In [ ]:
```
K.KISHORE KUMAR REDDY
1BY19MCA24
BMSIT&M
PYTHON&DataScience Internship
```

In [ ]:
```
PrintStatements(DAY2)
```

In [3]:
```
print("Hello World")
```

```
Hello World
```

In [4]:
```
print("hello world")
```

```
hello world
```

In [5]:
```
print("hello world")
print("my name is kkkreddy.")
```

```
hello world
my name is kkkreddy.
```

In [6]:
```
print("Hello " + "World")
```

```
Hello World
```

In [7]:
```
print("Hello", "World")
```

```
Hello World
```

In [8]:
```
print(7390)
```

```
7390
```

In [9]:
```
print(600 + 54)
```

```
654
```

In [10]:
```
print("data science is best") # this is not ok!!! print("hello world")
```

```
data science is best
```

In [11]:
```
print(43/56)
```

```
0.7678571428571429
```

In [12]:
```
print(True) # It would be either True or False
```

```
True
```

In [13]:
```
print(True or False)
```

```
True
```

In [14]:
```
print(True and False)
```

```
False
```

In [ ]:
```
Variables
```

In [15]:
```
school = "Dlite"
print(school)
```

```
Dlite
```

In [16]:
```
print(type(school))
```

```
<class 'str'>
```

In [17]:
```
a = 10
print(type(a))
```

```
<class 'int'>
```

In [18]:
```
another_school = "Dlite"
print(id(school))
```

```
2440161134640
```

In [19]:
```
print(id(another_school))
```

```
2440161134640
```

In [20]:
```
another_school = "Elite"
print(id(another_school))
```

```
2440161166512
```

In [21]:
```
print("My School is " + school)
```

```
My School is Dlite
```

In [22]:
```
print("My School is", school)
```

```
My School is Dlite
```

In [23]:
```
roll_no = 54
print(type(roll_no))
```

```
<class 'int'>
```

In [24]:
```
roll_no = 54/3
print(type(roll_no))
```

```
<class 'float'>
```

In [25]:
```
isGood = True # 0, 1 Binary
print(type(isGood))
```

```
<class 'bool'>
```

In [26]:
```
my_name = input("Please type your name -> ")
```

```
Please type your name -> kishore
```

In [27]:
```
print(my_name)
```

```
kishore
```

In [28]:
```
my_name
```

Out[28]:
```
'kishore'
```

In [29]:
```
my_age = input("Please enter your age ")
```

```
Please enter your age 10
```

In [30]:
```
fathers_age = my_age * 2
print("My fathers age is " + fathers_age)
```

```
My fathers age is 1010
```

In [31]:
```
print(type(my_age))
```

```
<class 'str'>
```

In [32]:
```
my_age_int = int(my_age)
```

```
print(type(my_age_int))
```

```
<class 'int'>
```

In [33]:
```
fathers_age_int = my_age_int * 2
print("My fathers age is " + str(fathers_age_int))
```

```
My fathers age is 20
```

In [ ]:
```
Strings
```

In [34]:
```
topic = "Long text with no meaning in it."
print(topic)
```

```
Long text with no meaning in it.
```

In [35]:
```
print(topic[0])
```

```
L
```

In [36]:
```
print(topic[10])
```

```
w
```

In [37]:
```
topic = "123123313 imp data"
print(topic[9:])
```

```
 imp data
```

In [38]:
```
print(topic[-1])
```

```
a
```

In [39]:
```
print(topic[:-1])
```

```
123123313 imp dat
```

In [40]:
```
print(topic[9:13])
```

```
 imp
```

In [41]:
```
topic = "Long text with no meaning in it."
print(topic.lower())
```

```
long text with no meaning in it.
```

In [42]:
```
print(topic.upper())
```

```
LONG TEXT WITH NO MEANING IN IT.
```

In [46]:
```
print(topic.find("with"))
```

```
10
```

In [47]:
```
print(topic.find("donthave"))
```

```
-1
```

In [48]:
```
print(topic.replace("it", "this"))
```

```
Long text wthish no meaning in this.
```

In [ ]:
```
DAY3
```

In [ ]:
```
if statements(conditional statements)
```

In [49]:
```
my_money = 10
diary_milk = 40
```

```
munch = 5
if my_money >= 40:
  print("I will buy it")
elif my_money >= munch:
  print("I go with munch")
else:
  print("I will buy some other day!")
```

```
I go with munch
```

In [ ]:
```
Looping statements(for,while)
```

In [50]:
```
print(1)
print(2)
print(3)
```

```
1
2
3
```

In [51]:
```
for i in range(10):
  print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [ ]:
```
range?
```

In [52]:
```
for i in range(5):
  print(i, i**2)
```

```
0 0
1 1
2 4
3 9
4 16
```

In [53]:
```
for i in range(0,20,2):
  print(i)
```

```
0
2
4
6
8
10
12
14
16
18
```

In [54]:
```
for i in range(1,20,2):
  print(i)
```

```
1
3
5
7
9
11
```

```
13
15
17
19
```

In [ ]:
```
Fibonacci Series
```

In [55]:
```
# 1
# 1
# 2
# 3
# 5
# for
a = 1
b = 1
print(a)
print(b)
for i in range(10):
  temp = a + b
  a = b
  b = temp
  print(temp)
```

```
1
1
2
3
5
8
13
21
34
55
89
144
```

In [56]:
```
# While
a = 1
b = 1
print(a)
print(b)
while b < 50:
  temp = a + b
  a = b
  b = temp
  print(temp)
print(b)
```

```
1
1
2
3
5
8
13
21
34
55
55
```

In [57]:
```
# how to write a function
def fibonacci(pos):
  a = 1
  b = 1
  for i in range(pos):
    temp = a + b
    a = b
```

```
    b = temp
  return temp
print(fibonacci(10))
```

```
144
```

In [58]:
```
def fibonacci(pos, a, b):
  for i in range(pos):
    temp = a + b
    a = b
    b = temp
  return temp
fibonacci(10, 1, 1)
```

Out[58]:
```
144
```

In [59]:
```
fibonacci(2, 13, 21)
```

Out[59]:
```
55
```

In [60]:
```
def fibonacci(pos, a=1, b=1):
  for i in range(pos):
    temp = a + b
    a = b
    b = temp
  return temp
fibonacci(10)
```

Out[60]:
```
144
```

In [61]:
```
fibonacci(2, 13, 21)
```

Out[61]:
```
55
```

In [ ]:
```
# Recursive Function
def fibonacci_recursive(pos, a=1, b=2):
  if pos > 1:
    return fibonacci_recursive(pos-1, b, a + b)
  else:
    return a + b
fibonacci_recursive(1000000)
```

In [ ]:
```
# callstack
# 1
# 1
# 1
# 1
# 1
# 1
# 1
# 1
# 1
```

In [ ]:
```
# Lamdba
# Syntax:  lambda arguments: expression
```

In [1]:
```
print(doubdouble = lambda
      x: x * 2
```

```
  File "<ipython-input-1-f474027291a8>", line 2
    x: x * 2
```

```
                ^
SyntaxError: unexpected EOF while parsing
```

In [2]:
```
double = lambda x: x * 2
print(double)
```

```
<function <lambda> at 0x0000020B9A4DE1F0>
```

In [3]:
```
print(double(10))
```

```
20
```

In [ ]:
```
DAY4
```

In [ ]:
```
Programs based on the lerarning of basic of python
```

In [ ]:
```
Problem 1

Prompt the user "Enter your weight (in kgs)" and record weight

Prompt the user "What is your preferred unit of height? Type "F" for feet and "M" fo

If user says "F" then prompt user "You will enter your height given as feet and inch

If user says "M" then prompt user "What is your height in meters" and record height

If user had chosen "F" then convert height into meters

Compute BMI using the following formula

BMI = weight / height ** 2

Depending on the value of BMI, report the user's type given by the following table:
```

In [4]:
```
# prompt the user about the weight
weight = input("Enter your weight (in kgs")
height_unit = input("What is your preferred unit of height")

if height_unit == "F":
  print("I am in the F Block.")
else:
  print("I am in the M Block.")
```

```
Enter your weight (in kgs12
What is your preferred unit of heightf
I am in the M Block.
```

In [5]:
```
# prompt the user about the weight
weight = input("Enter your weight (in kgs")
height_unit = input("What is your preferred unit of height")

if height_unit == "F":
  feet = input("Enter feet and inches. Lets start with feet")
  inches = input("Now enter inches")
else:
  meters = input("Please enter your height in meters")
```

```
Enter your weight (in kgs12
What is your preferred unit of height23
Please enter your height in meters23
```

In [6]:
```
# prompt the user about the weight
weight = input("Enter your weight (in kgs")
height_unit = input("What is your preferred unit of height")
```

```
if height_unit == "F":
  feet = input("Enter feet and inches. Lets start with feet")
  inches = input("Now enter inches")
  meters = (feet + inches/12) * 0.3048
else:
  meters = input("Please enter your height in meters")
```

```
Enter your weight (in kgs12
What is your preferred unit of heightF
Enter feet and inches. Lets start with feet6
Now enter inches3
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-978ec0c93202> in <module>
      6   feet = input("Enter feet and inches. Lets start with feet")
      7   inches = input("Now enter inches")
----> 8   meters = (feet + inches/12) * 0.3048
      9 else:
     10   meters = input("Please enter your height in meters")

TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

In [8]:
```
def input_single_integer(prompt):
  ret = int(input(prompt))
  return ret
def input_single_float(prompt):
  ret = float(input(prompt))
  return ret
# prompt the user about the weight
weight = input_single_float("Enter your weight (in kgs")
height_unit = input("What is your preferred unit of height")

if height_unit == "F":
  feet = input_single_integer("Enter feet and inches. Lets start with feet")
  inches = input_single_integer("Now enter inches")
  meters = (feet + inches/12) * 0.3048
else:
  meters = input_single_float("Please enter your height in meters")
```

```
Enter your weight (in kgs60
What is your preferred unit of height5
Please enter your height in meters4
```

In [9]:
```
# prompt the user about the weight
weight = input_single_float("Enter your weight (in kgs")
height_unit = input("What is your preferred unit of height")

if height_unit == "F":
  feet = input_single_integer("Enter feet and inches. Lets start with feet")
  inches = input_single_integer("Now enter inches")
  meters = (feet + inches/12) * 0.3048
else:
  meters = input_single_float("Please enter your height in meters")

height = meters
bmi = weight / height ** 2

print("My BMI value is ", str(bmi))

if bmi < 18.5:
  print("UNDERWEIGHT")
elif bmi < 25:
  print("NORMAL")
elif bmi < 30:
  print("OVERWEIGHT")
```

```
else:
  print("VERY-OVERWEIGHT")
```

```
Enter your weight (in kgs60
What is your preferred unit of heightF
Enter feet and inches. Lets start with feet5
Now enter inches4
My BMI value is  22.705123535247072
NORMAL
```

In [ ]:
```
DAY5
```

In [ ]:
```
1.DOC STRING
2.Eeeoe handling (Try---Except Block)
```

In [10]:
```
def fibo_seq(n, a = 1, b = 1):
  """
    This computes n'th number in
    fibonacci sequence starting with
    a and b
  """
  for i in range(n):
    temp = a + b
    a = b
    b = temp
  return temp
fibo_seq?
def fibo_seq(n, a = 1, b = 1):
  """
    This computes n'th number in
    fibonacci sequence starting with
    a and b.
    Value of a and b is default set to 1
  """
  for i in range(n):
    temp = a + b
    a = b
    b = temp
  return temp
fibo_seq?
def input_single_float(prompt):
  """
    This function will return an floating
    point Number.
  """
  user_input = input(prompt)
  converted_input = float(user_input)
  return converted_input
input_single_float?
print(input_single_float("Enter float no"))
```

```
Enter float no4.5
4.5
```

In [11]:
```
print(input_single_float("Enter float no"))
```

```
Enter float no4.5
4.5
```

In [12]:
```
def input_single_float(prompt):
  """
    This function will return an floating
    point Number.
  """
  user_input = input(prompt)
```

```
  try:
    converted_input = float(user_input)
  except:
    print("Please enter float")
  return converted_input
print(input_single_float("enter the float"))
```

```
enter the float4.5
4.5
```

In [13]:
```
print(input_single_float("enter the float"))
```

```
enter the floatkishore
Please enter float
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
<ipython-input-13-a1a7609eaf4b> in <module>
----> 1 print(input_single_float("enter the float"))

<ipython-input-12-bb94447ed1f0> in input_single_float(prompt)
      9   except:
     10     print("Please enter float")
---> 11   return converted_input
     12 print(input_single_float("enter the float"))

UnboundLocalError: local variable 'converted_input' referenced before assignment
```

In [14]:
```
def input_single_float(prompt):
  """
    This function will return an floating
    point Number.
  """
  user_input = input(prompt)
  try:
    converted_input = float(user_input)
  except:
    print("Please enter float")
    converted_input = None
  return converted_input
print(type(input_single_float("enter the float")))
```

```
enter the floatkishore
Please enter float
<class 'NoneType'>
```

In [ ]:
```
List
```

In [15]:
```
course_name = "Foundations of data science"
words = course_name.split()
print(words)
```

```
['Foundations', 'of', 'data', 'science']
```

In [16]:
```
print(type(words))
```

```
<class 'list'>
```

In [17]:
```
print(words[0])
```

```
Foundations
```

In [18]:
```
print(type(words[0]))
```

```
<class 'str'>
```

In [19]:
```
print(words[-1])
```

```
science
```

In [20]:
```
print(words[0].lower())
```

```
foundations
```

In [21]:
```
print(words[2:])
```

```
['data', 'science']
```

In [22]:
```
print(words[-2])
```

```
data
```

In [23]:
```
print(words[2:4])
```

```
['data', 'science']
```

In [24]:
```
new_words = ["machine", "learning"]
```

In [25]:
```
print(type(new_words))
```

```
<class 'list'>
```

In [27]:
```
fibo_list = [1, 1, 2, 3, 5, 8, 13, 21]
type(fibo_list)
```

Out[27]:
```
list
```

In [28]:
```
fibo_list
```

Out[28]:
```
[1, 1, 2, 3, 5, 8, 13, 21]
```

In [29]:
```
fibo_list[::-1]
```

Out[29]:
```
[21, 13, 8, 5, 3, 2, 1, 1]
```

In [ ]:
```
# Iteration
```

In [30]:
```
# Method 1
for i in fibo_list:
  print(i)
```

```
1
1
2
3
5
8
13
21
```

In [31]:
```
# Method 2
length = len(fibo_list)
print("Length", length)

for i in range(length):
  print(i)
```

```
Length 8
0
1
2
3
4
5
```

```
6
7
```

In [32]:
```
# Method 2
length = len(fibo_list)
print("Length", length)

for i in range(length):
  print(i, fibo_list[i])
```

```
Length 8
0 1
1 1
2 2
3 3
4 5
5 8
6 13
7 21
```

In [33]:
```
# Method 3
for index, value in enumerate(fibo_list):
  print(index, value)
```

```
0 1
1 1
2 2
3 3
4 5
5 8
6 13
7 21
```

In [37]:
```
bmi_record = ["David", 75, 1.73, False]
for item in bmi_record:
  print(item)
```

```
David
75
1.73
False
```

In [39]:
```
bmi_record[0] = "Not my name"
bmi_record
```

Out[39]:
```
['Not my name', 75, 1.73, False]
```

In [40]:
```
bmi_record.append("david@somedomain.com")
bmi_record
```

Out[40]:
```
['Not my name', 75, 1.73, False, 'david@somedomain.com']
```

In [42]:
```
bmi_record.insert(2, "Indian")
list.insert?
bmi_record
```

Out[42]:
```
['Not my name', 75, 'Indian', 'Indian', 1.73, False, 'david@somedomain.com']
```

In [ ]:
```
DAY6
```

In [ ]:
```
1.Tuple
2.Set
3.Dictionary
```

In [43]:
```
bmi_categories = ("Underweight", "Normal", "Overweight", "Very Overweight")
```

```
print(type(bmi_categories))
```

```
<class 'tuple'>
```

In [44]:
```
print(bmi_categories[0])
```

```
Underweight
```

In [45]:
```
print(bmi_categories[-1])
```

```
Very Overweight
```

In [46]:
```
print(bmi_categories[0:2])
```

```
('Underweight', 'Normal')
```

In [47]:
```
print(bmi_categories.index("Normal"))
```

```
1
```

In [48]:
```
print(bmi_categories.index("Very Overweight"))
```

```
3
```

In [49]:
```
print("Very Underweight" in bmi_categories)
```

```
False
```

In [50]:
```
%%timeit -n1 -r10
for i in range(1000000):
  my_list = ["hello1", "hello2", "hello3", "hello4", "hello5"]
```

```
129 ms ± 14.1 ms per loop (mean ± std. dev. of 10 runs, 1 loop each)
```

In [51]:
```
%%timeit?
%%timeit -n1 -r10
for i in range(1000000):
  my_tuple = ("hello1", "hello2", "hello3", "hello4", "hello5")
```

```
UsageError: Line magic function `%%timeit` not found.
```

In [ ]:
```
Set
```

In [53]:
```
batsmen = {"Rohit", "Virat", "Rahul", "Ravindra"}
print(type(batsmen))
```

```
<class 'set'>
```

In [54]:
```
print("Bumrah" in batsmen)
```

```
False
```

In [55]:
```
print("Virat" in batsmen)
```

```
True
```

In [56]:
```
print(batsmen.index("Rohit"))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-56-2031d29ef3f1> in <module>
----> 1 print(batsmen.index("Rohit"))

AttributeError: 'set' object has no attribute 'index'
```

In [57]:
```
bowlers = {"Ishant", "Bumrah", "Ravindra"}
print(type(bowlers))
```

```
<class 'set'>
```

In [59]:
```
allrounders = batsmen.intersection(bowlers)
print(allrounders)
```

```
{'Ravindra'}
```

In [60]:
```
players = batsmen.union(bowlers)
print(players)
```

```
{'Bumrah', 'Virat', 'Ravindra', 'Rahul', 'Ishant', 'Rohit'}
```

In [61]:
```
print(type(players))
```

```
<class 'set'>
```

In [62]:
```
bowlers = {"Ishant", "Bumrah", "Ravindra", "Bumrah"}
bowlers
```

Out[62]:
```
{'Bumrah', 'Ishant', 'Ravindra'}
```

In [ ]:
```
### Search Benchmarking
```

In [63]:
```
%%timeit -n1 -r10
my_list = list(range(10000))
for i in range(10000):
  b = 1947 in my_list
```

```
221 ms ± 20.3 ms per loop (mean ± std. dev. of 10 runs, 1 loop each)
```

In [64]:
```
%%timeit -n1 -r10
my_tuple = tuple(range(10000))
for i in range(10000):
  b = 1947 in my_tuple
```

```
207 ms ± 20.8 ms per loop (mean ± std. dev. of 10 runs, 1 loop each)
```

In [65]:
```
%%timeit -n1 -r10
my_set = set(range(10000))
for i in range(10000):
  b = 1947 in my_set
```

```
2.43 ms ± 456 µs per loop (mean ± std. dev. of 10 runs, 1 loop each)
```

In [ ]:
```
# Dictionary
```

In [67]:
```
labels = ("name", "weight", "height", "is_overweight")
bmi_dataset = [
               ["Krishna", 75, 1.73, False],
               ["David", 120, 1.78, True]
]
bmi_record = {
    "name" : "Krishna",
    "weight" : 75,
    "height" : 1.73,
    "is_overweight" : False
}
bmi_record["name"]
```

Out[67]:
```
'Krishna'
```

In [68]:
```
bmi_record["height"]
```

Out[68]:
```
1.73
```

In [69]:
```
bmi_record["email"] = "Krishna@mail.com"
```

```
bmi_record
```

Out[69]:
```
{'name': 'Krishna',
 'weight': 75,
 'height': 1.73,
 'is_overweight': False,
 'email': 'Krishna@mail.com'}
```

In [70]:
```
for key in bmi_record:
  print(bmi_record[key])
```

```
Krishna
75
1.73
False
Krishna@mail.com
```

In [71]:
```
for key in bmi_record:
  print(key, ":", bmi_record[key])
```

```
name : Krishna
weight : 75
height : 1.73
is_overweight : False
email : Krishna@mail.com
```

In [72]:
```
print(bmi_record.keys())
```

```
dict_keys(['name', 'weight', 'height', 'is_overweight', 'email'])
```

In [73]:
```
print(type(list(bmi_record.keys())))
```

```
<class 'list'>
```

In [74]:
```
bmi_record = {
    "name" : "Krishna",
    "weight" : 75,
    "height" : 1.73,
    "is_overweight" : False
}
new_bmi_record = {
    "name" : "David",
    "weight" : 120,
    "height" : 1.78,
    "is_overweight" : True
}
new_bmi_record
```

Out[74]:
```
{'name': 'David', 'weight': 120, 'height': 1.78, 'is_overweight': True}
```

In [75]:
```
records = [bmi_record, new_bmi_record]
records
```

Out[75]:
```
[{'name': 'Krishna', 'weight': 75, 'height': 1.73, 'is_overweight': False},
 {'name': 'David', 'weight': 120, 'height': 1.78, 'is_overweight': True}]
```

In [76]:
```
type(records[0]["height"])
```

Out[76]:
```
float
```

In [77]:
```
records[0]["height"]
```

Out[77]:
```
1.73
```

In [78]:
```
records[0]["name"]
```

Out[78]:
```
'Krishna'
```

In [ ]:
```
DAY7

1.Questions based on Lists
2.Questions based on Dictionary
```

In [79]:
```
aList = []
for i in range(10):
  aList.append(i)
print(aList)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [80]:
```
# single line for loop
aList = [i for i in range(10)]
print(aList)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [81]:
```
print([i for i in range(10)])
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [82]:
```
aList = []
for i in range(10):
  if i%2 == 0:
    aList.append(i)
print(aList)
```

```
[0, 2, 4, 6, 8]
```

In [83]:
```
print([i for i in range(10) if i%2 == 0])
```

```
[0, 2, 4, 6, 8]
```

In [84]:
```
print([i+2 for i in range(10)])
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

In [85]:
```
print([for i in range(10)])
```

```
  File "<ipython-input-85-fba72834332c>", line 1
    print([for i in range(10)])
          ^
SyntaxError: invalid syntax
```

In [86]:
```
for i in range(10):
```

```
  File "<ipython-input-86-e61162350cc4>", line 1
    for i in range(10):
                       ^
SyntaxError: unexpected EOF while parsing
```

In [ ]:
```
# List Questions
```

In [87]:
```
# question 1
aList = [100, 200, 300, 400, 500]
aList = aList[::-1]
print(aList)
```

```
[500, 400, 300, 200, 100]
```

In [88]:
```
# question 2
list1 = ["M", "na", "I", "Ke"]
list2 = ["y", "me", "s", "lly"]
```

```
list3 = [i+j for i, j in zip(list1, list2)]
zip?
# question 3
aList = [1,2,3,4,5,6,7]
new_list = [i*i for i in aList]
print(new_list)
```

```
[1, 4, 9, 16, 25, 36, 49]
```

In [89]:
```
# question 4
list1 = ["Hello ", "take "]
list2 = ["Dear", "Sir"]

list4 = []
for x in list1:
  for y in list2:
    list4.append(x+y)

list3 = [x+y for x in list1 for y in list2]
print(list3)
```

```
['Hello Dear', 'Hello Sir', 'take Dear', 'take Sir']
```

In [90]:
```
# question 5
list1 = ["Mike", "", "Emma", "Kelly"
, "", "Brad"]
result_list = list(filter(None, list1))
print(result_list)
```

```
['Mike', 'Emma', 'Kelly', 'Brad']
```

In [ ]:
```
# Dictionary Questions
```

In [91]:
```
# Question 1
keys = ['Ten', 'Twenty', 'Thirty']
values = [10, 20, 30]
my_dict = dict(zip(keys, values))
print(my_dict)
```

```
{'Ten': 10, 'Twenty': 20, 'Thirty': 30}
```

In [92]:
```
dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}
dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
dict3 = {**dict1, **dict2}
print(dict3)
```

```
{'Ten': 10, 'Twenty': 20, 'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
```

In [94]:
```
# question 3
sampleDict={
    "class" :{
        "student" :{
            "name" : "Mike",
            "marks" : {
                "physics":70,
                "history":80
            }
        }
    }
}
print(sampleDict['class']['student']['marks']['history'])
```

```
80
```

