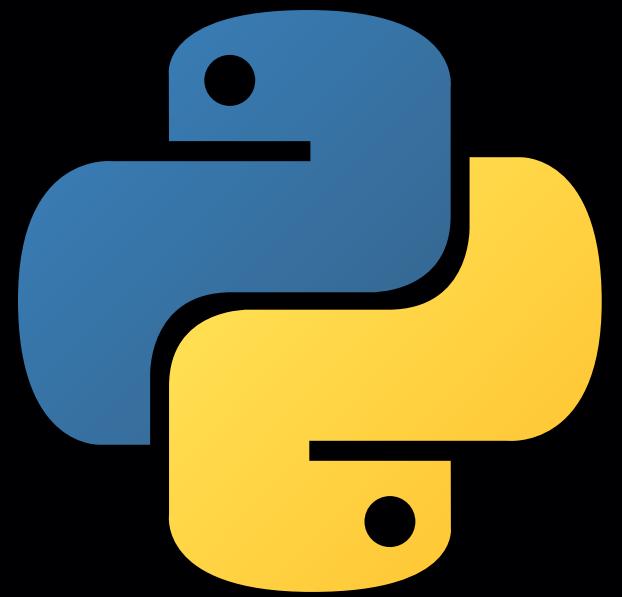


MPRASHANT



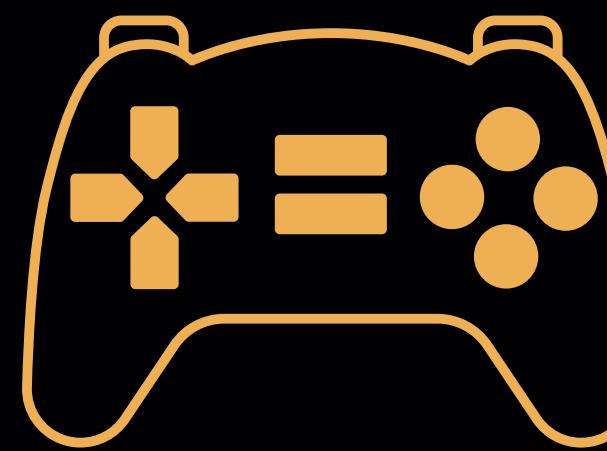
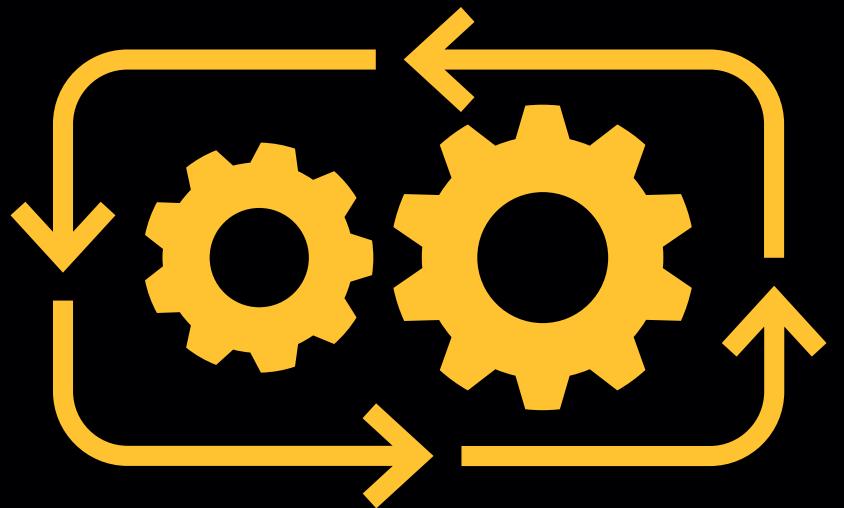
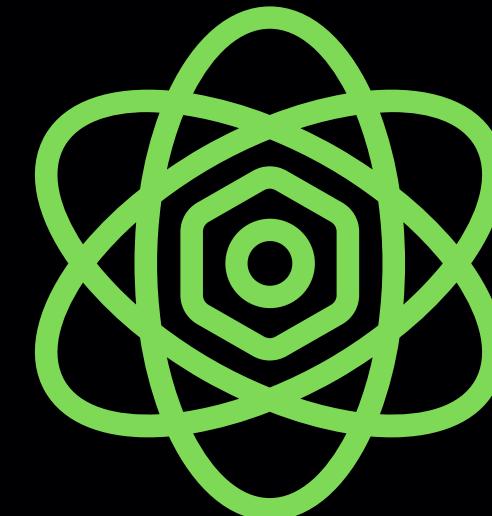
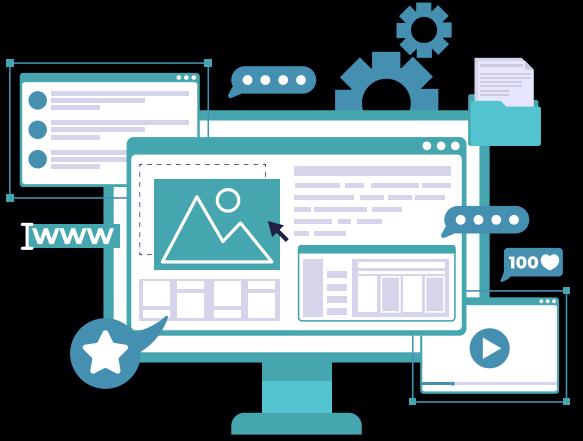
# PYTHON SCRIPTING

MPRASHANT



# Why Python?

- Easy to read and learn
- Various existing libraries and frameworks
- Large community and documentation
- Automate our day to day task
- Can create GUI based apps, websites and games



MPRASHANT

# Python Installation & Setup



# python™

Do

About

Downloads

Documentation

Community

## Download the latest version for macOS

[Download Python 3.11.5](#)

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [macOS](#),  
[Other](#)

Want to help test development versions of Python 3.12? [Prereleases](#), [Docker images](#)

