```python
name = input("Hello, enter you name \n")
print("Hi " + name)

# Formatting string
first = "John"
last = "Smith"
message = first + '[' + last + ....

msg = f'{first} [{last}] is a coder'
```

```python
string = "GeeksforGeeks"
string.swapcase() ---> "gEEKSFORgEEKS"
```

```python
# Dictionary Comprehension
my_dict = {i:i+7 for i in range(1, 10)}

# looping though dictionary
for key in likes:
...     print(key, "->", likes[key])
...
color -> blue
fruit -> apple
pet -> dog
```

```python
# How do you debug a Python program?
# By using this command we can debug a Python program:

$ python -m pdb python-script.py
```

```python
lang = input("What's the programming language you want to learn? ")

match lang:
    case "JavaScript":
        print("You can become a web developer.")

    case "Python":
        print("You can become a Data Scientist")
```

```python
my_list = [1, 2, 3, 4, 5]
removed_value = my_list.pop(-1)
print(my_list)  # Output: [1, 2, 3, 4]

my_list.append(6) # [1, 2, 3, 4, 5, 6]

my_list.insert(0, 1) # [1, 2, 3, 4, 5] or
```

```
my_list = [1] + my_list # [1, 2, 3, 4, 5]
```

```
list.sort(reverse=True|False, key=myFunc)                                    python
```

Expression vs. statements

All programs in JavaScript are made of statements. **Statements** just perform some actions but do not produce any value or output whereas **expressions** return some value. When interpreter sees an expression it retrieves its value and replaces expression with new value. Statements are used to manipulate those expressions.

Example: if **statement**
var x; if (y >= 0) { x = y; } else { x = -y; }

Equivalent code using **expression** with *conditional operator*
var x = (y >= 0 ? y : -y);

```
# Sorting a dictionary by values                                             python
footballers_goals = {'Eusebio': 120, 'Cruyff': 104, 'Pele': 150, 'Ronaldo': 132, 'Messi': 125}

sorted_footballers_by_goals = {i: j for i, j in sorted(footballers_goals.items(), key=lambda x:
x[1])}
print(sorted_footballers_by_goals)

# sorting by keys

sorted_footballers = sorted(footballers_goals.items())
```

Closures:

A Closure is a function object that remembers values in enclosing scopes even if they are not present in memory. Let us get to it step by step

Firstly, a **Nested Function** is a function defined inside another function. It's very important to note that the nested functions can access the variables of the enclosing scope.

```
def transmit_to_space(message):
    def data_transmitter():
        print(message)

    data_transmitter()
```

```
print(transmit_to_space("Test message"))

# Output:
# Test message
# None
```