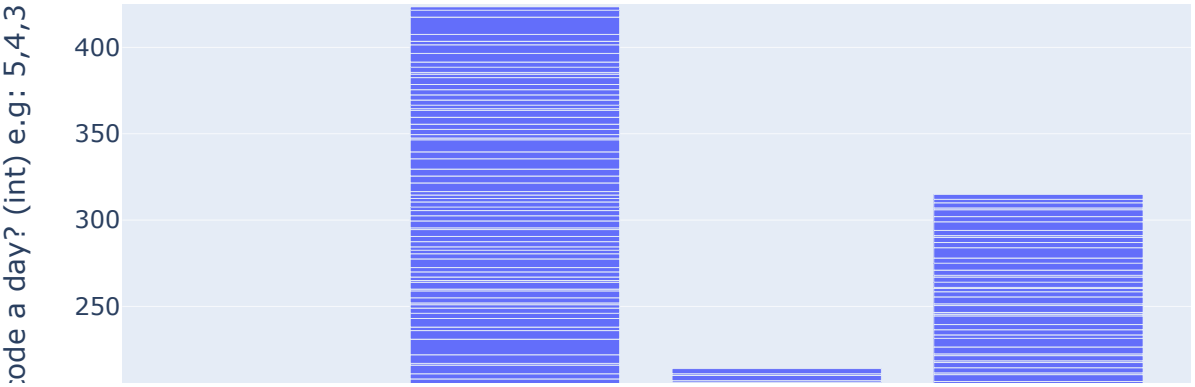
In [152…

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
chilla= pd.read_csv("chilla.csv")
sns.boxplot(x='What are you?',y='How many hours you code a day? (int) e.g: 5,4,3',data=
plt.title('chilla')
plt.show()
```



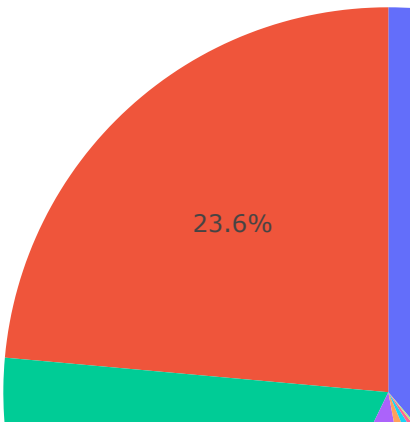
In [153…

```
import plotly.express as px
import seaborn as sns
import pandas as pd
import numpy as np
chilla = pd.read_csv("chilla.csv")
fig = px.bar(chilla, x='Age', y='How many hours you code a day? (int) e.g: 5,4,3')
fig.show()
```



In [154…

```
import plotly.express as px
import seaborn as sns
import pandas as pd
import numpy as np
chilla = pd.read_csv("chilla.csv")
fig = px.pie(chilla, values='Age (years)-Float/Int', names='Blood group ')
fig.show()
```



Arrays

In [155…
```
import numpy as np
a = np.array([1, 2, 3])
a
```

Out[155…
```
array([1, 2, 3])
```

In [156…
```
# An array filled with 0
np.zeros(4)
```

Out[156…
```
array([0., 0., 0., 0.])
```

In [157…
```
# An array filled with 1
np.ones(2)
```

Out[157…
```
array([1., 1.])
```

In [158…
```
# Create an empty array with 2 elements
np.empty(2)
```

Out[158…
```
array([1., 1.])
```

In [159…
```
np.arange(4) # 4 elements
```

Out[159…
```
array([0, 1, 2, 3])
```

In [160…
```
# First and Last element with a step size
np.arange(2, 9, 2)
```

Out[160…
```
array([2, 4, 6, 8])
```

In [161…
```
np.arange(3,18,3)
```

Out[161…
```
array([ 3,  6,  9, 12, 15])
```

In [162…
```
np.linspace(0, 10, num=5) # val are spaced linearly with a specific interval
```

Out[162…
```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

In [163…
```
np.linspace(0, 20, num=10)
```

Out[163…
```
array([ 0.        ,  2.22222222,  4.44444444,  6.66666667,  8.88888889,
       11.11111111, 13.33333333, 15.55555556, 17.77777778, 20.        ])
```

In [164…
```
x = np.ones(2, dtype=np.int64)
x
```

Out[164…
```
array([1, 1], dtype=int64)
```

In [165…
```
x = np.ones(2, dtype=np.float64)
x
```

Out[165…
```
array([1., 1.])
```

In [166…
```
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])
```

In [167…
```
arr
```

Out[167…
```
array([2, 1, 5, 3, 7, 4, 6, 8])
```

In [168…
```
np.sort(arr)
```

Out[168…
```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

In [169…
```
# Concatenation
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6]])
np.concatenate((x, y), axis=0)
```

Out[169…
```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

In [170…
```
import numpy as np
# one dimensional
y = np.array([1, 2, 9])
# Two dimensional
x = np.array([[1, 2, 9], [3, 4, 7],[0, 6, 3]])
```

In [171…
```
y
```

Out[171…
```
array([1, 2, 9])
```

In [172…
```
x
```

Out[172…
```
array([[1, 2, 9],
       [3, 4, 7],
       [0, 6, 3]])
```

In [173…
```
np.zeros((3,4))
```

Out[173…
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [174…
```
np.ones((5,6))
```