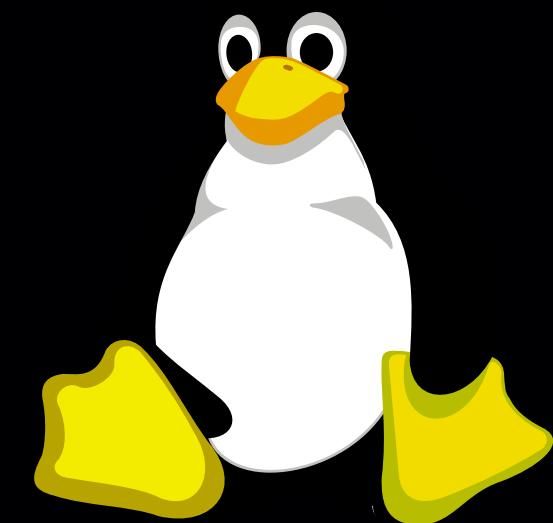
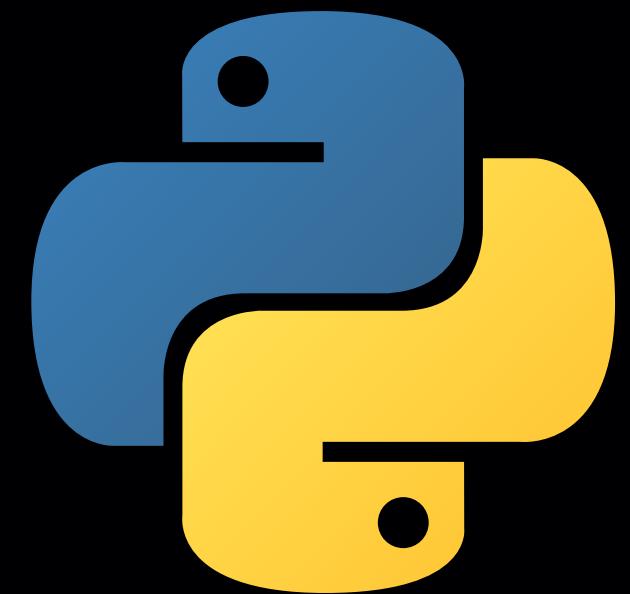


The Python IDE  
for Professional  
Developers

[DOWNLOAD](#)

Full-fledged Professional or Free Community

# Python on LINUX



MPRASHANT

SENDING OUTPUT  
TO TERMINAL..

```
print("Hello World!")
```



# MPRASHANT

# COMMENTS

# Using #

```
#This is a singleline comment
```

# Multi-line comment

11 / 11

first line

your comment here

# last line

1 / 1

# Variables



# WHAT ARE VARIABLES?

```
name="Paul"  
print(name)
```



# VARIABLES RULE

- Should not start with number
- No space
- No special character

# TYPES OF DATA IN VARIABLES?

`name="Paul"`  
`age=30`  
`price=25.5`  
`bol=True/False`

# WHAT ARE VARIABLES?

Check type of data

```
print(type(age))
```



# ARITHMETIC OPERATIONS?



a=10

b=2

`print(a+b)`

We can perform

+,-,/,\*

# ARITHMETIC OPERATIONS

// AND \*\*

To find the  
 $2^5$   
`print(2**5)`

# ARITHMETIC OPERATIONS

$$2 \times (3 + 4)^2 \div 5 - 6$$

**BODMAS rule**



# PRINTING STRING VARIABLE

```
name="Paul"
```

```
print(f'My name is {name}')
```

# String Operations

**Strings are immutable**

We can use ‘Hello’ or “Hello”

Use + to add two strings ‘Hello’ + ‘World’

# STRING OPERATIONS



`msg="Hello World!"`

`msg.lower()`

`msg.upper()`

`msg.title()`

`msg.split()`

# Lists

- Ordered Sequence of different types of values

How to define a List?

```
users=['raju', 'sham', 'paul']
```

# LISTS

How to get values from a list?

```
users[0]
```

```
users[1]
```

```
users[-1]
```



# LISTS MODIFY



Adding item to a List?

`users.append('alex')`

How to delete values from a list?

`users.remove('raju')`

How to modify values from a list?

`users[1] = "Shan"`

# LISTS MODIFY



Adding item to a List at given position?

```
users.insert(1, 'shetu')
```

How to delete values from a list using  
index no?

```
del users[2]
```

Length of list

```
len(users)
```

# LISTS SORTING



Sorting the items in list?

`users.sort()`

`users.sort(reverse=True)` or

`users.reverse()`

Temporary sorting of items?

`print(users.sorted())`

# LISTS

Popping the items in list?

`users.pop()`

`users.pop(2)`



# SLICING LISTS



Getting first two items

`users[:2]`

Getting middle 3 items

`users[1:4]`

Getting last 2 items

`users[-2:]`

# NUMERIC LISTS



```
sell=[2, 9, 13, 28, 35, 4, 56]
```

Getting min or max

`min(sell)`

`max(sell)`

Getting sum of items

`sum(sell)`

# Tuples

# TUPLES

Same as list, however you can't  
modify the items of Tuple.

Immutable

Ordered

```
days= ('mon', 'tue', 'wed')
```

# Dictionaries

key:value



# DICTIONARY

```
car={'brand':'Audi', 'model':'Q3'}
```

```
print("Brand of car is " + car['brand'])
```

```
print("Model of car is " + car['model'])
```



# DICTIONARY

```
car.get('brand')
```

By using the get() function, if key  
is not present it return none rather  
than error.



## DICTIONARY MODIFY



Adding item to a Dictionary?

```
car['color']="Red"
```

How to delete values from a  
dictionary?

```
del car['brand']
```



# DICTIONARY LENGTH

Getting no. of key-value a  
Dictionary?  
`len(car)`

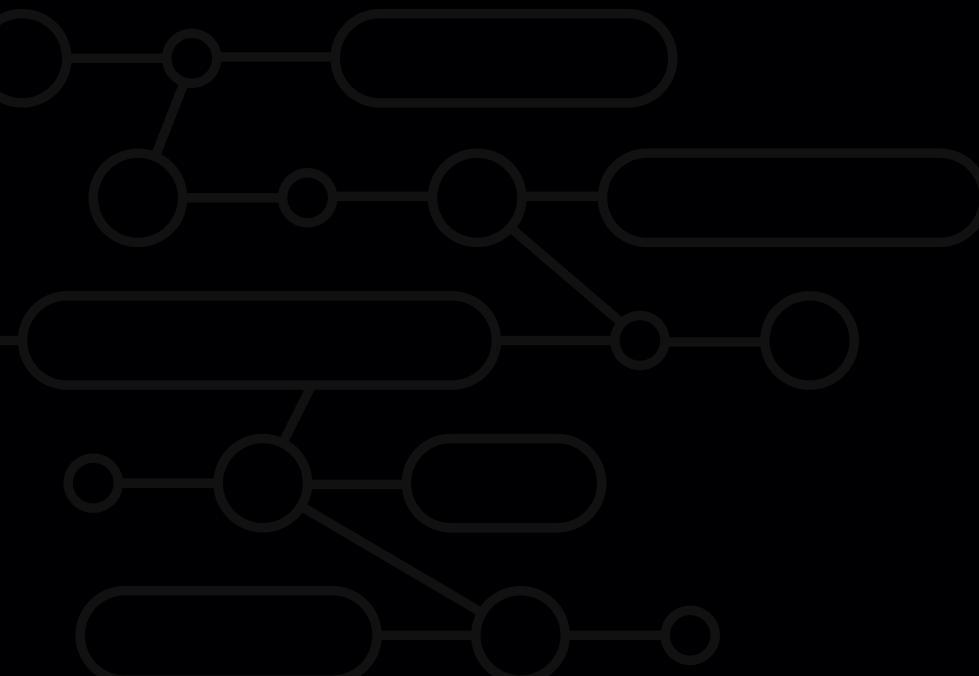


# Sets

Unordered and Unique elements

# SETS

```
my_set = {'A', 'B', 'C'}
```



# User Interaction

# TAKING INPUT FROM USER...

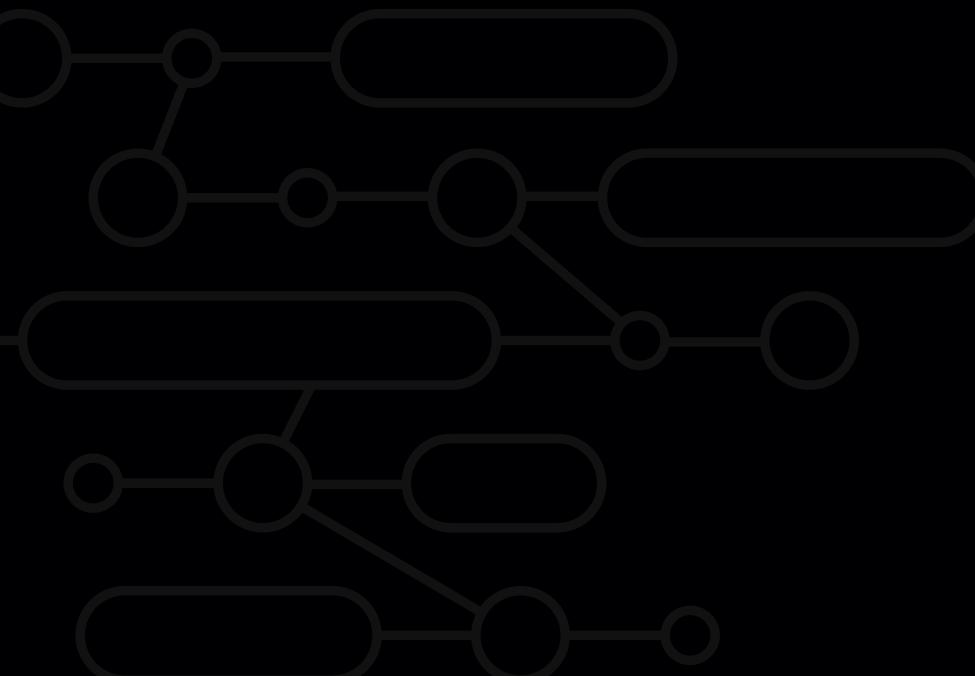
```
name = input("What's your name? ")  
age = int(input('Your age'))
```

# Conditional Statement



## IF-ELSE

```
if age >= 18:  
    print("You can vote!")  
else:  
    print("You can't vote!")
```





## IF-ELSE



```
users=['paul', 'raju', 'sham']  
if 'paul' in users:  
    print("User Exist")
```

---

```
elist=[]  
if elist:  
    print('List is not empty')  
else:  
    print('List is empty')
```

# OPERATORS



<b>Equal</b>	$x == 20$
<b>Greaterthanorequalto</b>	$x >= 20$
<b>Lessthanorequalto</b>	$x <= 50$
<b>Not Equal</b>	$x != 20$
<b>Greater Than</b>	$x > 20$
<b>Less Than</b>	$x < 20$



ELIF



```
if marks >= 80:  
    print("1st Division")  
  
elif marks >= 60:  
    print("2nd Division")  
  
else:  
    print("Failed!")
```

# If-else Nesting

# Logical Operators

## and, or

# LOGICAL OPERATORS

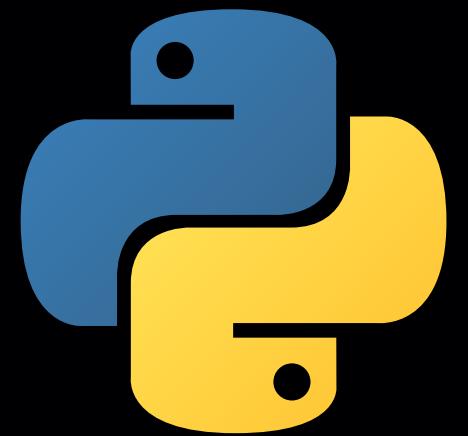


**condition1 and condition2**

If both conditions are true then  
true else false

**condition1 or condition2**

If any of the condition is true  
then true



# For Loop

**Execute a block of code for each item in the sequence  
(such as a list, tuple, string, or range)**

```
for i in 1,2,3,4,5:  
    print(i)
```

# FOR LOOP



Other ways to write For loop

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

## Other ways to write For loop

```
age_info={'Raju':25, 'Sham':30}
```

```
for name, age in age_info.items():
    print("Age of " + name +
          "is " + age)
```

```
for name in age_info.keys():
    for age in age_info.values():
```

FOR LOOP



# FizzBuzz Program

- The program should add numbers from 1 to a specified number in a list.
- For multiples of 3, it should be "Fizz,"
- For multiples of 5, it should be "Buzz,"
- For numbers multiples of both 3 and 5, it should be "FizzBuzz."
- Print the list

# While Loop

count=1

```
while count <= 5:  
    print(count)  
    count += 1
```

---

```
msg=''  
while msg != 'quit':  
    msg = input("Your message..")  
    print(msg)
```

# WHILE LOOP



# WHILE LOOP

**active=True**

**while active:**

```
msg = input("Your message..")  
if msg == 'exit':  
    active=False
```

# WHILE LOOP

```
num=[1, 29, 20, 29, 40, 50, 29]
```

```
while 29 in num:  
    num.remove(29)
```

# USEFUL CONCEPTS...

**break** - to stop the loop

**continue** - to stop current iteration  
of loop and start next iteration





BREAK



msg= ''

while True:

    msg = input("Your message..")  
    if msg == 'exit':

        break

    else:

        print(msg)

MPRASHANT



CONTINUE



```
users=[]
name=''

while True:
    name = input("Your name..")
    if msg == 'exit':
        break
    elif name in users:
        print("User already exist")
        continue
    else:
        print(name)
```

# Functions

## WHAT ARE FUNCTIONS?

- **Block of code which perform some task and run when it is called.**
- **Can be reuse many times in our program which lessen our lines of code.**
- **We can pass arguments to the method**

# HOW TO MAKE FUNCTIONS?



## Defining a Function

```
def myfun():
```

```
    print("Function is called")
```

## Calling a Function

```
myfun()
```

# ARGUMENTS PASSING FUNCTIONS?

## Defining a Function

```
def user_info(name, age):  
    print("Name is "+ name)  
    print(f"Age is {age}")
```

## Calling a Function

```
user_info('raju', 30)
```

# DEFAULT VALUES FUNCTIONS?

## Defining a Function

```
def user_info(name, age=20):  
    print("Name is " + name)  
    print(f"Age is {age}")
```

## Calling a Function

```
user_info('sham')
```

# NONE ARGUMENT FUNCTIONS?

## Defining a Function

```
def user_info(name, age=None):  
    print("Name is " + name)  
    if age:  
        print(f"Age is {age}")
```

## Calling a Function

```
user_info('sham')
```

# RETURN RESULT IN FUNCTIONS?

```
def addition(num1, num2):  
    sum = num1 + num2  
    return sum  
  
print(addition(10, 20))
```



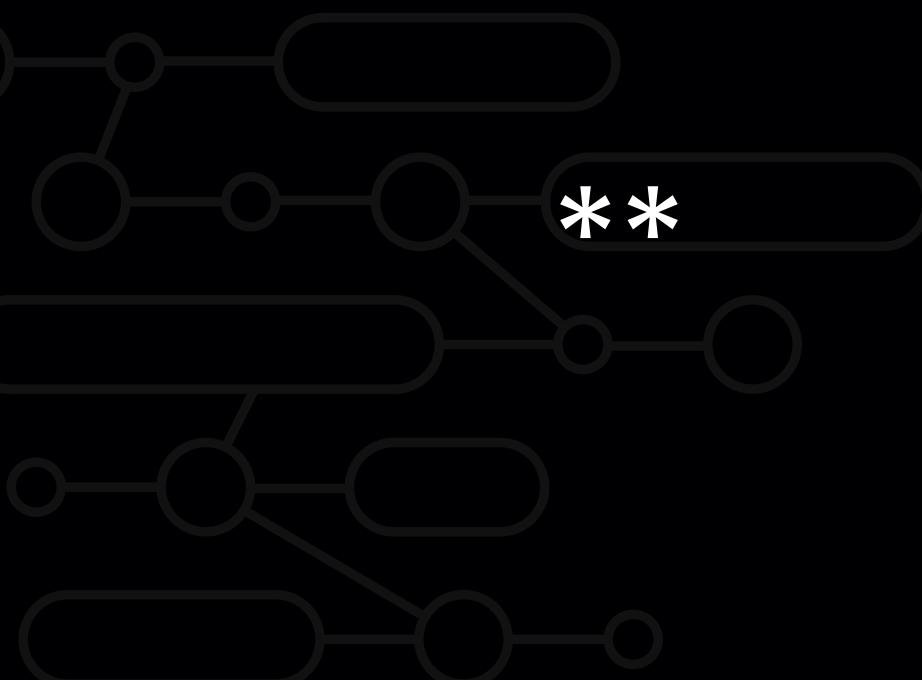
# DYNAMIC ARGUMENTS FUNCTIONS

```
def car_detail(brand, *features):
    print(f"Car brand - {brand}")
    for feature in features:
        print(f" - {feature}")
```

```
car_detail('Audi', 'Sunroof')
```

```
car_detail('Tata', '360cam', 'safe')
```

# DYNAMIC ARGUMENTS FUNCTIONS



```
def user_detail(name, **user_info):
    print(f"Name - {name}")
    for key, value in user_info.items():
        print(f" - {key}: {value}")

user_detail('Raju', age=18)
user_detail('Sham', age=18, city='Delhi')
```