In [95]:
```
# question 4
sampleDict = {
    "name" : "Kelly",
```

```
    "age":25,
    "salary":8000,
    "city" : "New York"
}
keys = ["name", "salary"]
newDict = {i:sampleDict[i] for i in keys}
print(newDict)
```

```
{'name': 'Kelly', 'salary': 8000}
```

In [96]:
```
sampleDict.keys()
```

Out[96]:
```
dict_keys(['name', 'age', 'salary', 'city'])
```

In [98]:
```
# question 5
sampleDict = {'a' : 100, 'b' : 200, 'c' : 300}
print(200 in sampleDict.values())
```

```
True
```

In [99]:
```
# question 6
sampleDict = {
    "name" : "kelly",
    "age" : 25,
    "salary" : 8000,
    "city" : "New York"
}
sampleDict['location'] = sampleDict.pop('city')
sampleDict
```

Out[99]:
```
{'name': 'kelly', 'age': 25, 'salary': 8000, 'location': 'New York'}
```

In [100…
```
# question 7
sampleDict = {
        'Physics': 82,
        'Math': 65,
        'history': 75
}
print(min(sampleDict, key=sampleDict.get))
```

```
Math
```

In [ ]:
```
DAY8&9
```

In [ ]:
```
1.Numpy Operations
2.Cricket data Set
```

In [101…
```
import numpy as np
# 2-D Array
arr2d = np.array([
                  [1, 2, 3],
                  [4, 5, 6]
])
arr2d1 = np.array([[1,2,3],[4,5,6]])
arr2d
```

Out[101…
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [102…
```
arr2d.ndim
```

Out[102…
```
2
```

```
arr2d.shape
```

In [103…

Out[103… (2, 3)

In [104…
```
arr2d.size
```

Out[104… 6

In [105…
```
# 3-D Array
arr3d = np.array([
                  [
                   [1,2,3],
                   [4,5,6]
                  ],
                  [
                   [7,8,9],
                   [10, 11, 12]
                  ]
])
arr3d
```

Out[105…
```
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

       [[ 7,  8,  9],
        [10, 11, 12]]])
```

In [106…
```
arr3d.shape
```

Out[106… (2, 2, 3)

In [107…
```
arr3d.size
```

Out[107… 12

In [ ]:
```
#Numpy Operations
```

In [108…
```
np.ones((2, 3))
```

Out[108…
```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

In [109…
```
120 * np.ones((2, 3))
```

Out[109…
```
array([[120., 120., 120.],
       [120., 120., 120.]])
```

In [110…
```
np.zeros((2, 3))
```

Out[110…
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [111…
```
# Random
np.random.randn(2, 3)
```

Out[111…
```
array([[ 0.45812998, -0.20921892, -0.01871416],
       [-0.74163859,  0.26025767,  0.8953582 ]])
```

In [112…
```
np.random.rand(2, 3)
```

Out[112…
```
array([[0.91186501, 0.02505894, 0.79951714],
       [0.70898653, 0.38568505, 0.02636636]])
```

In [113…
```
np.random.randint(0, 100, (2, 3))
```

Out[113…
```
array([[33, 96, 86],
       [67, 29, 92]])
```

In [114…
```
np.arange(10, 100, 2)
```

Out[114…
```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76,
       78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98])
```

In [115…
```
np.linspace(7, 70, 10)
```

Out[115…
```
array([ 7., 14., 21., 28., 35., 42., 49., 56., 63., 70.])
```

In [117…
```
str_arr = np.array(['1.1', '2.2', '3.3', '4.4'])
str_arr
```

Out[117…
```
array(['1.1', '2.2', '3.3', '4.4'], dtype='<U3')
```

In [118…
```
# conversion
arr = np.array(str_arr, dtype='float')
arr
```

Out[118…
```
array([1.1, 2.2, 3.3, 4.4])
```

In [119…
```
arr1 = np.random.rand(3,4)
arr2 = np.random.rand(3,4)
arr1
```

Out[119…
```
array([[0.16987265, 0.15873221, 0.22828483, 0.06448998],
       [0.25078844, 0.01384643, 0.35777322, 0.4769669 ],
       [0.28425567, 0.68397151, 0.38495341, 0.66877924]])
```

In [120…
```
arr2
```

Out[120…
```
array([[0.55528001, 0.81521227, 0.61076215, 0.51333864],
       [0.48640662, 0.67294611, 0.18610492, 0.40679511],
       [0.53562192, 0.16614845, 0.60215246, 0.63982478]])
```

In [121…
```
arr1 + arr2
```

Out[121…
```
array([[0.72515267, 0.97394448, 0.83904697, 0.57782862],
       [0.73719506, 0.68679254, 0.54387814, 0.88376201],
       [0.8198776 , 0.85011996, 0.98710587, 1.30860401]])
```

In [122…
```
arr1 - arr2
```

Out[122…
```
array([[-0.38540736, -0.65648005, -0.38247732, -0.44884867],
       [-0.23561818, -0.65909968,  0.1716683 ,  0.07017179],
       [-0.25136625,  0.51782306, -0.21719905,  0.02895446]])
```

In [123…
```
arr1 * arr2
```

Out[123…
```
array([[0.09432689, 0.12940045, 0.13942773, 0.0331052 ],
       [0.12198516, 0.0093179 , 0.06658336, 0.1940278 ],
       [0.15225357, 0.11364081, 0.23180065, 0.42790153]])
```

In [124…
```
arr1 / arr2
```

Out[124…
```
array([[0.30592251, 0.19471274, 0.37377042, 0.12562852],
       [0.51559422, 0.02057584, 1.92242753, 1.1724991 ],
       [0.53070209, 4.11662879, 0.63929559, 1.04525374]])
```

In [125…
```
np.exp(arr1)
```

Out[125…
```
array([[1.18515392, 1.17202405, 1.25644314, 1.06661489],
       [1.28503819, 1.01394274, 1.43014126, 1.61118012],
       [1.32877262, 1.98173259, 1.46954586, 1.95185312]])
```

In [126…
```
np.log(arr1)
```

Out[126…
```
array([[-1.77270622, -1.84053669, -1.47716119, -2.74124547],
       [-1.38314557, -4.27972788, -1.02785595, -0.74030818],
       [-1.25788119, -0.37983901, -0.95463296, -0.40230126]])
```

In [127…
```
np.log(np.exp(arr1))
```

Out[127…
```
array([[0.16987265, 0.15873221, 0.22828483, 0.06448998],
       [0.25078844, 0.01384643, 0.35777322, 0.4769669 ],
       [0.28425567, 0.68397151, 0.38495341, 0.66877924]])
```

In [128…
```
np.sqrt(arr1)
```

Out[128…
```
array([[0.4121561 , 0.39841212, 0.47779161, 0.25394877],
       [0.50078782, 0.11767085, 0.59814147, 0.6906279 ],
       [0.53315633, 0.8270257 , 0.62044614, 0.81778924]])
```

In [150…
```
pwd
```

Out[150…
```
'C:\\Users\\Kesam\\Downloads\\Kishorereddy'
```

In [151…
```
import pandas as pd
```

In [153…
```
data=pd.read_table(r'C:\Users\Kesam\Downloads\Kishorereddy\cric_data.tsv')
data
```

Out[153…

	Unnamed: 0	Sachin Tendulkar	Rahul Dravid	India
0	0	100	78	342
1	1	11	62	191
2	2	8	85	252
3	3	71	24	307
4	4	104	17	229
…	…	…	…	…
220	220	25	14	101
221	221	102	50	470
222	222	0	22	165
223	223	27	0	192
224	224	40	0	213

225 rows × 4 columns

In [ ]:
```
Question 1: Find the mean, median for Sachin, Rahul, India
```

In [155…
```
cric_data = np.loadtxt("cric_data.tsv", skiprows=1)
cric_data.shape
```

Out[155… (225, 4)

In [156…
```
cric_data = cric_data[:, [1, 2, 3]]
cric_data.shape
```

Out[156… (225, 3)

In [157…
```
sachin = cric_data[:, 0]
rahul = cric_data[:, 1]
india = cric_data[:, 2]
sachin.shape
```

Out[157… (225,)

In [158…
```
def stats(col):
  print('Mean', np.mean(col))
  print('Median', np.median(col))
```

```
  File "<ipython-input-158-73199a90d6b1>", line 4
    stats(sachin)
    ^
IndentationError: unexpected indent
```

In [159…
```
    stats(sachin)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-159-74eeb21933a5> in <module>
----> 1 stats(sachin)

NameError: name 'stats' is not defined
```

In [ ]:
```
 Question 2: Find histogram of Sachin's scores with 10 bins
```

In [160…
```
np.histogram(sachin)
```

Out[160… (array([99, 36, 28, 16, 11, 17, 8, 8, 1, 1], dtype=int64),
 array([0. , 18.6, 37.2, 55.8, 74.4, 93. , 111.6, 130.2, 148.8,
        167.4, 186.]))

In [ ]:
```
Question 3: find mean of sachin's scores grouped by 25 matches
```

In [161…
```
sachin.shape
```

Out[161… (225,)

In [163…
```
sachin25 = sachin.reshape(9, 25)
sachin25.shape
```

Out[163… (9, 25)

In [164…
```
np.mean(sachin25, axis=1)
```

Out[164… array([33.96, 49.4 , 38.48, 40.16, 39.36, 38.2 , 44.6 , 39.52, 35.2])

In [ ]:
```
Question 4: Find mean of Sachin's scores where he has scored a century
```

In [165…
```
sachin >= 100
```

Out[165…
```
array([ True, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False,  True, False, False, False, False, False,
       False,  True, False, False, False, False, False, False,  True,
       False, False,  True, False, False, False, False, False, False,
       False, False,  True,  True, False, False, False, False, False,
       False,  True, False, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False,  True,  True,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False,  True, False,  True, False, False,
       False,  True,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False,  True, False,  True, False, False, False, False, False,
       False, False, False, False, False,  True, False, False, False])
```

In [166…
```
sachin[sachin >= 100]
```

Out[166…
```
array([100., 104., 138., 141., 186., 146., 141., 123., 120., 122., 100.,
       152., 105., 122., 100., 117., 141., 139., 114., 127., 110., 146.,
       101., 140., 113., 102.])
```

In [167…
```
np.mean(sachin[sachin >= 100])
```

Out[167…
```
125.0
```

In [ ]:
```
Question 5: Find mean of Sachin's scores when Rahul has scored less than 10
```

In [168…
```
rahul <= 10
```

Out[168…
```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True,  True, False,  True,  True,
       False, False,  True, False, False,  True, False, False,  True,
        True, False,  True, False,  True, False, False, False, False,
       False, False, False,  True,  True,  True,  True, False, False,
       False, False, False, False,  True, False, False, False, False,
        True, False, False, False, False, False, False, False,  True,
        True, False,  True,  True, False,  True, False, False,  True,
       False, False, False, False, False, False, False,  True,  True,
       False, False,  True, False,  True, False, False, False, False,
        True,  True,  True, False,  True, False, False, False, False,
       False,  True, False, False, False, False, False, False,  True,
        True, False,  True,  True, False,  True, False, False,  True,
        True, False, False, False, False,  True, False, False, False,
        True, False, False,  True,  True, False,  True, False,  True,
        True,  True,  True, False,  True, False, False, False, False,
       False, False, False, False, False, False,  True,  True,  True,
        True, False, False, False, False, False, False, False, False,
       False, False, False,  True, False, False, False,  True,  True,
       False,  True,  True,  True, False, False, False,  True, False,
```

```
        True,  True,  True,  True, False, False, False,  True, False,
       False, False, False,  True, False, False, False, False, False,
       False, False, False,  True,  True, False, False, False, False,
       False, False, False, False, False, False, False, False,  True,
        True, False, False, False, False, False, False,  True,  True])
```

In [169…
```
sachin[rahul <= 10]
```

Out[169…
```
array([ 37.,  14.,   4.,   0.,  62.,  38.,  65.,   0.,  48.,  62.,  27.,
        27.,  51.,  18.,  65.,  28.,   2.,  54.,   4.,  14., 100.,  57.,
         0.,   5.,   0.,  30.,  83.,  93.,   0., 152.,   0.,   1.,  53.,
         0.,  81.,  78., 122.,   9.,  13.,  36.,  29.,  12.,  34., 100.,
        44.,  82.,   0.,   6.,  49.,  64.,  43.,  72.,   3.,  44.,  47.,
        17.,  35.,  88.,   0.,  33., 110., 146.,   7.,  11.,  94.,  55.,
         0.,  28.,   2.,  27.,  40.])
```

In [170…
```
np.mean(sachin[rahul <= 10])
```

Out[170…
```
40.2112676056338
```

In [ ]:
```
DAY10
```

In [ ]:
```
1.Pandas
2.nifty.csv(Data_set)
```

In [171…
```
import pandas as pd
import numpy as np
```

In [172…
```
# series, Dataframe
```

In [173…
```
s = pd.Series([0,1,1,2,3,5,8])
s
```

Out[173…
```
0    0
1    1
2    1
3    2
4    3
5    5
6    8
dtype: int64
```

In [174…
```
type(s)
```

Out[174…
```
pandas.core.series.Series
```

In [176…
```
s = pd.Series([0.0,1,1,2,3,5,8])
s
```

Out[176…
```
0    0.0
1    1.0
2    1.0
3    2.0
4    3.0
5    5.0
6    8.0
dtype: float64
```

In [177…
```
s.values
```

Out[177…
```
array([0., 1., 1., 2., 3., 5., 8.])
```

In [178…
```
type(s.values)
```

Out[178… numpy.ndarray

In [179…
```
s.index
```

Out[179… RangeIndex(start=0, stop=7, step=1)

In [180…
```
for v in s.values:
  print(v)
```

```
0.0
1.0
1.0
2.0
3.0
5.0
8.0
```

In [181…
```
for i in s.index:
  print(i)
```

```
0
1
2
3
4
5
6
```

In [182…
```
for item in zip(s.index, s.values):
  print(item, type(item))
```

```
(0, 0.0) <class 'tuple'>
(1, 1.0) <class 'tuple'>
(2, 1.0) <class 'tuple'>
(3, 2.0) <class 'tuple'>
(4, 3.0) <class 'tuple'>
(5, 5.0) <class 'tuple'>
(6, 8.0) <class 'tuple'>
```

In [183…
```
s[5]
```

Out[183… 5.0

In [184…
```
s[0]
```

Out[184… 0.0

In [185…
```
pd.Series([0.33, 57.9, 4222.6])
```

Out[185…
```
0       0.33
1      57.90
2    4222.60
dtype: float64
```

In [186…
```
# mass, diameter, dayLength


mercury = pd.Series([0.33, 57.9, 4222.6], index=["Mass", "Diameter", "DayLength"])
mercury
```

Out[186…
```
Mass            0.33
Diameter       57.90
DayLength    4222.60
dtype: float64
```

In [187…
```
mercury["Mass"]
```

Out[187…
```
0.33
```

In [188…
```
mercury["DayLength"]
```

Out[188…
```
4222.6
```

In [189…
```
arr = np.random.randint(0, 10, 10)
arr
```

Out[189…
```
array([6, 4, 1, 5, 3, 3, 0, 3, 0, 0])
```

In [190…
```
rand_series = pd.Series(arr)
rand_series
```

Out[190…
```
0    6
1    4
2    1
3    5
4    3
5    3
6    0
7    3
8    0
9    0
dtype: int32
```

In [191…
```
iLoc and Loc
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-191-0876149d6686> in <module>
----> 1 iLoc and Loc

NameError: name 'iLoc' is not defined
```

In [192…
```
mercury = pd.Series([0.33, 57.9, 4222.6], index=["Mass", "Diameter", "DayLength"])
mercury["Mass"]
```

Out[192…
```
0.33
```

In [193…
```
mercury[0]
```

Out[193…
```
0.33
```

In [194…
```
mercury.loc["DayLength"]
```

Out[194…
```
4222.6
```

In [ ]:
```
Nifty Case Study
```

In [196…
```
import pandas as pd
import numpy as np
```

In [198…
```
data=pd.read_csv(r'C:\Users\Kesam\Downloads\Kishorereddy\nifty.csv')
data
```

Out[198…

	Date	Close
0	01-Jan-2019	10910.10

	Date	Close
1	02-Jan-2019	10792.50
2	03-Jan-2019	10672.25
3	04-Jan-2019	10727.35
4	07-Jan-2019	10771.80
...	...	...
240	24-Dec-2019	12214.55
241	26-Dec-2019	12126.55
242	27-Dec-2019	12245.80
243	30-Dec-2019	12255.85
244	31-Dec-2019	12168.45

245 rows × 2 columns

In [203...
```
data.head()
```

Out[203...

	Date	Close
0	01-Jan-2019	10910.10
1	02-Jan-2019	10792.50
2	03-Jan-2019	10672.25
3	04-Jan-2019	10727.35
4	07-Jan-2019	10771.80

In [204...
```
data.head(10)
```

Out[204...

	Date	Close
0	01-Jan-2019	10910.10
1	02-Jan-2019	10792.50
2	03-Jan-2019	10672.25
3	04-Jan-2019	10727.35
4	07-Jan-2019	10771.80
5	08-Jan-2019	10802.15
6	09-Jan-2019	10855.15
7	10-Jan-2019	10821.60
8	11-Jan-2019	10794.95
9	14-Jan-2019	10737.60

In [205...
```
data.tail()
```

Out[205...

	Date	Close

	Date	Close
240	24-Dec-2019	12214.55
241	26-Dec-2019	12126.55
242	27-Dec-2019	12245.80
243	30-Dec-2019	12255.85
244	31-Dec-2019	12168.45

In [206…
```
data.tail(10)
```

Out[206…

	Date	Close
235	17-Dec-2019	12165.00
236	18-Dec-2019	12221.65
237	19-Dec-2019	12259.70
238	20-Dec-2019	12271.80
239	23-Dec-2019	12262.75
240	24-Dec-2019	12214.55
241	26-Dec-2019	12126.55
242	27-Dec-2019	12245.80
243	30-Dec-2019	12255.85
244	31-Dec-2019	12168.45

In [207…
```
np.mean(data)
```

Out[207…
```
Close    11432.632245
dtype: float64
```

In [208…
```
type(data)
```

Out[208…
```
pandas.core.frame.DataFrame
```

In [ ]:
```
Question: How many days did the markets close higher than the previous day's close.
```

In [211…
```
data[1:]
```

Out[211…

	Date	Close
1	02-Jan-2019	10792.50
2	03-Jan-2019	10672.25
3	04-Jan-2019	10727.35
4	07-Jan-2019	10771.80
5	08-Jan-2019	10802.15
…	…	…
240	24-Dec-2019	12214.55

	Date	Close
241	26-Dec-2019	12126.55
242	27-Dec-2019	12245.80
243	30-Dec-2019	12255.85
244	31-Dec-2019	12168.45

244 rows × 2 columns

In [212…
```
data[:-1]
```

Out[212…

	Date	Close
0	01-Jan-2019	10910.10
1	02-Jan-2019	10792.50
2	03-Jan-2019	10672.25
3	04-Jan-2019	10727.35
4	07-Jan-2019	10771.80
…	…	…
239	23-Dec-2019	12262.75
240	24-Dec-2019	12214.55
241	26-Dec-2019	12126.55
242	27-Dec-2019	12245.80
243	30-Dec-2019	12255.85

244 rows × 2 columns

In [ ]:
```
DAY11
```

Pandas

Numpy

In [216…
```
import pandas as pd
import numpy as np
```

In [217…
```
arr = np.random.randint(0, 10, (5, 3))
arr
```

Out[217…
```
array([[5, 2, 5],
       [0, 7, 2],
       [2, 1, 8],
       [0, 8, 3],
       [9, 7, 4]])
```

In [218…
```
df = pd.DataFrame(arr)
df
```

Out[218...

	0	1	2
0	5	2	5
1	0	7	2
2	2	1	8
3	0	8	3
4	9	7	4

In [219...
```
df.values
```

Out[219...
```
array([[5, 2, 5],
       [0, 7, 2],
       [2, 1, 8],
       [0, 8, 3],
       [9, 7, 4]])
```

In [220...
```
df.index
```

Out[220...
```
RangeIndex(start=0, stop=5, step=1)
```

In [221...
```
df.columns
```

Out[221...
```
RangeIndex(start=0, stop=3, step=1)
```

In [222...
```
for c in df.columns:
  print(c)
```

```
0
1
2
```

In [223...
```
df.values[1]
```

Out[223...
```
array([0, 7, 2])
```

In [224...
```
df.index = ["R1", "R2", "R3", "R4", "R5"]
df.columns = ["C1", "C2", "C3"]
df
```

Out[224...

	C1	C2	C3
R1	5	2	5
R2	0	7	2
R3	2	1	8
R4	0	8	3
R5	9	7	4

In [225...
```
df.loc["R3", "C2"]
```

Out[225...
```
1
```

In [228...
```
temp_df = df.iloc[2:4, 1:3]
temp_df
```

Out[228...

	C2	C3

	C2	C3
R3	1	8
R4	8	3

In [229…
```
type(temp_df)
```

Out[229…
```
pandas.core.frame.DataFrame
```

In [230…
```
df.iloc[0]
```

Out[230…
```
C1    5
C2    2
C3    5
Name: R1, dtype: int32
```

In [231…
```
type(df.iloc[0])
```

Out[231…
```
pandas.core.series.Series
```

In [232…
```
df.iloc[:, 0]
```

Out[232…
```
R1    5
R2    0
R3    2
R4    0
R5    9
Name: C1, dtype: int32
```

In [ ]:
```
DAY12
```

In [ ]:
```
1.Pandas
2.Numpy
```

In [233…
```
import pandas as pd
import numpy as np
```

In [234…
```
def create_df(nRows, nCols, maxRand=10):
  arr = np.random.randint(0, maxRand, (nRows, nCols))
  df = pd.DataFrame(arr)

  df.index = ['R' + str(x) for x in np.arange(1, nRows+1)]
  df.columns = ['C' + str(x) for x in np.arange(1, nCols+1)]

  return df
create_df(5, 3)
```

Out[234…

	C1	C2	C3
R1	7	5	1
R2	1	4	9
R3	3	2	8
R4	8	8	8
R5	2	1	4

In [235…
```
create_df(2, 5)
```

Out[235...

	C1	C2	C3	C4	C5
R1	1	0	4	8	0
R2	5	6	7	7	1

In [236...
```
import seaborn as sns
```

```
Matplotlib is building the font cache; this may take a moment.
```

In [237...
```
df = sns.load_dataset("planets")
df.head()
```

Out[237...

	method	number	orbital_period	mass	distance	year
0	Radial Velocity	1	269.300	7.10	77.40	2006
1	Radial Velocity	1	874.774	2.21	56.95	2008
2	Radial Velocity	1	763.000	2.60	19.84	2011
3	Radial Velocity	1	326.030	19.40	110.62	2007
4	Radial Velocity	1	516.220	10.50	119.47	2009

In [238...
```
df.tail()
```

Out[238...

	method	number	orbital_period	mass	distance	year
1030	Transit	1	3.941507	NaN	172.0	2006
1031	Transit	1	2.615864	NaN	148.0	2007
1032	Transit	1	3.191524	NaN	174.0	2007
1033	Transit	1	4.125083	NaN	293.0	2008
1034	Transit	1	4.187757	NaN	260.0	2008

In [239...
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1035 entries, 0 to 1034
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   method          1035 non-null   object 
 1   number          1035 non-null   int64  
 2   orbital_period  992 non-null    float64
 3   mass            513 non-null    float64
 4   distance        808 non-null    float64
 5   year            1035 non-null   int64  
dtypes: float64(3), int64(2), object(1)
memory usage: 48.6+ KB
```

In [240...
```
df.describe()
```

Out[240...

	number	orbital_period	mass	distance	year
count	1035.000000	992.000000	513.000000	808.000000	1035.000000
mean	1.785507	2002.917596	2.638161	264.069282	2009.070531
std	1.240976	26014.728304	3.818617	733.116493	3.972567

	number	orbital_period	mass	distance	year
min	1.000000	0.090706	0.003600	1.350000	1989.000000
25%	1.000000	5.442540	0.229000	32.560000	2007.000000
50%	1.000000	39.979500	1.260000	55.250000	2010.000000
75%	2.000000	526.005000	3.040000	178.500000	2012.000000
max	7.000000	730000.000000	25.000000	8500.000000	2014.000000

