Output

In [1]:
```
print("sanjan pandit")
```

```
sanjan pandit
```

In [11]:
```
a = 10
b = 20
print (a)
print (b)
```

```
10
20
```

print function has five arguments which are following

end argument

In [3]:
```
print(a,end = "")
print(b)
```

```
1020
```

In [4]:
```
print(a, end = "\t")
print(b)
```

```
10      20
```

\t -> is by default take 4 white space

In [5]:
```
print(a, end = "\b")
print(b)
```

```
120
```

This happens because \b moves the cursor back to one position, and "0" is overwritten by the space and "2".

sep argument

In [6]:
```
print("Hello", "world", sep = "-")
```

```
Hello-world
```

In [7]:
```
print(a,b , sep = "&")
```

```
10&20
```

In [12]:
```
print(a, sep = ' ', end = '\n', file = None, flush = False)
```

```
10
```

formating in print

There are many ways to print the same formating

```
In [15]: name = "Sanjan pandit"
         age = 23
         print("My name is {} and I am {} years old.".format(name, age))
```

```
My name is Sanjan pandit and I am 23 years old.
```

```
In [8]: print(f"The value of a is {a} and b is {b}")
```

```
The value of a is 10 and b is 20
```

```
In [9]: print("The value of a is ",a ," and b is ",b)
```

```
The value of a is  10  and b is  20
```

```
In [16]: name = "sanjan pandit"
         age = 23
         print("My name is %s and I am %d years old." % (name, age))
```

```
My name is sanjan pandit and I am 23 years old.
```

input function

There are two way to taking input

Input function has taken by default string

```
In [10]: input("Enter any number")
```

```
Out[10]: '8'
```

```
In [12]: input(prompt="Enter any number")
```

```
Out[12]: '8'
```

We can check input type using (int(), float(), eval()) function

Types of Errors In Python

Syntax Error

Runtime Error

Syntax Errors: Syntaxes are **Rules of language** an when we break these rules, the error which occurs is called **syntax Error**

Misspelled keywords

Incorrect use of operator

Omitting parentheses in a function call

```
In [13]: a+
  Cell In[13], line 1
    a+
      ^
SyntaxError: invalid syntax
```

Incorrect use of operator

In [14]:
```
print(a
```

```
  Cell In[14], line 1
    print(a
           ^
SyntaxError: incomplete input
```

Runtime Error

In [15]:
```
a = 10
b = 0
c = a/b
print(c)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[15], line 3
      1 a = 10
      2 b = 0
----> 3 c = a/b
      4 print(c)

ZeroDivisionError: division by zero
```

variable name / class name / function name Rule name

1. We can take a - z (samall latter)
2. We can take A - Z (Capital latter)
3. We can take Special character only one [under score]
4. Any name
5. variable should not be start with any digit

keyword

Keyword: keyword is nothing it is a word which is reserved by proggramming language

In [7]:
```
import keyword
```

We can check keyword this way

In [20]:
```
key = keyword.kwlist
print(key)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'clas
s', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from',
'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass',
'raise', 'return', 'try', 'while', 'with', 'yield']
```

```
In [25]: len(key)
```

```
Out[25]: 35
```

To check keywords using module **iskeyword**

```
In [23]: keyword.iskeyword('if')
```

```
Out[23]: True
```

keyword.softkwlist is a list that contains all the soft keywords in Python. Soft keywords are words that have a special meaning in certain contexts but are not reserved keywords. Unlike regular keywords, which cannot be used as identifiers (like variable names), soft keywords can be used as identifiers outside of their special contexts.

```
In [21]: keyword.softkwlist
```

```
Out[21]: ['_', 'case', 'match', 'type']
```

The keyword.issoftkeyword function takes a single string argument and returns True if the string is a soft keyword in Python, otherwise it returns False.

```
In [8]: keyword.issoftkeyword('case')
```

```
Out[8]: True
```