Make a file named **cal.py**

# FUNCTIONS AS MODULE

```
def addition(num1, num2):  
    sum = num1 + num2  
    return sum
```

---

```
import cal
```

```
cal.addition(10, 20)
```

**We can pass list as argument in a function**

# Object Oriented Programming

Classes

Objects

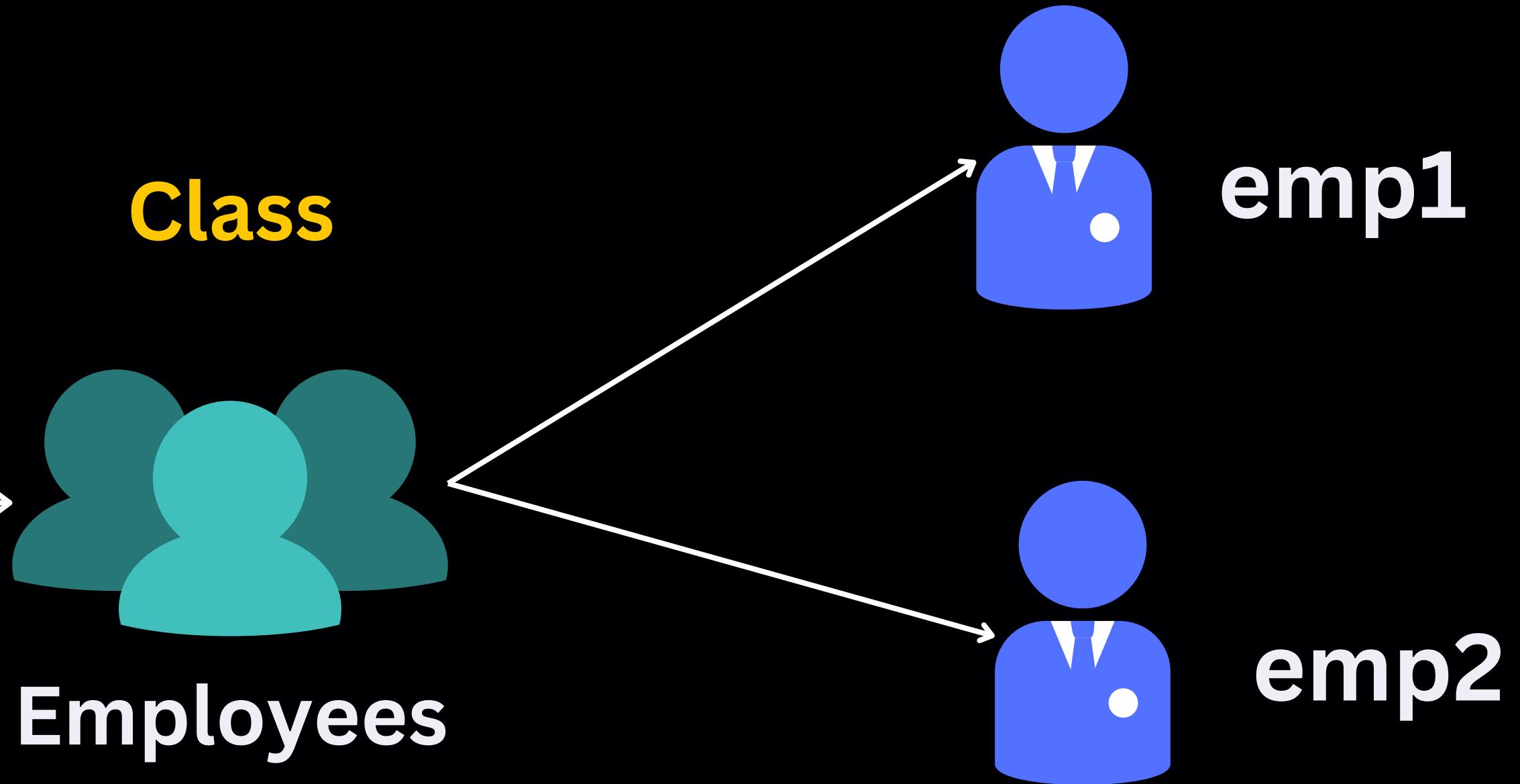
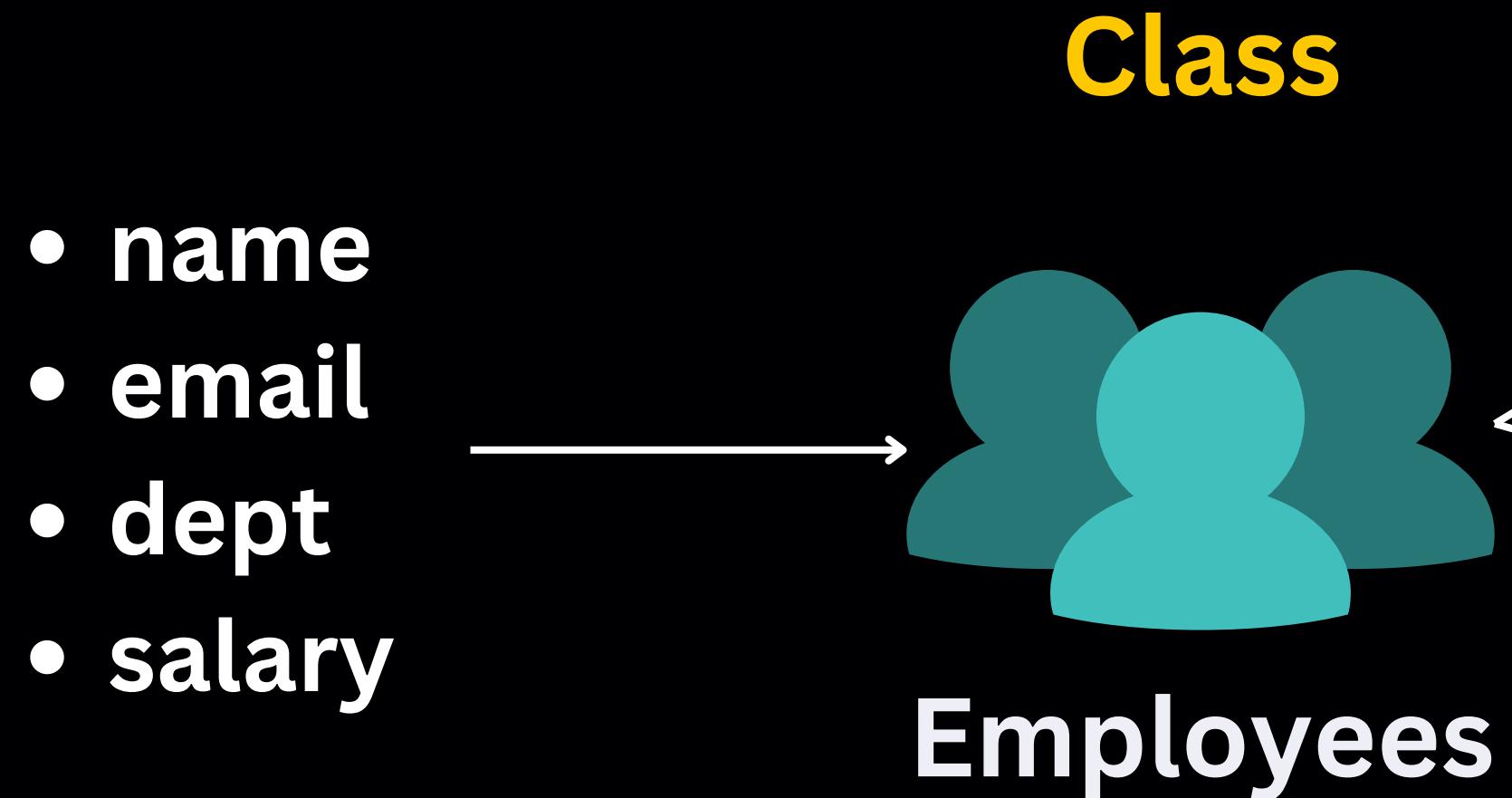
# Classes

A class is a blueprint or a template for creating objects and it defines the structure and behaviour of that objects.

# Objects

An object is an instance of a class. with its own unique data and the ability to perform actions

# Objects



```
class Employee:  
    def __init__(self, emp_id, name, department, salary):  
        self.emp_id = emp_id  
        self.name = name  
        self.department = department  
        self.salary = salary  
  
    def display_info(self):  
        print(f"Employee ID: {self.emp_id}")  
        print(f"Name: {self.name}")  
        print(f"Department: {self.department}")  
        print(f"Salary: {self.salary}")  
  
    def change_department(self, new_department):  
        print(f"{self.name} is moving to the {new_department} department.")  
        self.department = new_department
```

# Creating Objects

```
# Creating instances of the Employee class
employee1 = Employee(101, "Alice", "HR", 50000)
employee2 = Employee(102, "Bob", "IT", 60000)

# Accessing attributes and methods of the instances
print(employee1.name) # Output: Alice
print(employee2.department) # Output: IT

employee1.display_info()
employee2.change_department("Finance")
```

# Modifying Attributes

```
# Modifying Attributes  
employee1.department="Finance"  
print(employee1.department)
```

# Working with FILES



# Writing into a File

```
with open('test.txt', 'w') as fo:  
    fo.write("This is test message.")
```

```
with open('test.txt', 'a') as fo:  
    fo.write("Adding new line to file.")
```

# Reading from a File (whole content)

```
filename = "user.txt"

with open(filename) as fo:
    content = fo.read()

print(content)
```

# Reading from a File (line by line)

```
filename = "user.txt"

with open(filename) as fo:
    lines = fo.readlines()

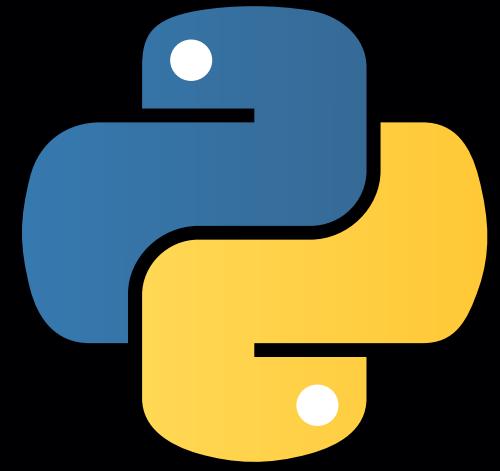
for line in lines:
    print(line.rstrip())
```

# Exceptional Handling

```
try:  
    print(10/0)  
except ZeroDivisionError:  
    print("Kindly do not divide by zero")
```

```
try:  
    with open("hola.txt") as f:  
        content = f.read()  
except FileNotFoundError:  
    print("File is not found")
```

```
try:  
    with open("hola.txt") as f:  
        content = f.read()  
except Exception as e:  
    print(e, type(e))
```



# Python Modules & Packages

# Python Built-in Module

- random
- datetime

```
import random

# Generate a random float between 0 and 1
random_float = random.random()
print(random_float)
```

```
import random

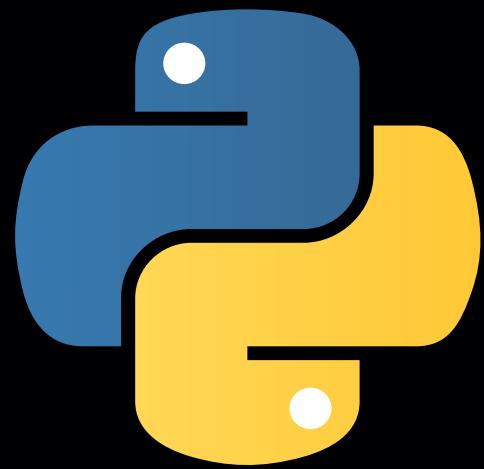
# Generate a random integer between 1 and 100
random_integer = random.randint(1, 100)
print(random_integer)
```

```
import random

# Generate a random choice from a list
options = ["apple", "banana", "cherry", "date"]
random_choice = random.choice(options)
print(random_choice)
```

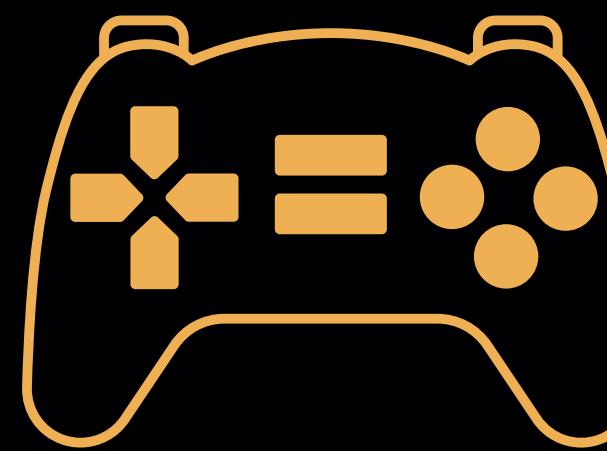
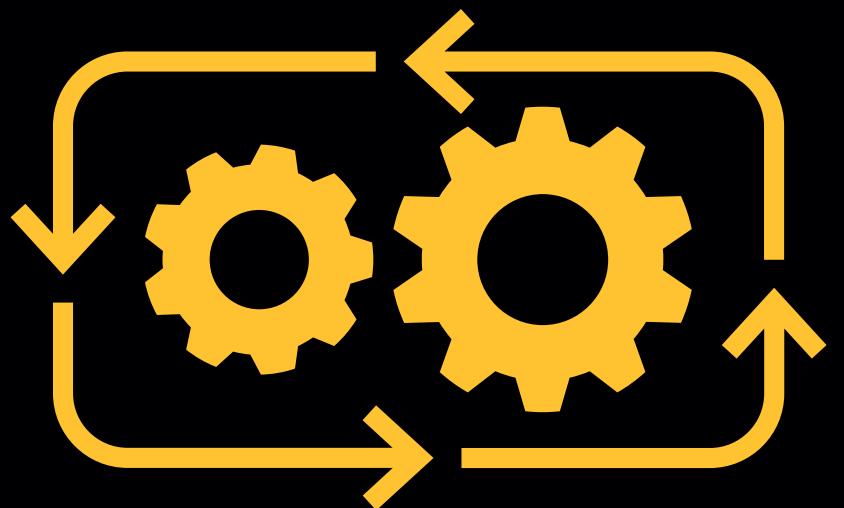
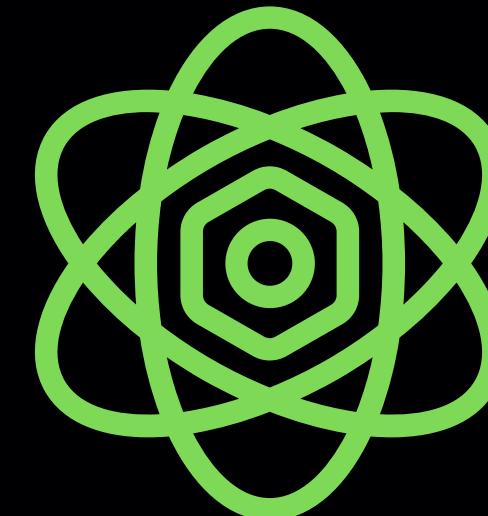
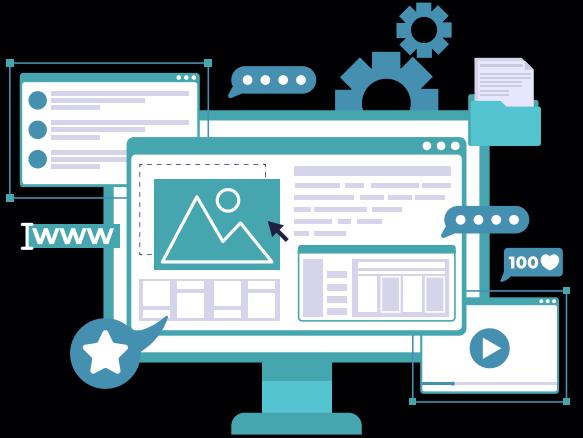
# Dice Game





