->COMMENTS

Comments are used to write something that the programmer does not want to execute.

Comments are something that are ignored by the python interpreter.

TYPES OF COMMENTS

1. Single-line comments

We have to use '#' for writing a comment in Python.

2. Multiline comments

Multiline comments are not available in Python, but we can achieve it by using Doc

String """ Multiline """.

""" This is an amazing

```
example of a

Multiline comment!"""
```

Line Continuation

Use a backslash () to continue a statement to the next line

total=1+2+3+4+5+6+7+
4+5+6

print(total)

Variables in python

A variable is the name given to a memory location in a program. For example.

Rules to declare a variable

---- valid variable declaration -- For example

a=2

A=3

num1=5

Num=1

stuid=123

stu_id=321

_stu_name ="priya"

a,b,c=2,4,2

a=7;b=4;c=3;

----- Invalid variable declaration__

```
1num=54

stu name =234

num#=4       -- we can not use special characters.
```

--> You can not use numbers at variable start.

--> You can not use spaces in variables.

--> You should not use special characters in variables.

input/output function

Print()- outfuction input()-inputfunction

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

-> Data Types

> Data types are the things we store in Variables, and they define what data type variables are.

A. Basic Data Types

- Integers
- Floating-point numbers
- Strings
- Booleans

B. Advanced Data Types

- Lists
- Tuples
- Sets
- Dictionaries

> Python has built-in data types for different kinds of data.

Numbers

Integer - All the numbers excluding decimal places and fractions.

Float - All the decimal numbers and fraction values are Float.

Complex Numbers - with real and Imaginary parts are complex.

Strings

Strings - This is used to store anything in Python, literally anything that is available on your keyboard.

You have to use quotes to store anything, and it will be considered a string. You can use double Quotes ("") or single quotes (") to store both works the same

Boolean

Boolean - This is the data type that will always give the result of True & False

String Indexing

> You must have thought there are so many characters in a string but can you access everyone.

> Yes, that's possible using indexing. Indexing starts from 0 and goes till the number of characters you have.

-- Ex- a= "Hello" print(a[0]) ==> output - "H"

> There is negative indexing as well and it starts from -1, but the starting position is from the back of the string.

-- Ex- a ="Hello" print(a[-1]) ==> output - "o"

String Slicing

You know how to access characters in string. But there are slicing option as well.

Slicing means cutting out a slice from string and this is also done using index values.

eg a= "hello" print(a[1:4:1]) ==> output "ell"

So here we have start, stop and steps position and keep a note if we use stop at 4 it will slice till 3 only.

Type conversion

For understanding type conversion you have to look at these 4 things.

int()

float()

str()

bool()

There are more functions like this but these are 4 main function, looking at these functions you can guess these are used to convert one data type to another.

ex- a = 12

a = str(a) print(a) ==> "12" (a will be converted to string)

In [6]:

```
# Q. Write a program to read employee data: emp_id, emp_name, and emp_salary

emp_id =int(input("Enter your id :- "))
emp_name =str(input("Enter your name :- "))
emp_salary=float(input("Enter your salary :- "))
print(emp_id, emp_name,emp_salary)


print("Id Num:-",emp_id)
print("Emp name :-",emp_name)
print("Sal :-",emp_salary)

## . formate
print ("emp_id= {}\nemp_name= {},\nemp_salary= {}".format(emp_id, emp_name,emp_s

## f string method
print (f"emp_id= {emp_id},emp_name= {emp_name}, emp_salary= {emp_salary}")

print (f"emp_id= {emp_id},\nemp_name= {emp_name}, \nemp_salary= {emp_salary}")
```

```
22 22 22.0
Id Num:- 22
Emp name :- 22
Sal :- 22.0
emp_id= 22
emp_name= 22,
emp_salary= 22.0
emp_id= 22,emp_name= 22, emp_salary= 22.0
emp_id= 22,
emp_name= 22,
emp_salary= 22.0
```

In [ ]:

In [ ]: