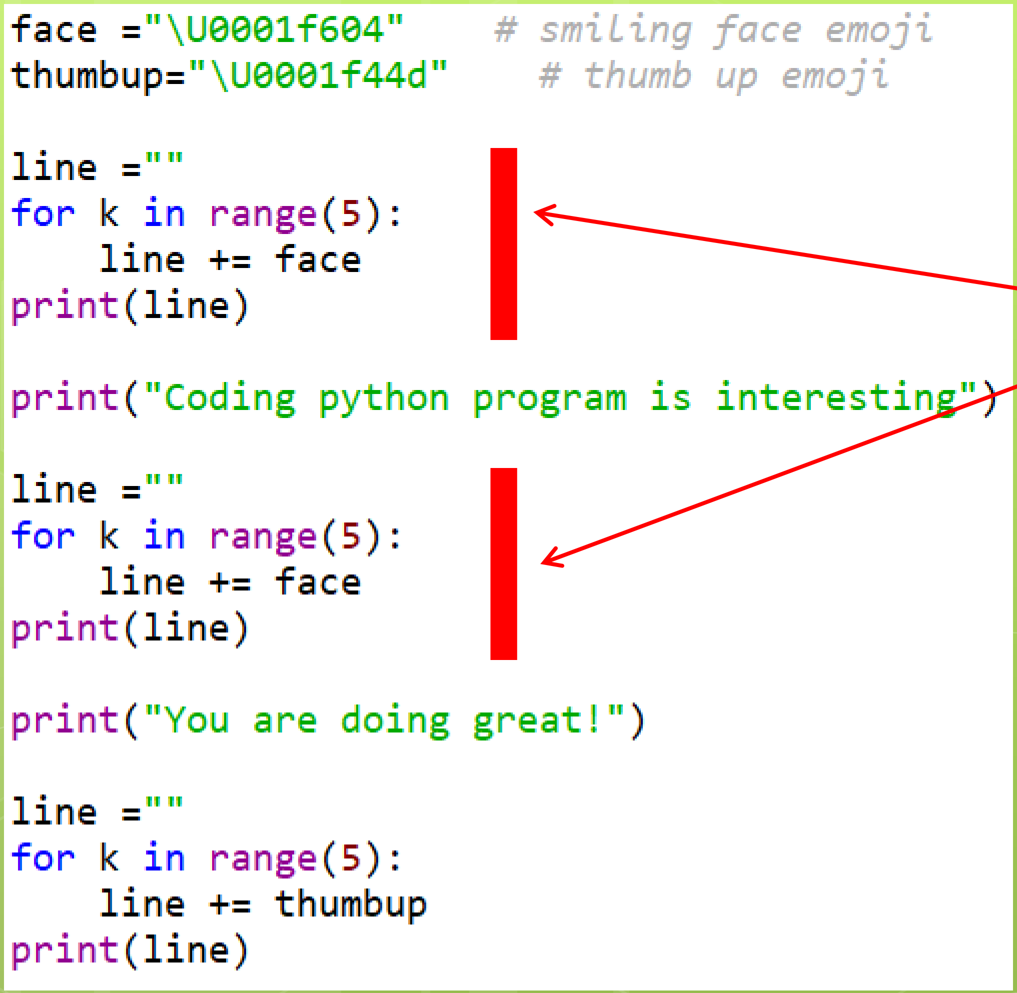
# Creating functions

## Function

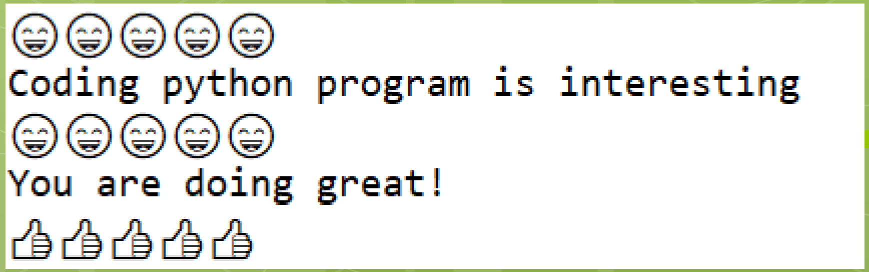
- For a block of codes that is used in multiple places of a program, we can define a function for that block
- Using function also makes the program more readable
- Function definition

```
def function-name ( ) :
        statements-to-be-executed
        statements-to-be-executed
```

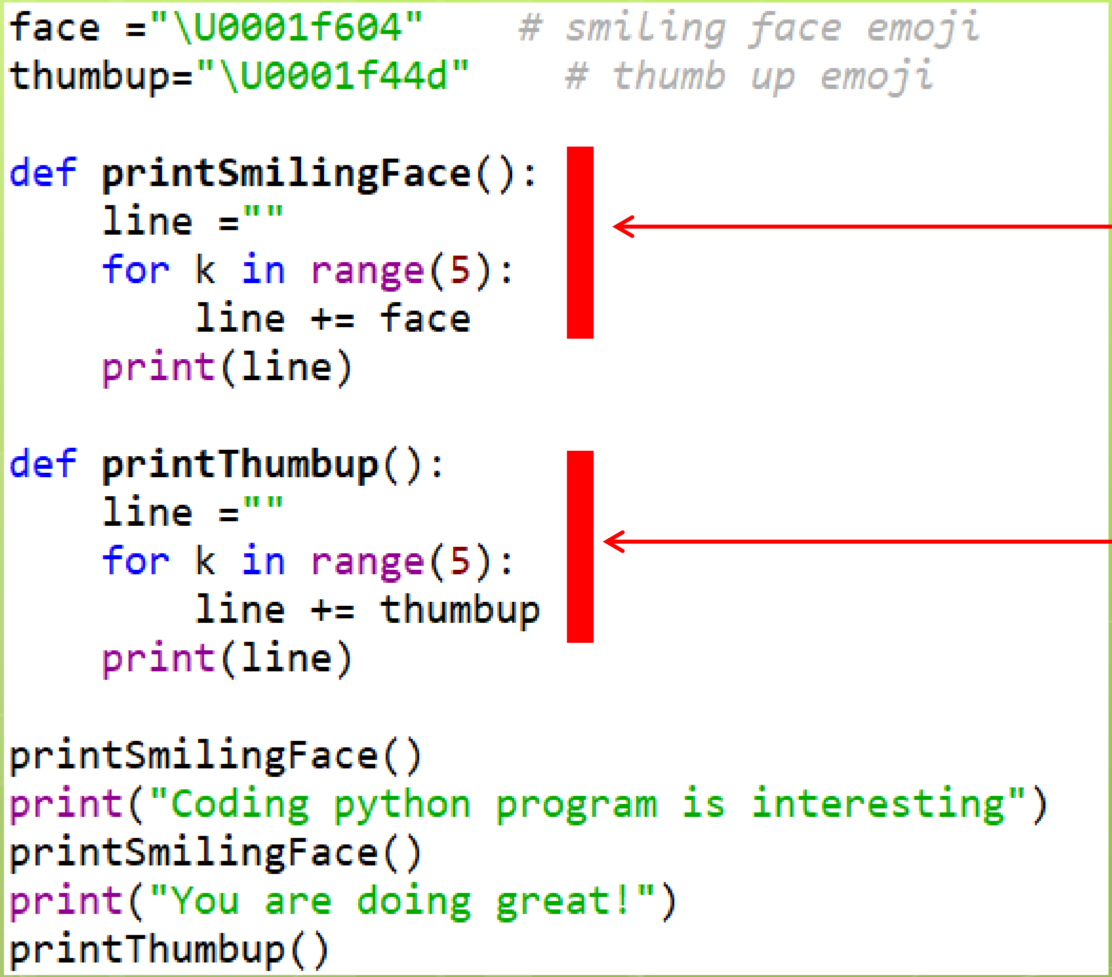


```
face ="\U0001f604"     # smiling face emoji
thumbup="\U0001f44d"     # thumb up emoji

line =""
for k in range(5):
    line += face
print(line)

print("Coding python program is interesting")

line =""
for k in range(5):
    line += face
print(line)

print("You are doing great!")

line =""
for k in range(5):
    line += thumbup
print(line)
```

The same code block is used in two places



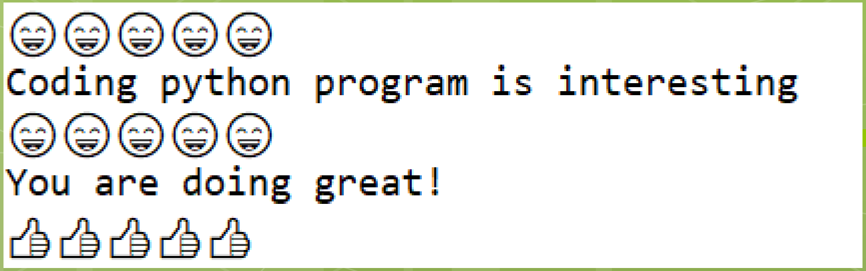
```
😄😄😄😄😄
Coding python program is interesting
😄😄😄😄😄
You are doing great!
👍👍👍👍👍
```



```python
face ="\U0001f604"     # smiling face emoji
thumbup="\U0001f44d"     # thumb up emoji

def printSmilingFace():
    line =""
    for k in range(5):
        line += face
    print(line)

def printThumbup():
    line =""
    for k in range(5):
        line += thumbup
    print(line)

printSmilingFace()
print("Coding python program is interesting")
printSmilingFace()
print("You are doing great!")
printThumbup()
```

**Creating a function called printSmilingFace()**

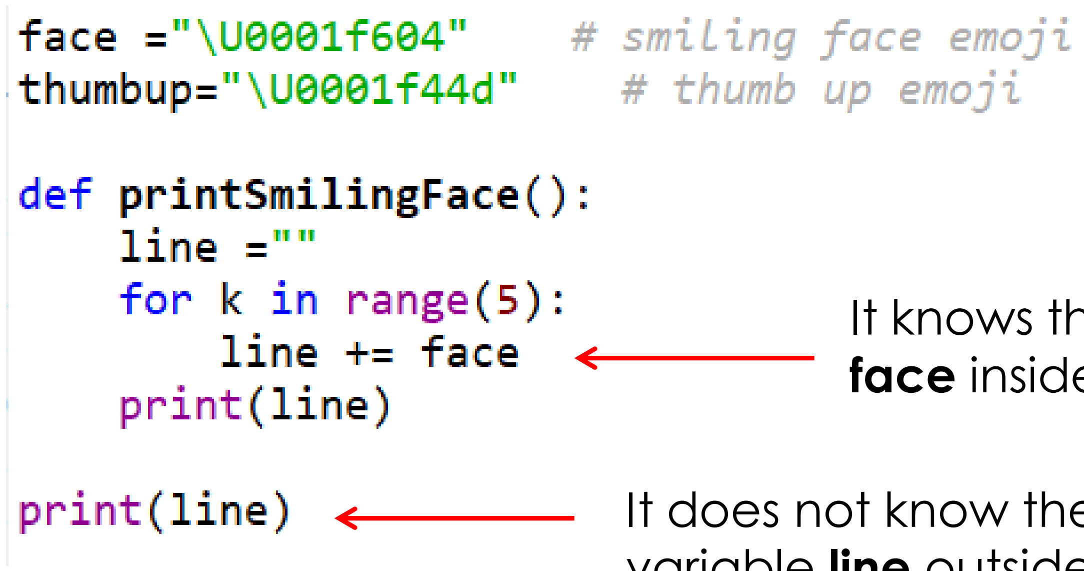
**Creating a function called printThumbup()**



```
😄😄😄😄😄
Coding python program is interesting
😄😄😄😄😄
You are doing great!
👍👍👍👍👍
```

## Scope of variable

- Variables declared outside functions can be seen by codes inside the functions
- Variables declared inside functions cannot be seen by codes outside the functions



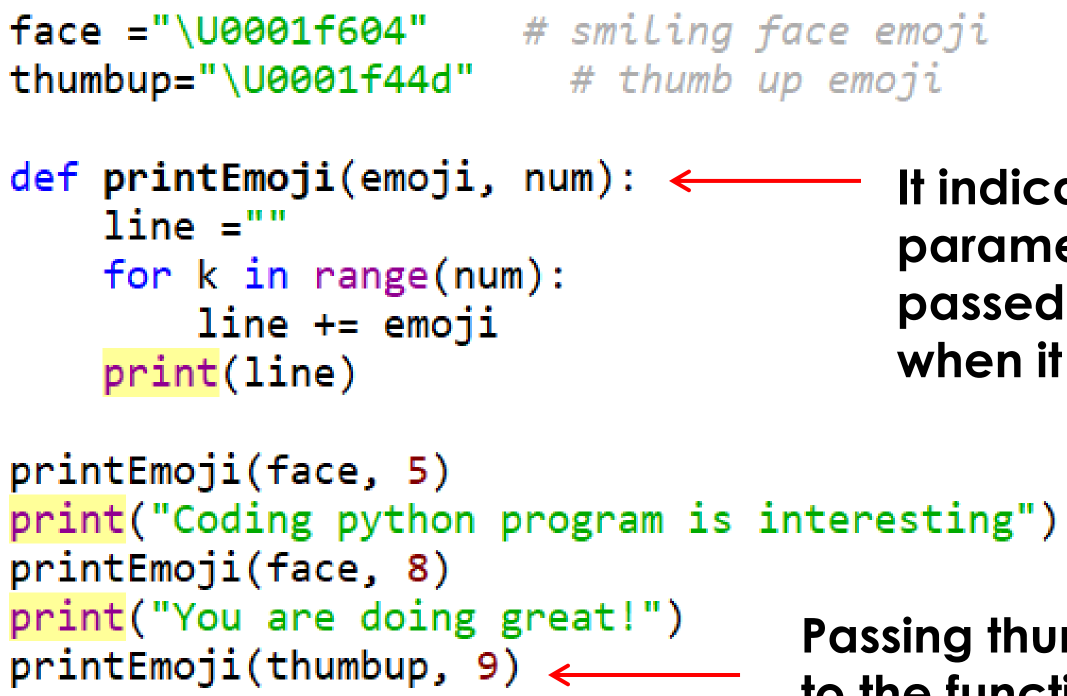
```
face ="\U0001f604"     # smiling face emoji
thumbup="\U0001f44d"     # thumb up emoji

def printSmilingFace():
    line =""
    for k in range(5):
        line += face
    print(line)

print(line)
```

It knows the value of variable **face** inside function

It does not know the value of variable **line** outside function

# Passing parameters to function

- We can pass parameters to a function to make it more versatile



```
face ="\U0001f604"     # smiling face emoji
thumbup="\U0001f44d"     # thumb up emoji

def printEmoji(emoji, num):
    line =""
    for k in range(num):
        line += emoji
    print(line)

printEmoji(face, 5)
print("Coding python program is interesting")
printEmoji(face, 8)
print("You are doing great!")
printEmoji(thumbup, 9)
```

**It indicates two parameters will be passed to the function when it is called**

**Passing thumbup and 9 to the function**

## Return values from function

- Function can also return its results

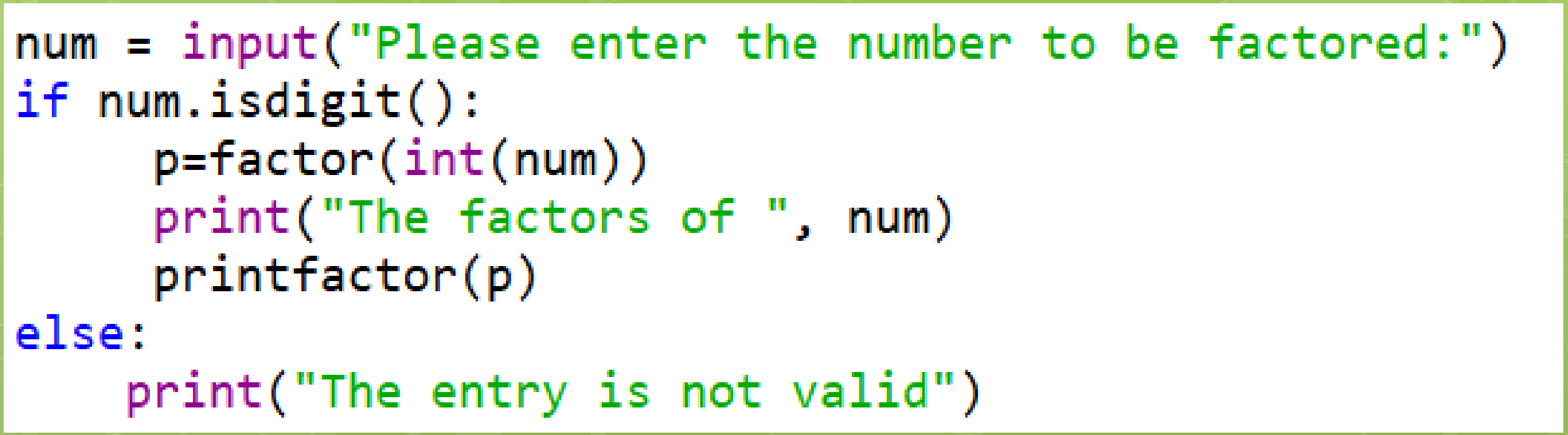
```
import random
face ="\U0001f604"    # smiling face emoji
thumbup="\U0001f44d"    # thumb up emoji

def printEmoji(emoji):
    line =""
    num = random.randint(1, 8)
    for k in range(num):
        line += emoji
    print(line)
    return num

a = printEmoji(face)
print("Coding python program is interesting")
b = printEmoji(face)
print("You are doing great!")
c = printEmoji(thumbup)
print("Number of Emojis printed is:", a+b+c)
```

**Return the value of num**

**Assign returned value to variable a**

## Today's Challenge:

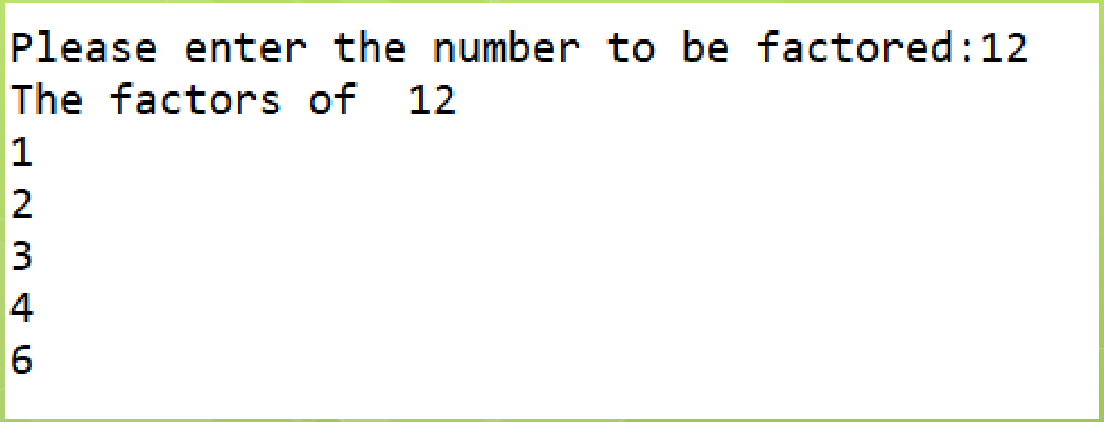
- Write two missing functions used in a program that asks user to type a positive integer, then the program print all the factors of the integer
- The two missing functions are:
  - **factor(num):** find all the factors and return them in a list
  - **printfactor(p):** print all the factors



```python
num = input("Please enter the number to be factored:")
if num.isdigit():
     p=factor(int(num))
     print("The factors of ", num)
     printfactor(p)
else:
    print("The entry is not valid")
```

## Today’s Challenge:

- Program output



```
Please enter the number to be factored:12
The factors of  12
1
2
3
4
6
```

- Hint: if y%x==0, x is a factor of y