# What is PyPI & PiP







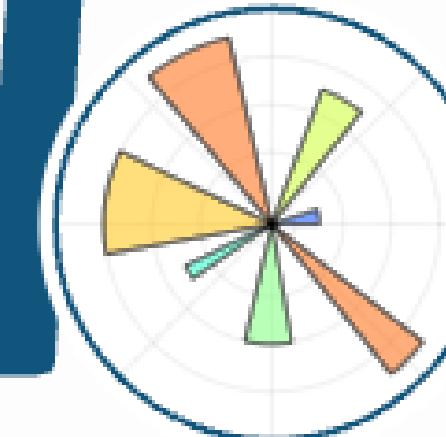
Most powerful and flexible open source  
**data analysis / manipulation tool**

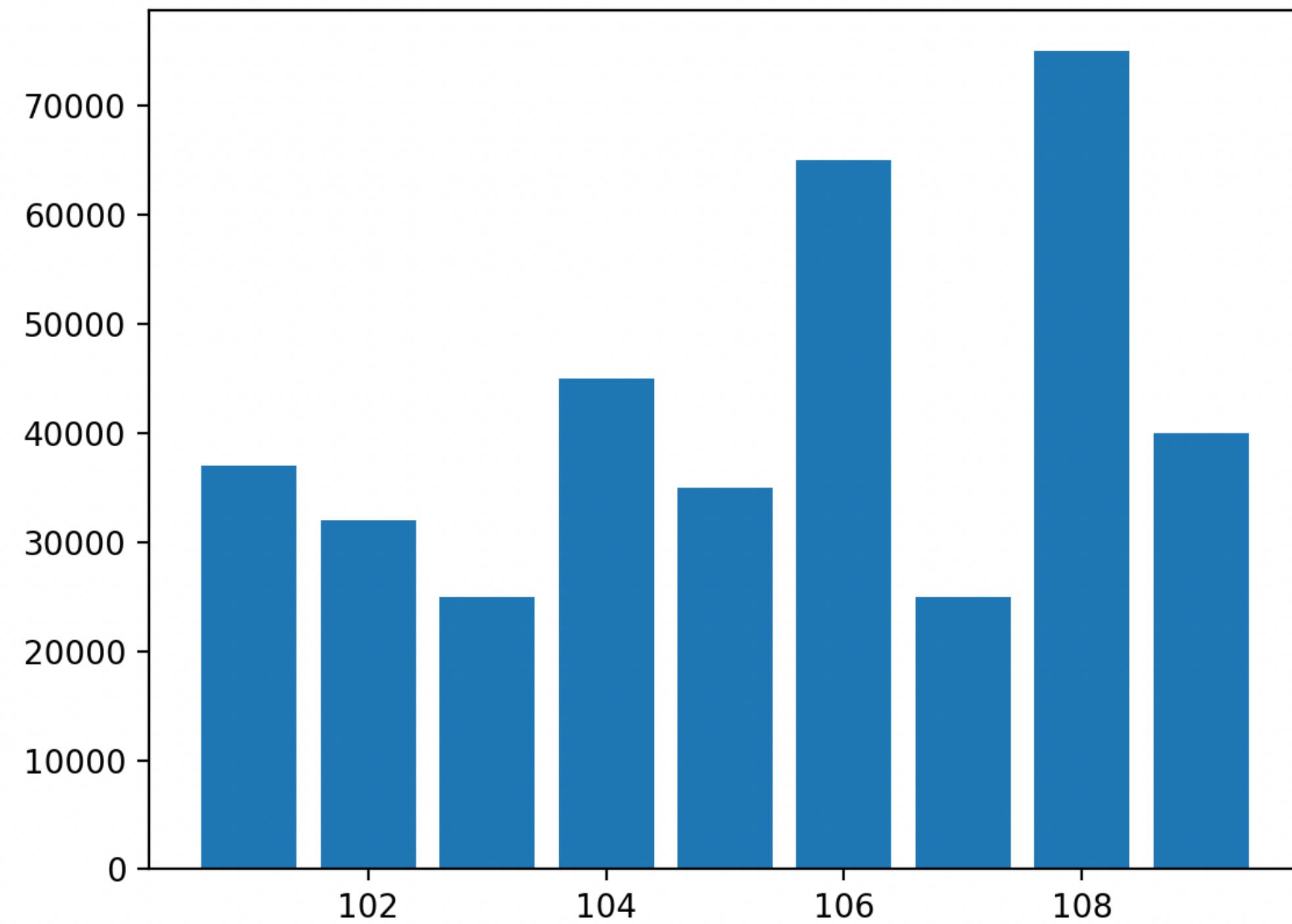
emp_id	fname	lname	desig	dept	salary
101	Raju	Rastogi	Manager	Loan	37000
102	Sham	Mohan	Cashier	Cash	32000
103	Baburao	Apte	Associate	Loan	25000
104	Paul	Philip	Accountant	Account	45000
105	Alex	Watt	Associate	Deposit	35000
106	Rick	Watt	Manager	Account	65000
107	Leena	Jhonson	Lead	Cash	25000
108	John	Paul	Manager	IT	75000
109	Alex	Watt	Probation	Loan	40000

- Print the csv file
- Print only salary column
- Find min and max salary
- Find data of employee with
  - Emp\_id 105
  - Max salary
- Find total & average salary of employees
- Print full name of employee id 103