In [ ]:
```
Question 1: Go through each row of the dataframe and delete it(Drop) if any of the c
```

In [241…
```
# Method 1
for row in df.index:
  for column in df.columns:
    if pd.isnull(df.loc[row, column]):
      df.drop(row, inplace=True)
      break
df.describe()
```

Out[241…

	number	orbital_period	mass	distance	year
count	498.00000	498.000000	498.000000	498.000000	498.000000
mean	1.73494	835.778671	2.509320	52.068213	2007.377510
std	1.17572	1469.128259	3.636274	46.596041	4.167284
min	1.00000	1.328300	0.003600	1.350000	1989.000000
25%	1.00000	38.272250	0.212500	24.497500	2005.000000
50%	1.00000	357.000000	1.245000	39.940000	2009.000000
75%	2.00000	999.600000	2.867500	59.332500	2011.000000
max	6.00000	17337.500000	25.000000	354.000000	2014.000000

In [242…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 498 entries, 0 to 784
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   method          498 non-null    object 
 1   number          498 non-null    int64  
 2   orbital_period  498 non-null    float64
 3   mass            498 non-null    float64
 4   distance        498 non-null    float64
 5   year            498 non-null    int64  
dtypes: float64(3), int64(2), object(1)
memory usage: 27.2+ KB
```

In [243…
```
df = sns.load_dataset("planets")

# Method 2
for i, r in df.iterrows():
  print(i)
  print(r)
  break
```

```
0
```

```
method            Radial Velocity
number                          1
orbital_period              269.3
mass                          7.1
distance                     77.4
year                         2006
Name: 0, dtype: object
```

In [244…
```
for i, r in df.iterrows():
  print(pd.isnull(r).any())
  break
```

```
False
```

In [245…
```
for i, r in df.iterrows():
  if pd.isnull(r).any():
    df.drop(i, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 498 entries, 0 to 784
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   method          498 non-null    object 
 1   number          498 non-null    int64  
 2   orbital_period  498 non-null    float64
 3   mass            498 non-null    float64
 4   distance        498 non-null    float64
 5   year            498 non-null    int64  
dtypes: float64(3), int64(2), object(1)
memory usage: 27.2+ KB
```

In [246…
```
# Method 3
df = sns.load_dataset("planets")

df.dropna(inplace=True)
df.describe()
```

Out[246…

	number	orbital_period	mass	distance	year
count	498.00000	498.000000	498.000000	498.000000	498.000000
mean	1.73494	835.778671	2.509320	52.068213	2007.377510
std	1.17572	1469.128259	3.636274	46.596041	4.167284
min	1.00000	1.328300	0.003600	1.350000	1989.000000
25%	1.00000	38.272250	0.212500	24.497500	2005.000000
50%	1.00000	357.000000	1.245000	39.940000	2009.000000
75%	2.00000	999.600000	2.867500	59.332500	2011.000000
max	6.00000	17337.500000	25.000000	354.000000	2014.000000

In [ ]:
```
Question 2: Filter and show only those rows which have planets that are found in the
```

In [247…
```
df_ = df.copy()
# Query the DataFrame
df_ = df_[
          (df_['year'] >= 2010) &
          ((df_['method'] == 'Radial Velocity') |
          (df_['method'] == 'Transit'))
```

```
]
df_['year'].unique()
```

Out[247…
```
array([2011, 2010, 2013, 2012, 2014], dtype=int64)
```

In [248…
```
df_.head()
```

Out[248…

	method	number	orbital_period	mass	distance	year
2	Radial Velocity	1	763.0	2.60	19.84	2011
9	Radial Velocity	2	452.8	1.99	74.79	2010
10	Radial Velocity	2	883.0	0.86	74.79	2010
28	Radial Velocity	1	181.4	3.20	45.52	2013
45	Radial Velocity	1	380.8	1.80	20.21	2010

In [249…
```
df_['method'].unique()
```

Out[249…
```
array(['Radial Velocity', 'Transit'], dtype=object)
```

In [250…
```
df = sns.load_dataset("planets")
df.head()
```

Out[250…

	method	number	orbital_period	mass	distance	year
0	Radial Velocity	1	269.300	7.10	77.40	2006
1	Radial Velocity	1	874.774	2.21	56.95	2008
2	Radial Velocity	1	763.000	2.60	19.84	2011
3	Radial Velocity	1	326.030	19.40	110.62	2007
4	Radial Velocity	1	516.220	10.50	119.47	2009

In [251…
```
df['method'].unique()
```

Out[251…
```
array(['Radial Velocity', 'Imaging', 'Eclipse Timing Variations',
       'Transit', 'Astrometry', 'Transit Timing Variations',
       'Orbital Brightness Modulation', 'Microlensing', 'Pulsar Timing',
       'Pulsation Timing Variations'], dtype=object)
```

In [ ]:
```
DAY13
```

In [ ]:
```
1.Visualization
2.Tabulation
3.Read Complex json files
4.Histogram
4.Barplot
```

In [252…
```
import seaborn as sns
import pandas as pd
import numpy as np
```

In [253…
```
import urllib.request
```

In [ ]:
```
Read Complex Json files
```

In [254…
```
url = "https://api.covid19india.org/states_daily.json"
```

```
urllib.request.urlretrieve(url, 'data.json')
```

Out[254… ('data.json', <http.client.HTTPMessage at 0x20b9f592e50>)

In [255…
```
covid_data = pd.read_json('data.json')
covid_data
```

Out[255…

	states_daily
0	{'an': '0', 'ap': '1', 'ar': '0', 'as': '0', '...
1	{'an': '0', 'ap': '0', 'ar': '0', 'as': '0', '...
2	{'an': '0', 'ap': '0', 'ar': '0', 'as': '0', '...
3	{'an': '0', 'ap': '0', 'ar': '0', 'as': '0', '...
4	{'an': '0', 'ap': '0', 'ar': '0', 'as': '0', '...
...	...
1558	{'an': '2', 'ap': '1835', 'ar': '255', 'as': '...
1559	{'an': '0', 'ap': '16', 'ar': '0', 'as': '10',...
1560	{'an': '1', 'ap': '909', 'ar': '165', 'as': '7...
1561	{'an': '0', 'ap': '1543', 'ar': '249', 'as': '...
1562	{'an': '0', 'ap': '13', 'ar': '0', 'as': '10',...

1563 rows × 1 columns

In [256…
```
import json
```

In [257…
```
with open('data.json') as f:
  data = json.load(f)
data
data = data['states_daily']
covid_data = pd.json_normalize(data)
covid_data
```

Out[257…

	an	ap	ar	as	br	ch	ct	date	dateymd	dd	...	sk	status	tg	tn	tr
0	0	1	0	0	0	0	0	14-Mar-20	2020-03-14	0	...	0	Confirmed	1	1	0
1	0	0	0	0	0	0	0	14-Mar-20	2020-03-14	0	...	0	Recovered	0	0	0
2	0	0	0	0	0	0	0	14-Mar-20	2020-03-14	0	...	0	Deceased	0	0	0
3	0	0	0	0	0	0	0	15-Mar-20	2020-03-15	0	...	0	Confirmed	2	0	0
4	0	0	0	0	0	0	0	15-Mar-20	2020-03-15	0	...	0	Recovered	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

	an	ap	ar	as	br	ch	ct	date	dateymd	dd	...	sk	status	tg	tn	tr
1558	2	1835	255	857	38	1	114	15-Aug-21	2021-08-15	0	...	213	Recovered	582	1842	253
1559	0	16	0	10	0	0	1	15-Aug-21	2021-08-15	0	...	0	Deceased	1	23	4
1560	1	909	165	758	14	2	68	16-Aug-21	2021-08-16	0	...	20	Confirmed	405	1851	52
1561	0	1543	249	1014	42	3	224	16-Aug-21	2021-08-16	0	...	147	Recovered	577	1911	223
1562	0	13	0	10	0	0	1	16-Aug-21	2021-08-16	0	...	0	Deceased	3	28	1

1563 rows × 42 columns

In [258…
```
covid_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1563 entries, 0 to 1562
Data columns (total 42 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   an       1563 non-null   object
 1   ap       1563 non-null   object
 2   ar       1563 non-null   object
 3   as       1563 non-null   object
 4   br       1563 non-null   object
 5   ch       1563 non-null   object
 6   ct       1563 non-null   object
 7   date     1563 non-null   object
 8   dateymd  1563 non-null   object
 9   dd       1563 non-null   object
 10  dl       1563 non-null   object
 11  dn       1563 non-null   object
 12  ga       1563 non-null   object
 13  gj       1563 non-null   object
 14  hp       1563 non-null   object
 15  hr       1563 non-null   object
 16  jh       1563 non-null   object
 17  jk       1563 non-null   object
 18  ka       1563 non-null   object
 19  kl       1563 non-null   object
 20  la       1563 non-null   object
 21  ld       1563 non-null   object
 22  mh       1563 non-null   object
 23  ml       1563 non-null   object
 24  mn       1563 non-null   object
 25  mp       1563 non-null   object
 26  mz       1563 non-null   object
 27  nl       1563 non-null   object
 28  or       1563 non-null   object
 29  pb       1563 non-null   object
 30  py       1563 non-null   object
 31  rj       1563 non-null   object
 32  sk       1563 non-null   object
 33  status   1563 non-null   object
 34  tg       1563 non-null   object
```

```
 35  tn       1563 non-null   object
 36  tr       1563 non-null   object
 37  tt       1563 non-null   object
 38  un       1563 non-null   object
 39  up       1563 non-null   object
 40  ut       1563 non-null   object
 41  wb       1563 non-null   object
dtypes: object(42)
memory usage: 513.0+ KB
```

In [259…

```
# covid_data = pd.to_datetime(covid_data.date)
# covid_data.info()

covid_data = covid_data[covid_data.status == 'Confirmed']
covid_data
```

Out[259…

	an	ap	ar	as	br	ch	ct	date	dateymd	dd	...	sk	status	tg	tn	tr	
0	0	1	0	0	0	0	0	14-Mar-20	2020-03-14	0	...	0	Confirmed	1	1	0	
3	0	0	0	0	0	0	0	15-Mar-20	2020-03-15	0	...	0	Confirmed	2	0	0	
6	0	0	0	0	0	0	0	16-Mar-20	2020-03-16	0	...	0	Confirmed	1	0	0	
9	0	0	0	0	0	0	0	17-Mar-20	2020-03-17	0	...	0	Confirmed	1	0	0	
12	0	0	0	0	0	0	0	18-Mar-20	2020-03-18	0	...	0	Confirmed	8	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1548	0	1859	180	935	43	12	98	12-Aug-21	2021-08-12	0	...	100	Confirmed	453	1942	140	4(
1551	0	1746	166	763	47	15	77	13-Aug-21	2021-08-13	0	...	150	Confirmed	427	1933	180	3{
1554	0	1535	161	755	39	4	83	14-Aug-21	2021-08-14	0	...	129	Confirmed	420	1916	137	3(
1557	0	1506	48	411	28	1	49	15-Aug-21	2021-08-15	0	...	152	Confirmed	245	1896	142	3:
1560	1	909	165	758	14	2	68	16-Aug-21	2021-08-16	0	...	20	Confirmed	405	1851	52	2‹

521 rows × 42 columns

In [260…

```
covid_data.drop('status', axis=1, inplace=True)
covid_data.drop('dateymd', axis=1, inplace=True)
```

```
C:\anaconda\lib\site-packages\pandas\core\frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

In [261…
```
covid_data
```

Out[261…

	an	ap	ar	as	br	ch	ct	date	dd	dl	…	rj	sk	tg	tn	tr	tt	un	up
0	0	1	0	0	0	0	0	14-Mar-20	0	7	…	3	0	1	1	0	81	0	12
3	0	0	0	0	0	0	0	15-Mar-20	0	0	…	1	0	2	0	0	27	0	1
6	0	0	0	0	0	0	0	16-Mar-20	0	0	…	0	0	1	0	0	15	0	0
9	0	0	0	0	0	0	0	17-Mar-20	0	1	…	0	0	1	0	0	11	0	2
12	0	0	0	0	0	0	0	18-Mar-20	0	2	…	3	0	8	1	0	37	0	2
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1548	0	1859	180	935	43	12	98	12-Aug-21	0	49	…	17	100	453	1942	140	40081	0	15
1551	0	1746	166	763	47	15	77	13-Aug-21	0	50	…	24	150	427	1933	180	38761	0	25
1554	0	1535	161	755	39	4	83	14-Aug-21	0	50	…	14	129	420	1916	137	36135	0	42
1557	0	1506	48	411	28	1	49	15-Aug-21	0	53	…	18	152	245	1896	142	33245	0	30
1560	1	909	165	758	14	2	68	16-Aug-21	0	27	…	11	20	405	1851	52	24696	0	17

521 rows × 40 columns

In [262…
```
covid_data.set_index('date', inplace=True)
covid_data.tail(7)
```

Out[262…

date	an	ap	ar	as	br	ch	ct	dd	dl	dn	…	rj	sk	tg	tn	tr	tt	un	up
10-Aug-21	2	1461	233	929	44	8	112	0	52	1	…	11	110	494	1893	205	38376	0	19

date	an	ap	ar	as	br	ch	ct	dd	dl	dn	...	rj	sk	tg	tn	tr	tt	un	up
11-Aug-21	0	1869	188	886	47	5	83	0	37	0	...	19	157	482	1964	244	41586	0	24
12-Aug-21	0	1859	180	935	43	12	98	0	49	1	...	17	100	453	1942	140	40081	0	15
13-Aug-21	0	1746	166	763	47	15	77	0	50	0	...	24	150	427	1933	180	38761	0	25
14-Aug-21	0	1535	161	755	39	4	83	0	50	0	...	14	129	420	1916	137	36135	0	42
15-Aug-21	0	1506	48	411	28	1	49	0	53	0	...	18	152	245	1896	142	33245	0	30
16-Aug-21	1	909	165	758	14	2	68	0	27	2	...	11	20	405	1851	52	24696	0	17

7 rows × 39 columns

In [263…
```
covid_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 521 entries, 14-Mar-20 to 16-Aug-21
Data columns (total 39 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   an      521 non-null    object
 1   ap      521 non-null    object
 2   ar      521 non-null    object
 3   as      521 non-null    object
 4   br      521 non-null    object
 5   ch      521 non-null    object
 6   ct      521 non-null    object
 7   dd      521 non-null    object
 8   dl      521 non-null    object
 9   dn      521 non-null    object
 10  ga      521 non-null    object
 11  gj      521 non-null    object
 12  hp      521 non-null    object
 13  hr      521 non-null    object
 14  jh      521 non-null    object
 15  jk      521 non-null    object
 16  ka      521 non-null    object
 17  kl      521 non-null    object
 18  la      521 non-null    object
 19  ld      521 non-null    object
 20  mh      521 non-null    object
 21  ml      521 non-null    object
 22  mn      521 non-null    object
 23  mp      521 non-null    object
 24  mz      521 non-null    object
 25  nl      521 non-null    object
 26  or      521 non-null    object
 27  pb      521 non-null    object
 28  py      521 non-null    object
```

```
 29  rj      521 non-null    object
 30  sk      521 non-null    object
 31  tg      521 non-null    object
 32  tn      521 non-null    object
 33  tr      521 non-null    object
 34  tt      521 non-null    object
 35  un      521 non-null    object
 36  up      521 non-null    object
 37  ut      521 non-null    object
 38  wb      521 non-null    object
dtypes: object(39)
memory usage: 162.8+ KB
```

In [264…
```
pd.to_numeric(covid_data.ka)
```

Out[264…
```
date
14-Mar-20       6
15-Mar-20       0
16-Mar-20       1
17-Mar-20       2
18-Mar-20       5
               ...
12-Aug-21    1857
13-Aug-21    1669
14-Aug-21    1632
15-Aug-21    1431
16-Aug-21    1065
Name: ka, Length: 521, dtype: int64
```

In [265…
```
covid_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 521 entries, 14-Mar-20 to 16-Aug-21
Data columns (total 39 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   an      521 non-null    object
 1   ap      521 non-null    object
 2   ar      521 non-null    object
 3   as      521 non-null    object
 4   br      521 non-null    object
 5   ch      521 non-null    object
 6   ct      521 non-null    object
 7   dd      521 non-null    object
 8   dl      521 non-null    object
 9   dn      521 non-null    object
 10  ga      521 non-null    object
 11  gj      521 non-null    object
 12  hp      521 non-null    object
 13  hr      521 non-null    object
 14  jh      521 non-null    object
 15  jk      521 non-null    object
 16  ka      521 non-null    object
 17  kl      521 non-null    object
 18  la      521 non-null    object
 19  ld      521 non-null    object
 20  mh      521 non-null    object
 21  ml      521 non-null    object
 22  mn      521 non-null    object
 23  mp      521 non-null    object
 24  mz      521 non-null    object
 25  nl      521 non-null    object
 26  or      521 non-null    object
 27  pb      521 non-null    object
 28  py      521 non-null    object
 29  rj      521 non-null    object
 30  sk      521 non-null    object
 31  tg      521 non-null    object
 32  tn      521 non-null    object
```

```
 33  tr      521 non-null    object
 34  tt      521 non-null    object
 35  un      521 non-null    object
 36  up      521 non-null    object
 37  ut      521 non-null    object
 38  wb      521 non-null    object
dtypes: object(39)
memory usage: 182.8+ KB
```

In [266…
```
covid_data = covid_data.apply(pd.to_numeric)
covid_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 521 entries, 14-Mar-20 to 16-Aug-21
Data columns (total 39 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   an      521 non-null    int64
 1   ap      521 non-null    int64
 2   ar      521 non-null    int64
 3   as      521 non-null    int64
 4   br      521 non-null    int64
 5   ch      521 non-null    int64
 6   ct      521 non-null    int64
 7   dd      521 non-null    int64
 8   dl      521 non-null    int64
 9   dn      521 non-null    int64
 10  ga      521 non-null    int64
 11  gj      521 non-null    int64
 12  hp      521 non-null    int64
 13  hr      521 non-null    int64
 14  jh      521 non-null    int64
 15  jk      521 non-null    int64
 16  ka      521 non-null    int64
 17  kl      521 non-null    int64
 18  la      521 non-null    int64
 19  ld      521 non-null    int64
 20  mh      521 non-null    int64
 21  ml      521 non-null    int64
 22  mn      521 non-null    int64
 23  mp      521 non-null    int64
 24  mz      521 non-null    int64
 25  nl      521 non-null    int64
 26  or      521 non-null    int64
 27  pb      521 non-null    int64
 28  py      521 non-null    int64
 29  rj      521 non-null    int64
 30  sk      521 non-null    int64
 31  tg      521 non-null    int64
 32  tn      521 non-null    int64
 33  tr      521 non-null    int64
 34  tt      521 non-null    int64
 35  un      521 non-null    int64
 36  up      521 non-null    int64
 37  ut      521 non-null    int64
 38  wb      521 non-null    int64
dtypes: int64(39)
memory usage: 182.8+ KB
```

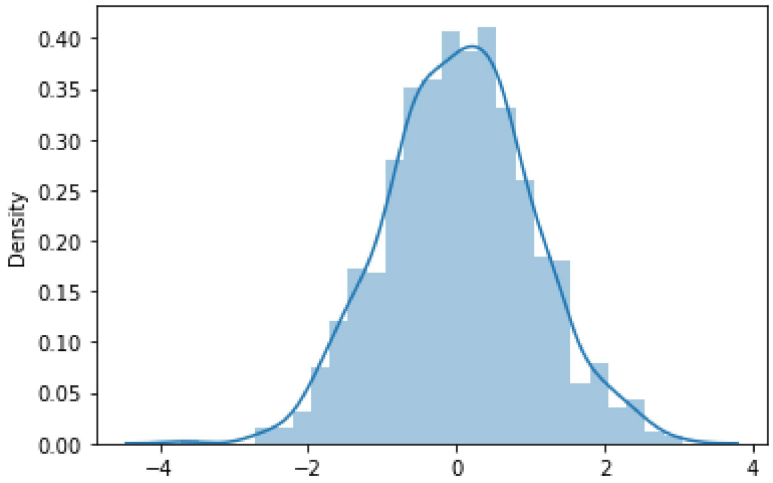
In [ ]:
```
Histogram
```

In [267…
```
x = np.random.normal(size=1000)
x
sns.distplot(x) # Distribution plot
```

```
C:\anaconda\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplo
t` is a deprecated function and will be removed in a future version. Please adapt yo
ur code to use either `displot` (a figure-level function with similar flexibility) o
```

```
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

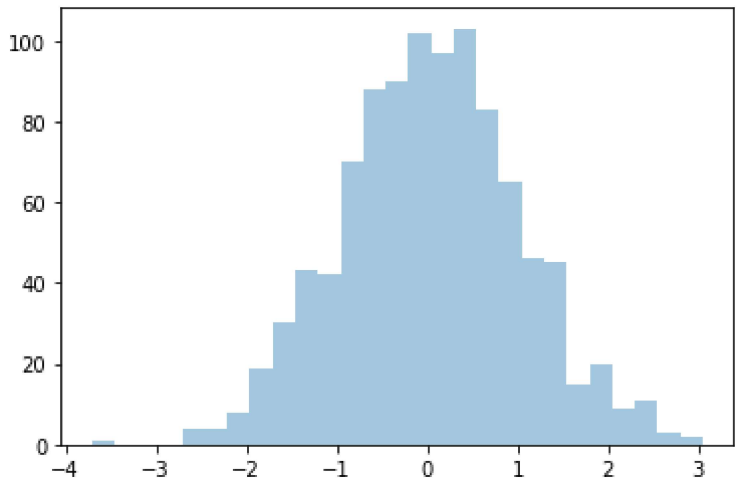
Out[267… <AxesSubplot:ylabel='Density'>



In [268…
```
sns.distplot(x, kde=False)
```

```
C:\anaconda\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplo
t` is a deprecated function and will be removed in a future version. Please adapt yo
ur code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

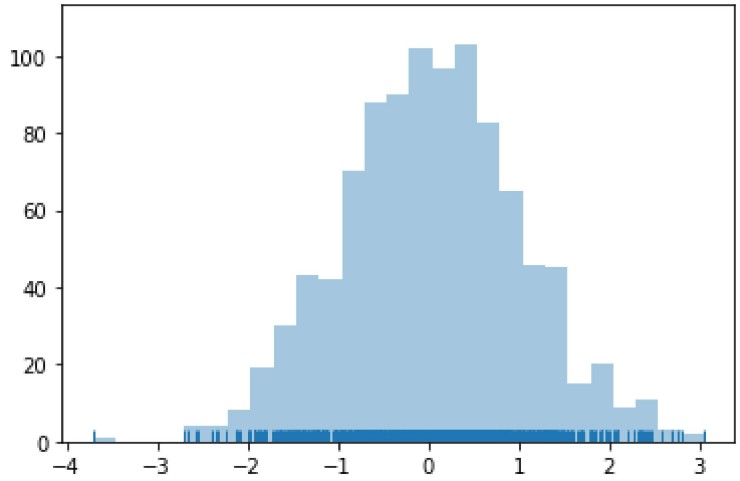
Out[268… <AxesSubplot:>



In [269…
```
sns.distplot(x, kde=False, rug=True)
```

```
C:\anaconda\lib\site-packages\seaborn\distributions.py:2055: FutureWarning: The `axi
s` variable is no longer used and will be removed. Instead, assign variables directl
y to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```

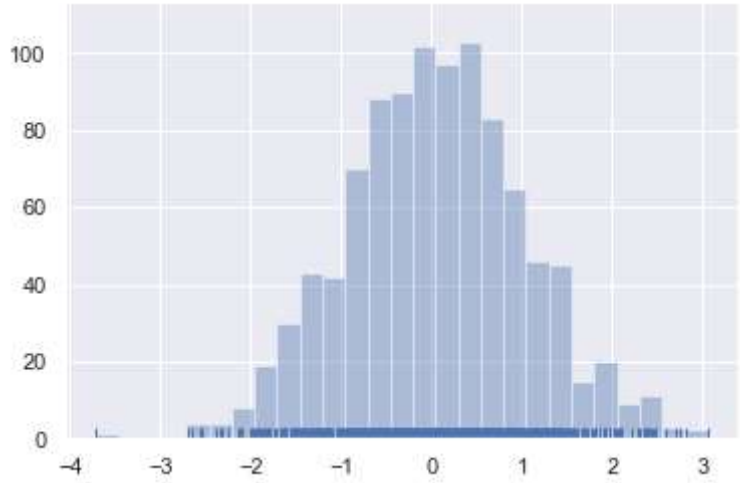
Out[269… <AxesSubplot:>



In [270…
```
sns.set(color_codes=True)
sns.distplot(x, kde=False, rug=True)
```

