Python:

```
spam_amount = 0
print(spam_amount)
spam_amount = spam_amount + 4
              [ 0 + 4]
if spam_amount > 0:
    print("But I don't want any spam!")
viking_song = "spam spam spam"
print(viking_song)
```

- Strings can either be marked by double or single quotation marks

Operator	Name	Description
a + b	Addition	
a - b	Subtraction	
a * b	Multiplication	
a / b	Division	
a // b	Floor division	Quotient of a, b.
a % b	Modulus	Remainder
a ** b	Exponentiation	a raised to the power of b
- a	Negation	The negative of a.

- print(5/2)
  O/P → 2.5
- print(5//2)
  O/P → 2

- print (min (1, 2, 3))
  print (max (1, 2, 3))

  O/P → 1
  3

- print (abs (32))
  print (abs (-32))

  O/P → 32
  32

Functions in python:

- def greet():
      print ("hello, world!")

- def greet (name, message = "welcome!"):
      print ("Hello, {name}!, {message}")

  greet (name = "Anshuver") O/P: Hello, Anshuver! welcome!

  greet ("Anshuver", "Good Morning") O/P: Hello, Anshuver!
  Good Morning!

- def add (a, b):
      return a + b

  result = add (3, 5)

  print (result)

-

→

Comparison Operators:

Operation

a == b

a < b

a <= b

a != b

a > b

a >= b

- '3' == 3

  False → O/P

Lists

- primes = [2, 3, 5, 7]

- hands = [

  ['J', 'Q', 'K'],

  ['2', '2', '2'],

  ['6', 'A', 'K'],

  ]

Ordering can be done by:

planets[a]

Slicing

planets[0:3]

→ prints 1st three planets at index 0, 1, 2 & leaves out the 3rd

Pandas:

- Pandas is a powerful & widely used python library for data manipulation & analysis.
- It provides high performance, easy to use data structures like Series & Dataframe.
- import numpy as np
  import pandas as pd
- dict1 = {
  "name": ['harry', 'rohan'],
  "marks": [92, 34],
  "city": ['rampur', 'kolkata']
  }
  Dictionaries are a built in python data structure for mapping keys to value.
- df = pd.DataFrame(dict1)
  # Creates a dataframe / table
- Exporting to excel sheet
  df.to_csv('friends.csv')
- , index = false # to hide the index
- df.head(2) → first 2 rows
- df.tail(2) → last 2 rows
- df.describe()
  →
- to read a csv file
  harry = pd.read_csv('harry.csv')
  harry → shift + enter

- describe() → generates a high level summary of the attributes of the given column. It is 'type aware', i.e its o/p changes based on the data type of the input
- mean() → to see the mean of the points allotted
- unique() → unique value

Maps:
Takes one set of values & "maps" them to another set of values.

O/P → 360

Step	Mult	product (updated)
1	2	1 × 2 = 2
2	2	2 × 2 = 4
3	2	4 × 2 = 8
4	3	8 × 3 = 24
5	3	24 × 3 = 72
6	5	72 × 5 = 360

Range ( )
→ It is a function that returns a sequence of numbers.

```
for i in range (5):
    print ("Doing imp work. i =", i)
```

"while loop" iterates until some condition is met.

```
i = 0
while i < 10;
    print(i, end=' ')
    i += 1
```

O/P 0 1 2 3 4 5 6 7 8 9

- newdf.T

  T → Transpose

- Sort the index :-

  new.df.sort_index(axis=0, ascending=False)

  axis=0 → rows; ascending → or descending

- newdf.loc[0,0] = 654

  newdf.head(2)

- newdf.columns = list("ABCDE")

	A	B	C	D	E
0					
1					

- newdf.drop(0, axis=1)

axis=1 → column
axis=0 → row

iloc

- → Integer based indexing
- → Uses integer positions to locate data
- → Excludes the endpoint in slices (like lists)
- → Works with integer only

loc

- → Label based indexing
- → Uses labels/names of rows & columns
- → Includes
- → Works with labels, booleans, & int.

- planets[3:]

→ Starting from 4th till end

- planets[-3:]

→ The last 3 planets

List functions:

- len(planets) → gives length
- sorted(planets)
- sum()
- max()

List methods:

- planets.append('Pluto') → adds Pluto
- help(planets.append)
- planets.pop()

Lists	Tuples
→ Mutable	→ Immutable
→ Defined using []	→ Defined using {}
→ Slower	→ Faster
→ Dynamic	

Loops:

multiplicands = (2, 2, 2, 3, 3, 5)

product = 1

for mult in multiplicands:

product = product * mult

- harry ['Speed']
  → Speed is a column in harry
- harry ['Speed'][0] = 50
  Speed changed to 50 in row 0
- harry.to_csv('harry.csv')
  → save the changes in csv file
- index 0, 1, 2, 3 → something else
  harry.indix = ['first', 'second', 'third']..

Pandas has two types of DS

Series

→ 1D array with indexes
→ Stores a single column or row of data in a DF

Data Frame

→ Tabular spreadsheet like structure representing rows each of which contain one or multiple column.

- ser = pd.Series(np.random.rand(34))
  type(ser)
- newdf = pd.DataFrame(np.random.rand(334, 5), index=np.arrange(334))

newdf.head()

- newdf.dtypes

→

What you type	what you get	example - print(ex)
\'	'	'what\'s up? → what's up
\"	"	"That's "cool\" " → That's "cool"
\\	\	"Look, a maintain: /\\" → Look, a maintain: /\
~~\n~~		
\n		"1\n2 3" → 1 2 3

String Methods:

- claim.upper()
- claim.lower()
- claim.index('plan')
- claim.startswith('planet')

O/P → True