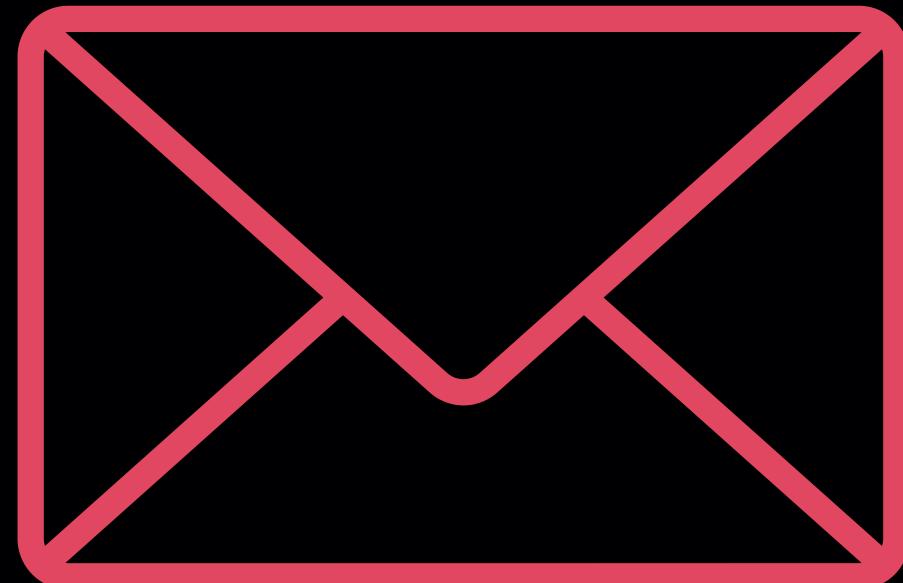
- Change the salary of Sham to 80000
- Remove the data of Leena
- Sort the data with salary (min to max)
- Add a new column ‘bonus’ with value 10%  
of salary
- Drop the ‘salary’ column

# matplotlib





# Working with EMAILS



```
import smtplib

hostname = 'smtp.gmail.com'
email = '<your_email>'
password = '<your_password>'

with smtplib.SMTP(host=hostname) as connection:
    connection.starttls()
    connection.login(user=email, password=password)
    connection.sendmail(
        from_addr=email,
        to_addrs=email,
        msg=f'Subject: Test Python Email\n\n Hi Paul, This is Test Email'
    )
```



# Weather Checking App

Working with API

```
import requests

API_KEY = '<your_api_key>'
city_name = input('Enter the name of the city: ')

url = f'https://api.openweathermap.org/data/2.5/weather?q={city_name}&appid={API_KEY}'

response = requests.get(url)

if response.status_code == 200:
    weather_data = response.json()
    weather_disc = weather_data['weather'][0]['description']
    temp = weather_data['main']['temp'] - 273.15

    # Display weather info
    print(f'Weather in {city_name}: {round(temp, 2)} °C with {weather_disc}')
else:
    print(f'City name {city_name} is not found or incorrect')
```

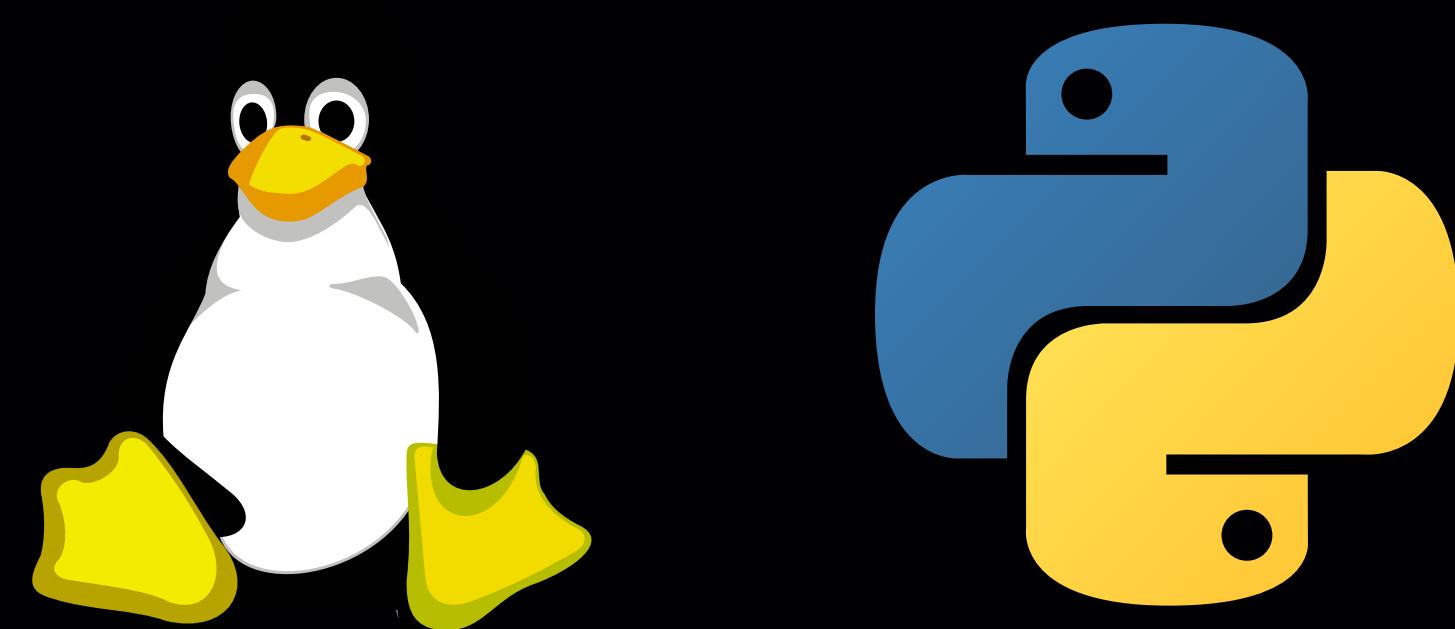
|

# Assignment

## Create a Calculator Program

```
Enter the first number: 10
Enter the next number: 20
+
-
/
*
Select the operation: +
Result of 10 + 20 is 30
Continue the operation with 30? y/n: y
Enter the next number: 50
+
-
/
*
Select the operation: +
Result of 30 + 50 is 80
Continue the operation with 80? y/n: |
```

# Python Program to Run Linux Commands



MPRASHANT

THANK YOU

