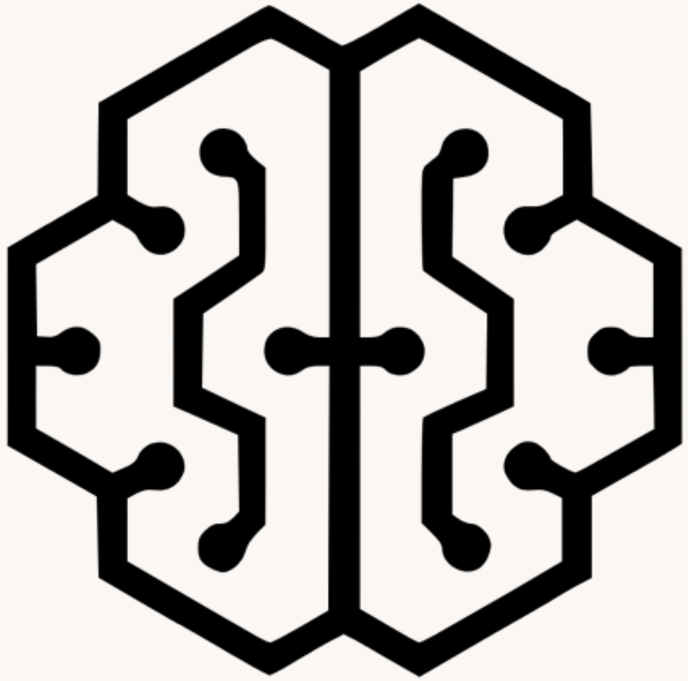
Indian Institute of Technology Madras



AI CLUB


# Intro to PYTHON

# Table of CONTENTS

# INTRODUCTION

Python is a programming language created by Guido Van Rossum

***ADVANTAGES OF PYTHON:***

1. **Simple Syntax** : Easy to code, read and understand
2. Has a lot of libraries and modules created by third parties , extending its functionality for various tasks
3. **Dynamically typed** - no need to declare data types explicitly, making it flexible
4. **Object Oriented Programming Language**

***BUT WHY USE FOR ML?***

Python has libraries like **NumPy** (for numerical calculations) and **Pandas** (for analyzing data)

# Basics :

- Variables: storage containers for values.
- This is how we declare variables:

```
Name = "Anupama"
Math_Mark = 93.5
age = 17
```

- We can display these variables using the print statement:
- The syntax is : print(<output you want to be displayed>)

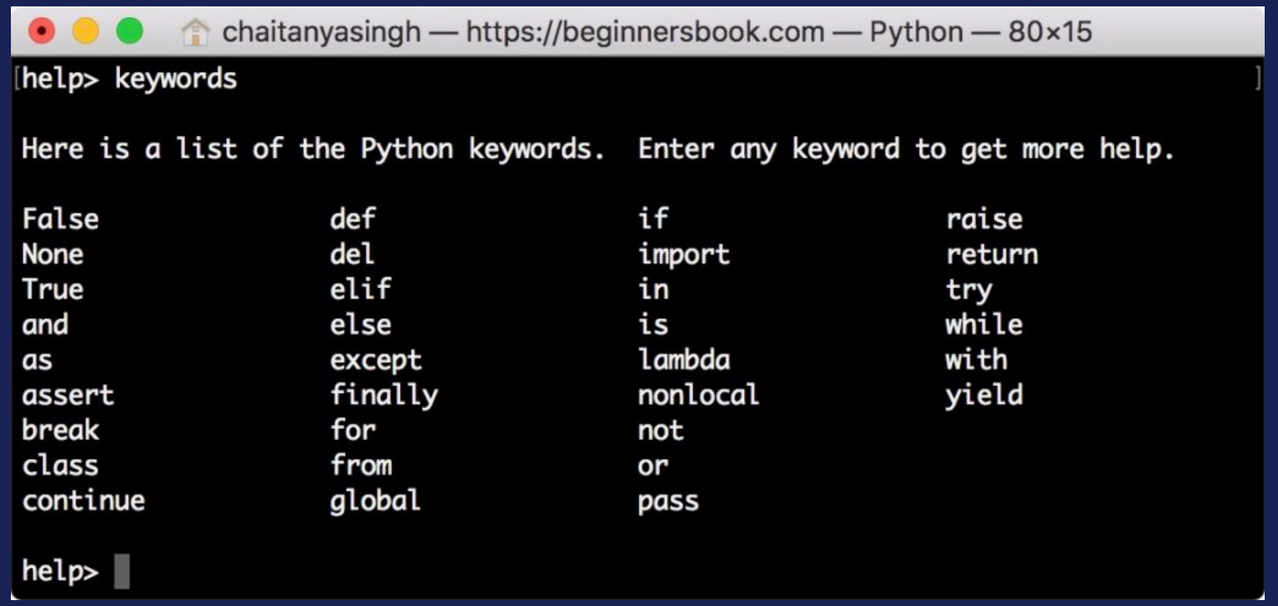
```
print("Details are :", Name, age, Math_Mark)
```

```
Details are : Anupama 17 93.5
```

```
[6]  class = "12"     # Raises an error
```

```
  File "<ipython-input-6-8b945fceb5b3>", line 1
    class = "12"     # Raises an error
          ^
SyntaxError: invalid syntax
```

- Keywords:  reserved words (that have a specific meaning), can not be used as a variable name, function name.



# Formatted Strings:

```
print(f"My name is {Name}")  #Another way to print an output
```

```
My name is Anupama
```

# Comments:

```
0s
#Lines of code that are green(followed by #) are ignored by the interpreter.
#These are called comments and they are used to add information next to the code
```

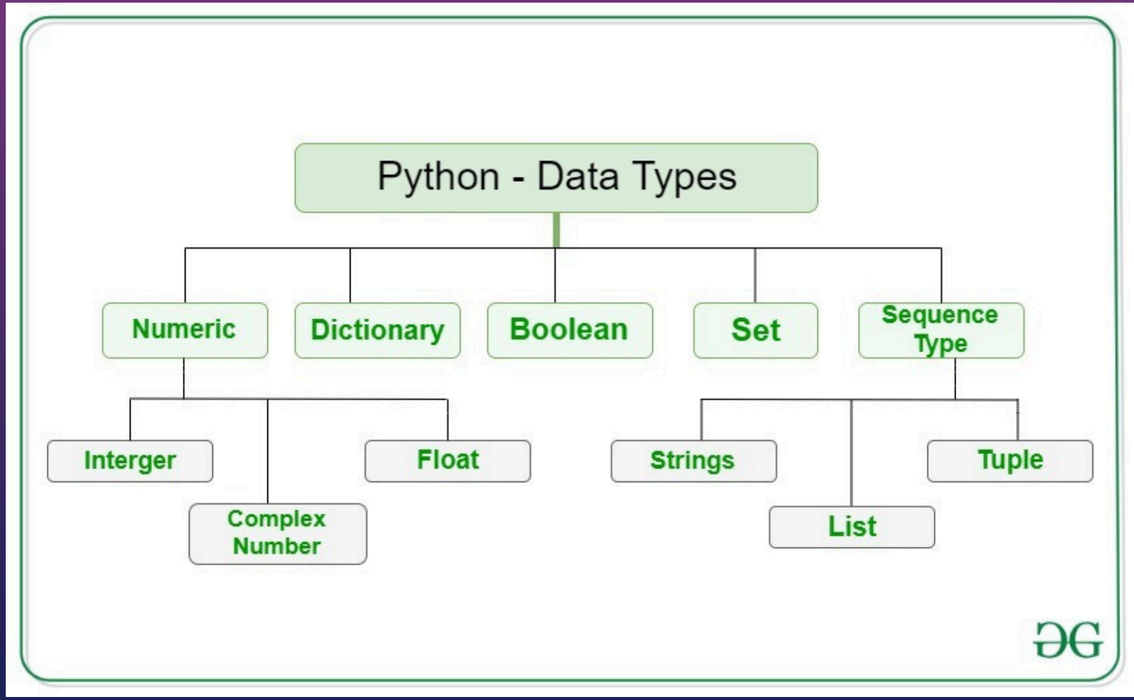
- Displays no output, when we run the cell.

# DATA TYPES

- Datatypes are the categorization of data items. Variables can store data of different types
- Operations that are performed on the data depend on the data type

## BUILT-IN DATA TYPES

- Int --- integers
- Float -- decimal values
- String -- characters, words or sentences
- Boolean - True or false
- None Type



# DATA TYPES...

```
[13] x = 10  #int
     y = 34
     decimal = 1.23 #float
     z = y < x
     Name = "Preethana"     # When assigning variables to strings, the string should be in double quotations

     # how to see what these variables store
     print(x, y, Name, z )
```

```
10 34 Preethana False
```

```
#how to see the data type these variables store
print( "we see data type using type() fucntion")
print(type(x))
print(type(y))
print(type(decimal))
print(type(Name))

# python automatically detects the data type of the variable
```

- Using the type function we can see the data type of the variable

```
we see data type using type() fucntion
<class 'int'>
<class 'int'>
<class 'float'>
<class 'str'>
```

## **Type Casting:** Converting data from one type to another

```
a = float(1)        # a will be 1.0
b = str(5.6)      # note use str not string
c = int(5.6)      # c will be 5          NOTE: The number is NOT rounded off but the decimal part is stripped away

print(a, type(a))
print(b, type(b))
print(c, type(c))
```

```
1.0 <class 'float'>
5.6 <class 'str'>
5 <class 'int'>
```

- **Input statement:** Used when we want to take input from the user
- Syntax: input(“<Enter message to be displayed>”). Default data type (of input) : str

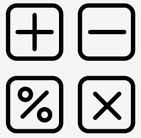
```
class_name = input("enter the class")
print("You are in class", class_name)

type(class_name)
```

```
enter the class12
You are in class 12
str
```

```
[ ] ## Supoose we only want to input an integer, then we type cast as int
    class_name = int(input("enter the class"))
    print("You are in class", class_name)
    type(class_name)
```

```
enter the class12
You are in class 12
int
```



Arithmetic Operators

+, -, /, *,**,%

>, <, ==, >=, <=
Comparison Operators

Logical Operators

AND, OR, NOT

# Operators

Assignment Operators

=, +=, -=, %=

**Perform different processes on variables and constant**

- **OPERATORS:** These are the special symbols. Eg- + , * , /, etc.
- **OPERAND**: It is the value on which the operator is applied.

## Arithmetic Operators:

1. Addition (+)
2. Subtaction (-)
3. Multiplication(*)
4. Division (/)
5. Modulus or remainder (%)
6. Exponent (**)

```
a = 5
b = 6

# operations in order

print('a + b = ', a + b)
print('b - 2 = ', b - 2)
print('a*4 =', a * 4)
print('b/3 = ', b/3)
print('a % b = ', a%b) # 5/6 q=0 rem=5
print('a^2 = ', a**2 )    # Don't forget: exponent is not ^, it is **
```

```
a + b =  11
b - 2 =  4
a*4 = 20
b/3 =  2.0
a % b =  5
a^2 =  25
```

## Comparison Operators

1. Check equality (== )
2. Check inequality(!= or <>)
3. Greater than (>)
4. Less than(<)
5. Greater than or equal to (>=)
6. Less than or equal to (<=)

When used in expressions, the truth value of the expression returned.

```
a = 5
b = 6

#operations in order

print(a==b)
print(b!=2)
print(a>4)
print(b<3)
print(a>=b)
print(a<=2)
print(type(a==b))
```

```
False
True
True
False
False
False
<class 'bool'>
```

## Logical Operators

1. and
2. or
3. not

When used in expressions, the truth value of the expression returned.

```
[ ] a = 5
    b = 6
    print(a<b or a==b) # t or f
    print(a<b and (a+b)==11) # t and t
    print(not a<b) # not t
```

```
True
True
False
```

1. or : Yields false only when both are false
2. and : Yields true only if both conditions are true
3. not: Complements true to false and false to true

## Assignment Operators: used to assign values to a variable

e.g: =, +=, -=, %=

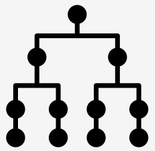
```
[ ] x = 2
    print(x)
    x += 2   # equivalent to x = x + 2
    print(x)
```

```
2
4
```

- None Data type:  None Keyword is used to define a null value, or no value at all.
- None is not the same as 0, False, or an empty string. None is a data type of its own (NoneType) and only None can be None.

```
0s
    a = None
    b = a or 4
    c = None and 4
    print(a, type(a))
    print(b, type(b))
    print(c, type(c))
```

```
None <class 'NoneType'>
4 <class 'int'>
None <class 'NoneType'>
```



# Conditional STATEMENTS

- Used to make decisions in python

**Keywords**:

- if, elif, else
- we can use either “ if statement” or “if..elif.. else” or “if...else”
- Sometimes there can be conditional statements inside conditional statements:: **Nested Conditional blocks**

**Syntax and structure:**

## ‘if’ <condition>:

- If the following condition is true the block under it gets exectuted

## ‘elif’ <condition> :

- Comes to this only  if the if statement is false
- There can be multiple elif blocks

## ‘else’  :

- It accounts for the remaining possibilities
- If none of the above conditions are true then none of the above blocks are executed and the else block executes

```
Age = 17
 # simple if....else
if Age >= 18:
  print("you can vote")
else:
  print("You can't vote")
```

```
You can't vote
```

## Nested Conditional Statements:

```
## if number is positive or negative

i = -15

if i != 0:
    if i > 0:
        print("Positive")
    # condition 3
    elif i < 0:
        print("Negative")
elif i == 0:   #use == as we are comapring i and 0, not assigning i as 0
  print("zero")
else:
  print("invalid, checking only numbers")
```

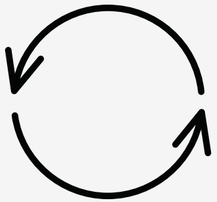
```
Negative
```

```
Summer = False
Winter = True
Rainy = False

if(Summer):
  print("Lets eat ice cream!")
elif(Winter):
  print("Lets play with snow!")
elif(Rainy):
  print("Lets drink Chai!")
else:
  print("Lets just go and study:/")
```

```
Lets play with snow!
```

# LOOPING statements



- Looping is to repeatedly execute a block of statements until a particular condition is satisfied.
- **Iteration:** Go through the loop
- **Nested Loops:** Are loops within loops .

01

## For loop

- When the number of iterations are known.
- **SYNTAX:**

```
for < condition>:
        <statements to execute>
```

02

## While loop

- When the number of iterations is not known.
- The loop is terminated based on a given criteria.
- **SYNTAX:**

```
while < condition>:
        <statements to execute>
```

- **FOR loop:**

```
for i in range(1,10):       # range function gets a range of integers from 1 to 9 (the last number is not included)
# for i in (1,2,3, 4, 5 ,6, 7, 8, 9)
  print(i)      # prints all numbers from 1 to 9
```

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

**Break statement is used to break out of the loop:**

```
for i in range(1,10):
    print (i)
    if i > 5:
      break           # this break statements breaks off from the loop
      #this prints from 1 to 6, because only after the print statement it checks if i > 5 and breaks away from the loop
```

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

- **While loop:**

```
number = int(input('Enter a number: '))

total = 0

# iterate until the user enters 0
while number != 0:
    total += number
    number = int(input('Enter a number: '))

print('The sum is', total)
```

**o/p:**

```
Enter a number: 12
Enter a number: 45
Enter a number: 89
Enter a number: 0
The sum is 146
```

# FUNCTIONS

- Functions are used to accomplish specific tasks.
- They are defined by the user .
- Executed only when they are called.
- Can be called multiple times
- Takes in input through parameters, processes it and gives o/p.

- Some important aspects of the code:

def ---> first we have to define the name of the function along with the parameters involved

arguments------> values given in place of parameters by user, while calling the function

return --------> Optional, if used then the function returns the output of the return statement

```python
##defining a function to check if a number is odd or even

def odd_even(num):
  if num % 2 == 0:
    return 'Even'
  else:
    return 'Odd'
```

- Calling the function: function_name(pass in the parameters)

```python
print(odd_even(8))
print(odd_even(5))
```

```
Even
Odd
```

```python
## if no function us used:
num = 4
if num % 2 == 0:
    print('Even')
else:
    print('Odd')


num = 5
if num % 2 == 0:
    print('Even')
else:
    print('Odd')


### Tedious to check for many numbers
```

6215 648

```
x = 4
for i in range(1, x + 1):
    for j in range(1, i + 1):
        print(j, end = "")
```

What is the output of the above code?

```
x = 4
for i in range(1, x + 1):
    for j in range(1, i + 1):
        print(j, end = "")
```

What is the output of the above code?

This is the answer:

```
1121231234
```

```
x = 4
for i in range(1, x + 1):
    for j in range(1, i + 1):
        print(j, end = "")
    print()
```

```
1
12
123
1234
```

# OOPS

- Object Oriented Programming
- As there are only a few built in data types, we can create new objects (or new data types) as per our wish using OOP aspect of python
- Data type gives a set of properties to an instance of that data type, so similarly we can give new properties to our data type

# CLASSES

Classes contain blueprints that are used to make objects.


Features of an object are called Attributes/ Fields .

E.g:
A car will have engine, color, brand

Different functions/ operators of the class

Called like :
Myobj.function()

- Objects are the instance of a class.
- Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.

# SOME MORE INFO ON CLASSES:

## Class and instance variables:

- Class Variable: These variables are shared by all instances of the class, their value is assigned in the class
- Instance Variable: Unique to each instance, their value is assigned inside a constructor or a method with self

## __init__ method:

- Used to initializing the object’s state.
- It runs as soon as an object of a class is instantiated. The method is useful to do any initialization you want to do with your object.
- We must pass the parameters of the init method, else it raises an error

## self parameter:

- The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.
- It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any method in the class.

# Inheritance:

- It is a mechanism that allows you to create a hierarchy of classes that share a set of properties and methods by deriving a class from another class.

- **Syntax:**

```
Class BaseClass:     # parent class
  {Body}
Class DerivedClass(BaseClass):      #child class
  {Body}
```