```
C:\anaconda\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplo
t` is a deprecated function and will be removed in a future version. Please adapt yo
ur code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\anaconda\lib\site-packages\seaborn\distributions.py:2055: FutureWarning: The `axi
s` variable is no longer used and will be removed. Instead, assign variables directl
y to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```

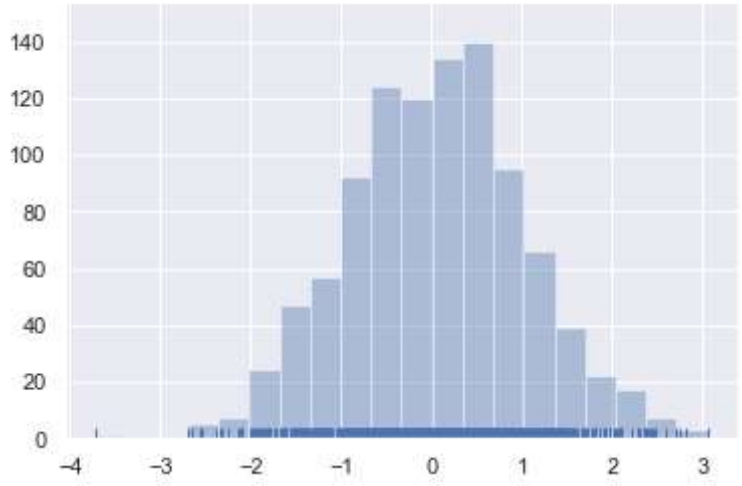
Out[270… <AxesSubplot:>



In [271…
```
sns.distplot(x, kde=False, rug=True, bins=20)
```

```
C:\anaconda\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplo
t` is a deprecated function and will be removed in a future version. Please adapt yo
ur code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\anaconda\lib\site-packages\seaborn\distributions.py:2055: FutureWarning: The `axi
s` variable is no longer used and will be removed. Instead, assign variables directl
y to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```

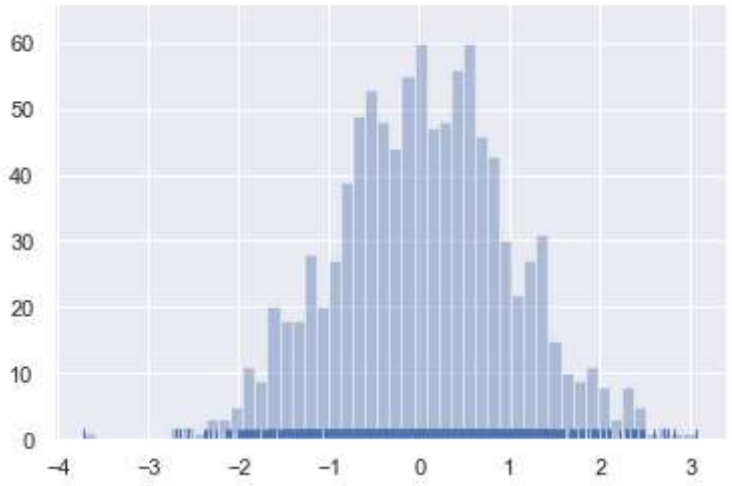
Out[271… <AxesSubplot:>



In [272…
```
sns.distplot(x, kde=False, rug=True, bins=50)
```

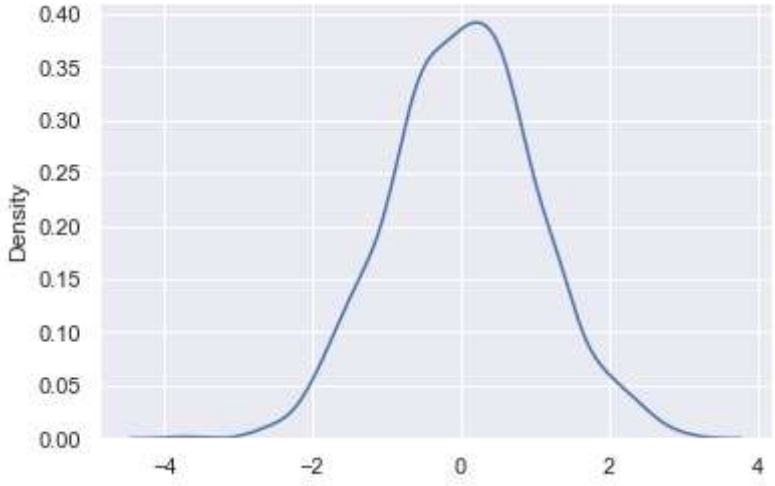
```
C:\anaconda\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplo
t` is a deprecated function and will be removed in a future version. Please adapt yo
ur code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\anaconda\lib\site-packages\seaborn\distributions.py:2055: FutureWarning: The `axi
s` variable is no longer used and will be removed. Instead, assign variables directl
y to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```

Out[272…
```
<AxesSubplot:>
```



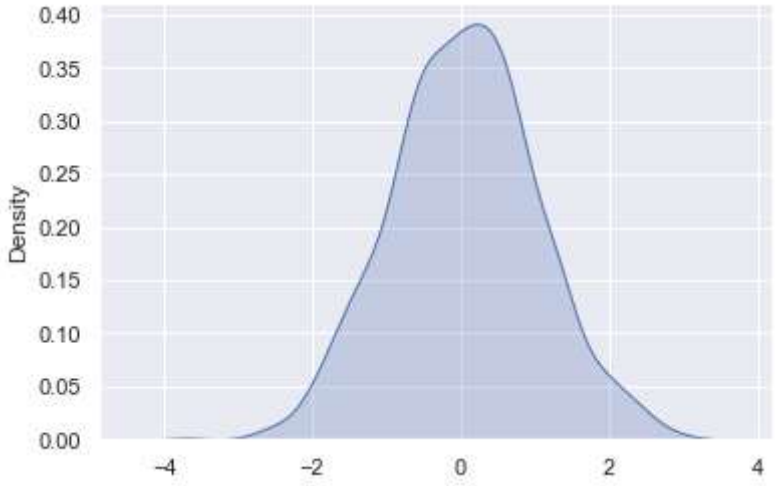
In [273…
```
sns.kdeplot(x)
```

Out[273…
```
<AxesSubplot:ylabel='Density'>
```



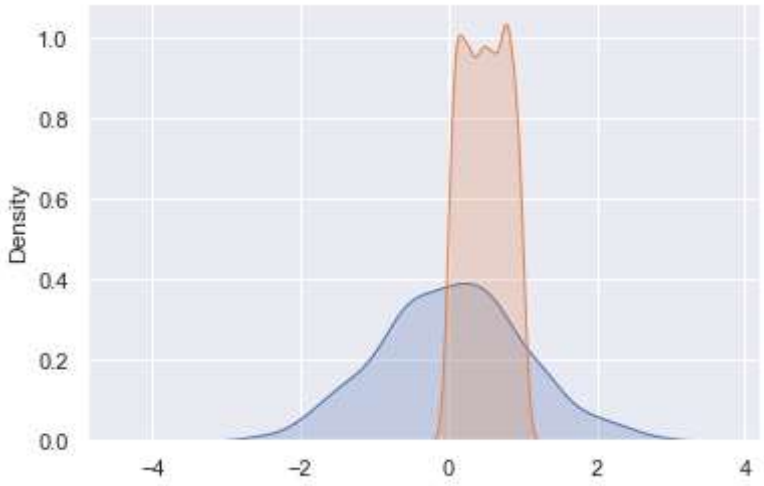
In [274…
```
sns.kdeplot(x, shade=True)
```

Out[274… <AxesSubplot:ylabel='Density'>



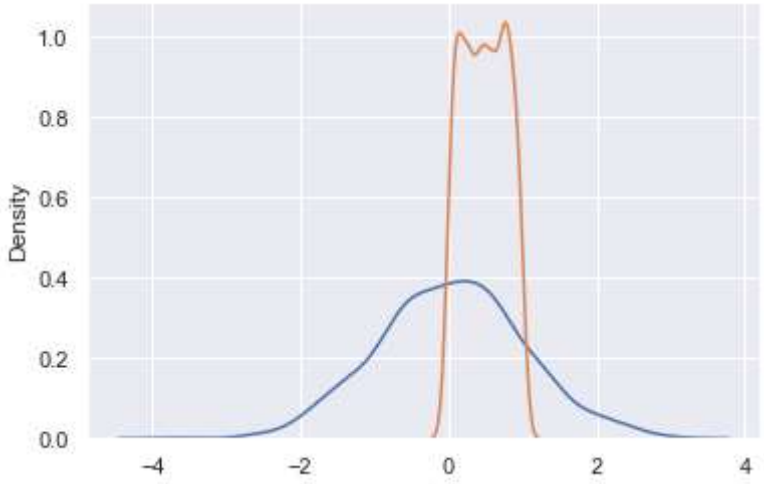
In [275…
```
y = np.random.uniform(size=1000)
sns.kdeplot(x, shade=True)
sns.kdeplot(y, shade=True)
```

Out[275… <AxesSubplot:ylabel='Density'>



In [276…
```
sns.kdeplot(x)
sns.kdeplot(y)
```

Out[276… <AxesSubplot:ylabel='Density'>



In [277…
```
sns.distplot?
```

In [ ]:
```
Bar Plot
```

In [278…
```
diamonds = sns.load_dataset("diamonds")
diamonds.head()
```

Out[278…

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

In [279…
```
diamonds.groupby('cut').count()
```

Out[279…

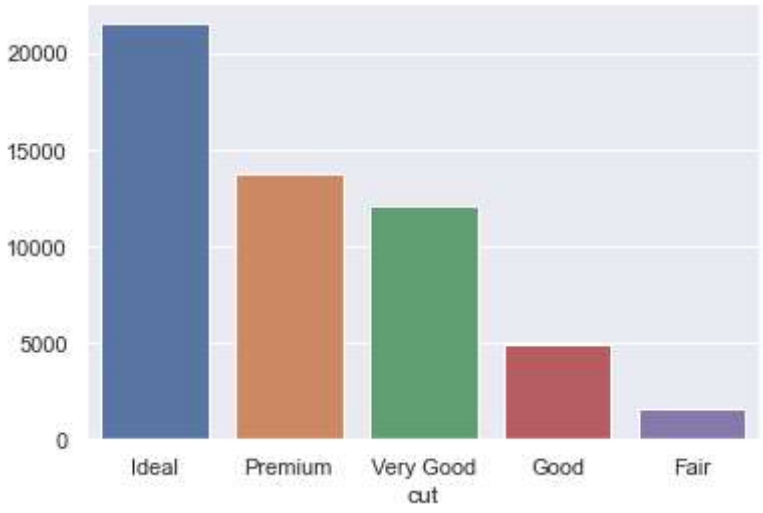
cut	carat	color	clarity	depth	table	price	x	y	z
Ideal	21551	21551	21551	21551	21551	21551	21551	21551	21551
Premium	13791	13791	13791	13791	13791	13791	13791	13791	13791
Very Good	12082	12082	12082	12082	12082	12082	12082	12082	12082
Good	4906	4906	4906	4906	4906	4906	4906	4906	4906
Fair	1610	1610	1610	1610	1610	1610	1610	1610	1610

In [280…
```
cut = diamonds.groupby('cut')['cut'].count()
cut
```

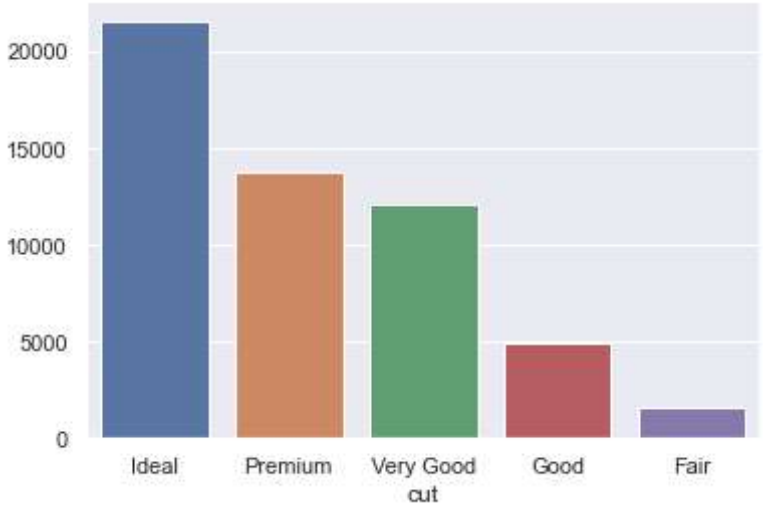
Out[280…
```
cut
Ideal        21551
Premium      13791
Very Good    12082
Good          4906
Fair          1610
Name: cut, dtype: int64
```

In [281…
```
sns.barplot(x=cut.index, y=cut.values)
```

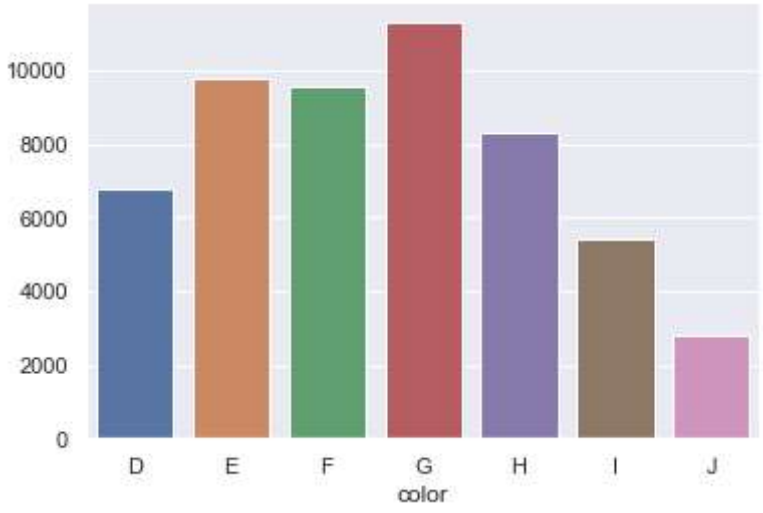
Out[281… <AxesSubplot:xlabel='cut'>



In [282…
```
sns.barplot(x=cut.index, y=cut.values);
```



In [283…
```
color = diamonds.groupby('color')['color'].count()
sns.barplot(x=color.index, y=color.values);
```

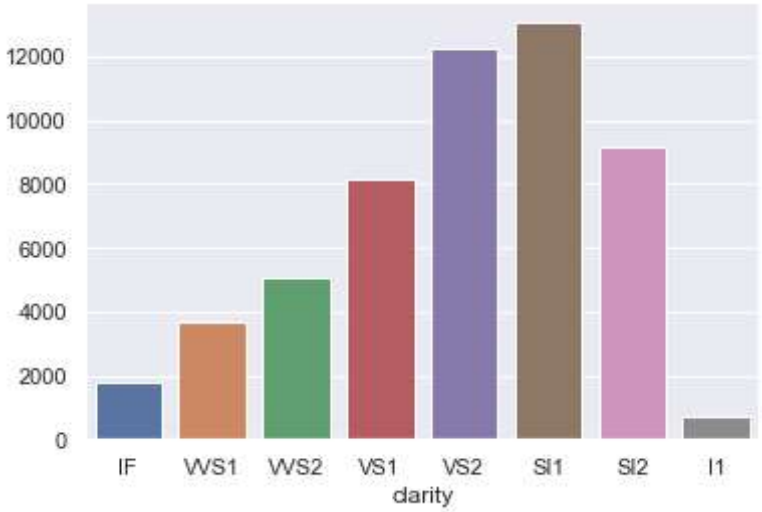


In [284…
```
color
```

Out[284…
```
color
D     6775
E     9797
F     9542
```

```
G    11292
H     8304
I     5422
J     2808
Name: color, dtype: int64
```

In [285…
```
clarity = diamonds.groupby('clarity')['clarity'].count()
sns.barplot(x=clarity.index, y=clarity.values);
```



In [286…
```
clarity
```

Out[286…
```
clarity
IF       1790
VVS1     3655
VVS2     5066
VS1      8171
VS2     12258
SI1     13065
SI2      9194
I1        741
Name: clarity, dtype: int64
```

In [ ]:
```
DAY14
```

In [ ]:
```
Regression Problem
```

In [287…
```
import pandas as pd
```

In [288…
```
df = pd.read_csv('https://raw.githubusercontent.com/instituteofai/ML-101/master/Data
df.head()
```

Out[288…

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0

In [289…
```
df.shape
```

```
(20640, 10)
```

Out[289…

In [290…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

In [291…
```
df.describe()
```

Out[291…

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populatior
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.00000(
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.00000(
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.00000(
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.00000(
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.00000(
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.00000(

In [292…
```
labels = df[['median_house_value']]
labels.head()
```

Out[292…

	median_house_value
0	452600.0
1	358500.0
2	352100.0
3	341300.0
4	342200.0

In [293…
```
features = df[['population', 'total_rooms']]
features.head()
```

Out[293…

	population	total_rooms
0	322.0	880.0
1	2401.0	7099.0

	population	total_rooms
2	496.0	1467.0
3	558.0	1274.0
4	565.0	1627.0

In [294…
```
from sklearn.model_selection import train_test_split
# it will return four variables, X_train, X_test, y_train, y_test
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.15
X_train.shape, X_test.shape
```

Out[294… ((17544, 2), (3096, 2))

In [295…
```
y_train.shape, y_test.shape
```

Out[295… ((17544, 1), (3096, 1))

In [297…
```
from sklearn import linear_model
model = linear_model.LinearRegression()
model.fit(X_train, y_train)
```

Out[297… LinearRegression()

In [298…
```
from sklearn.metrics import r2_score
r2 = r2_score(y_train, predictions)
print("R2 Score :", r2)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-298-1da71e80aafc> in <module>
      1 from sklearn.metrics import r2_score
----> 2 r2 = r2_score(y_train, predictions)
      3 print("R2 Score :", r2)

NameError: name 'predictions' is not defined
```

In [ ]:
```
Try 2
Lets try to add more features
```

In [299…
```
features = df[['longitude', 'latitude',         'housing_median_age',    'total_rooms
features.head()
```

Out[299…

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0

In [300…
```
labels = df[['median_house_value']]
labels.head()
```

Out[300…

	median_house_value

	median_house_value
0	452600.0
1	358500.0
2	352100.0
3	341300.0
4	342200.0

In [301…
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.15
```

In [302…
```
X_train.shape, X_test.shape
```

Out[302… ((17544, 8), (3096, 8))

In [303…
```
y_train.shape, y_test.shape
```

Out[303… ((17544, 1), (3096, 1))

In [304…
```
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
dtypes: float64(8)
memory usage: 1.3 MB
```

In [305…
```
features = df[['longitude', 'latitude',         'housing_median_age',    'total_rooms
features.head()
```

Out[305…

	longitude	latitude	housing_median_age	total_rooms	population	households	median_income
0	-122.23	37.88	41.0	880.0	322.0	126.0	8.3252
1	-122.22	37.86	21.0	7099.0	2401.0	1138.0	8.3014
2	-122.24	37.85	52.0	1467.0	496.0	177.0	7.2574
3	-122.25	37.85	52.0	1274.0	558.0	219.0	5.6431
4	-122.25	37.85	52.0	1627.0	565.0	259.0	3.8462

In [306…
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.15
```

In [307…
```
from sklearn import linear_model
model = linear_model.LinearRegression()
model.fit(X_train, y_train)
```

Out[307… LinearRegression()

In [ ]:
```
Lets deal with null values now!!!
```

In [308…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

In [309…
```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='median')
total_bedroom_column = df[['total_bedrooms']]
total_bedroom_column
```

Out[309…

	total_bedrooms
0	129.0
1	1106.0
2	190.0
3	235.0
4	280.0
…	…
20635	374.0
20636	150.0
20637	485.0
20638	409.0
20639	616.0

20640 rows × 1 columns

In [310…
```
new_df = pd.DataFrame(imputer.fit_transform(total_bedroom_column))
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       20640 non-null  float64
dtypes: float64(1)
memory usage: 161.4 KB
```

```
features.loc[:, 'total_bedrooms'] = imputer.fit_transform(total_bedroom_column)
```

In [311…

```
C:\anaconda\lib\site-packages\pandas\core\indexing.py:1596: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  self.obj[key] = _infer_fill_value(value)
C:\anaconda\lib\site-packages\pandas\core\indexing.py:1783: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  self.obj[item_labels[indexer[info_axis]]] = value
```

In [312…

```
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   population          20640 non-null  float64
 5   households          20640 non-null  float64
 6   median_income       20640 non-null  float64
 7   total_bedrooms      20640 non-null  float64
dtypes: float64(8)
memory usage: 1.3 MB
```

In [313…

```
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.15
model = linear_model.LinearRegression()

model.fit(X_train, y_train)
```

Out[313…

```
LinearRegression()
```

In [ ]:

```
One Hot Encoding
```

In [ ]:

```
# [1, 0, 0, 0, 0]
# [0, 0, 1, 0, 0]
# [0, 0, 0, 1, 0]
```

In [314…

```
import sklearn
onehot_encoder = sklearn.preprocessing.OneHotEncoder(sparse=False)
```

In [315…

```
onehot_encoder.fit_transform(df[['ocean_proximity']])
```

Out[315…

```
array([[0., 0., 0., 1., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 1., 0.],
       ...,
       [0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0.]])
```

In [316…

```
onehot_encoder.get_feature_names()
```

Out[316…

```
array(['x0_<1H OCEAN', 'x0_INLAND', 'x0_ISLAND', 'x0_NEAR BAY',
       'x0_NEAR OCEAN'], dtype=object)
```

In [317…
```
df_ocean_proximity = pd.DataFrame(onehot_encoder.fit_transform(df[['ocean_proximity'
```

In [318…
```
df_ocean_proximity
```

Out[318…

	x0_<1H OCEAN	x0_INLAND	x0_ISLAND	x0_NEAR BAY	x0_NEAR OCEAN
0	0	0	0	1	0
1	0	0	0	1	0
2	0	0	0	1	0
3	0	0	0	1	0
4	0	0	0	1	0
…	…	…	…	…	…
20635	0	1	0	0	0
20636	0	1	0	0	0
20637	0	1	0	0	0
20638	0	1	0	0	0
20639	0	1	0	0	0

20640 rows × 5 columns

In [319…
```
features = pd.concat([features, df_ocean_proximity], axis=1)
features
```

Out[319…

	longitude	latitude	housing_median_age	total_rooms	population	households	median_inco
0	-122.23	37.88	41.0	880.0	322.0	126.0	8.32
1	-122.22	37.86	21.0	7099.0	2401.0	1138.0	8.30
2	-122.24	37.85	52.0	1467.0	496.0	177.0	7.25
3	-122.25	37.85	52.0	1274.0	558.0	219.0	5.64
4	-122.25	37.85	52.0	1627.0	565.0	259.0	3.84
…	…	…	…	…	…	…	
20635	-121.09	39.48	25.0	1665.0	845.0	330.0	1.56
20636	-121.21	39.49	18.0	697.0	356.0	114.0	2.55
20637	-121.22	39.43	17.0	2254.0	1007.0	433.0	1.70
20638	-121.32	39.43	18.0	1860.0	741.0	349.0	1.86
20639	-121.24	39.37	16.0	2785.0	1387.0	530.0	2.38

20640 rows × 13 columns

In [320…
```
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.15
model = linear_model.LinearRegression()

model.fit(X_train, y_train)
```

```
LinearRegression()
```

Out[320…

In [321…
```
import seaborn as sns
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(20,10))
sns.scatterplot(x="longitude", y="latitude", hue='ocean_proximity', data=df, ax=ax)
plt.show()
```

In [ ]:
```
DAY15-30
```

In [ ]:
```
Assignment Problem
(1)Fish Market
(2)GPU Kernal Performance Dataset
(3)Vehicle Dataset
(4)New York Stock Exchange
```

In [ ]:
```
Fish Market

Case Study
```

In [323…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```