Out[174…
```
array([[1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.]])
```

In [175…
```
np.empty((2,3))
```

Out[175…
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [176…
```
n3=np.arange(24).reshape(2,3,4)
n3
```

Out[176…
```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

In [2]:
```
import numpy as np
np.ones(7,dtype=np.int64)
```

Out[2]:
```
array([1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [3]:
```
np.ones(10,dtype=np.float64)
```

Out[3]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

Functions in Array

In [4]:
```
a=np.array([5,1,2,3,7,8,3.2,4.1,6.7])
a
```

Out[4]:
```
array([5. , 1. , 2. , 3. , 7. , 8. , 3.2, 4.1, 6.7])
```

In [5]:
```
b=np.array([4.5,3,3.4,2,1,7,8])
```

```
b
```

Out[5]:
```
array([4.5, 3. , 3.4, 2. , 1. , 7. , 8. ])
```

In [8]:
```
c=np.concatenate((a,b))
c
```

Out[8]:
```
array([5. , 1. , 2. , 3. , 7. , 8. , 3.2, 4.1, 6.7, 4.5, 3. , 3.4, 2. ,
       1. , 7. , 8. ])
```

In [10]:
```
c.sort()
c
```

Out[10]:
```
array([1. , 1. , 2. , 2. , 3. , 3. , 3.2, 3.4, 4.1, 4.5, 5. , 6.7, 7. ,
       7. , 8. , 8. ])
```

2-D Array

In [12]:
```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
a
```

Out[12]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [13]:
```
b=np.array([[12,32,13],[24,45,16],[172,28,39]])
b
```

Out[13]:
```
array([[ 12,  32,  13],
       [ 24,  45,  16],
       [172,  28,  39]])
```

In [14]:
```
c=np.concatenate((a,b),axis=0)
c
```

Out[14]:
```
array([[  1,   2,   3],
       [  4,   5,   6],
       [  7,   8,   9],
       [ 12,  32,  13],
       [ 24,  45,  16],
       [172,  28,  39]])
```

In [15]:
```
c=np.concatenate((a,b),axis=1)
c
```

Out[15]:
```
array([[  1,   2,   3,  12,  32,  13],
       [  4,   5,   6,  24,  45,  16],
       [  7,   8,   9, 172,  28,  39]])
```

In [17]:
```
c.ndim
```

Out[17]:
```
2
```

In [18]:
```
c.size
```

Out[18]:
```
18
```

In [19]:
```
c.shape
```

Out[19]:
```
(3, 6)
```

In [20]:
```
c=np.concatenate((a,b),axis=0)
c
```

Out[20]:
```
array([[  1,   2,   3],
       [  4,   5,   6],
       [  7,   8,   9],
       [ 12,  32,  13],
       [ 24,  45,  16],
       [172,  28,  39]])
```

In [21]:
```
c.shape
```

Out[21]:
```
(6, 3)
```

In [26]:
```
d=np.arange(16) #4*4
d
```

Out[26]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [28]:
```
f=d.reshape(4, 4)
f
```

Out[28]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [33]:
```
# Reshape
np.reshape(f, newshape=(1,16),order='C')
```

Out[33]:
```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15]])
```

In [35]:
```
# Convert 1-D to 2-D
a=np.array([1,2,3,4,5,6,7,8,9])
a
```

Out[35]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [36]:
```
a.shape
```

Out[36]:
```
(9,)
```

In [37]:
```
b=a[np.newaxis,:]
b
```

Out[37]:
```
array([[1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

In [38]:
```
b.shape
```

Out[38]:
```
(1, 9)
```

In [40]:
```
# Column wise
c=a[:,np.newaxis]
c
```

Out[40]:
```
array([[1],
       [2],
       [3],
       [4],
       [5],
       [6],
       [7],
       [8],
       [9]])
```

In [41]:
```
b=a[np.newaxis,:]
b
```

Out[41]:
```
array([[1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

In [42]:
```
a[2:9]
```

Out[42]:
```
array([3, 4, 5, 6, 7, 8, 9])
```

In [43]:
```
a*7
```

Out[43]:
```
array([ 7, 14, 21, 28, 35, 42, 49, 56, 63])
```

In [44]:
```
a+6
```

Out[44]:
```
array([ 7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [46]:
```
a.sum()
```

Out[46]:
```
45
```

In [47]:
```
a.mean()
```

Out[47]:
```
5.0
```

In [48]:
```
# Returns True if any of the elements of a given iterable( List, Dictionary, Tuple, set
a.any()
```

Out[48]:
```
True
```

In [49]:
```
a.all()
```

Out[49]:
```
True
```

In [50]:
```
a.argmax() #returns indices of the max element
```

Out[50]:
```
8
```

In [51]:
```
a
```

Out[51]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [52]:
```
a.argmin()
```

Out[52]:
```
0
```

In [58]:
```
a.argpartition(0)
```

Out[58]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8], dtype=int64)
```

In [56]:
```
a.argsort()
```

Out[56]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8], dtype=int64)
```

In [ ]:
```

```