# Objectives for class 8

--- Chapter 4 ---

4.9 To obtain a character in a string using the index operator [] (§4.3.11).

4.10 To obtain a substring in a string using the slicing operator [start : end] (§4.3.12).

4.11 Use repetition operator * to duplicate strings (§4.3.6).

4.12 To introduce objects and methods (§4.5).

4.13 To introduce the methods in the str class (§4.6).

4.14 To program using characters and strings (§4.7.1).

4.15 To invoke the print function with the end argument (§4.3.4).

--- Chapter 5 ---

5.1 To write programs for executing statements repeatedly using a while loop (§5.2).

# Use an index operator[] to access one character from a string

| str | P | r | o | g | r | a | m | m | i | n | g |  |  | 2 | 3 | ! |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |

**`str[index]`**

**`str[0]`** `stores 'P'`

**`str[1]`** `stores 'r'`

**`str[12]`** `stores ' ' (a space)`

**`str[17]`** `causes runtime error (out of index)`

# Use slice operator **[start:end]** to access one substring from a string

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| str | P | r | o | g | r | a | m | m | i | n | g | | | 2 | 3 | ! |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

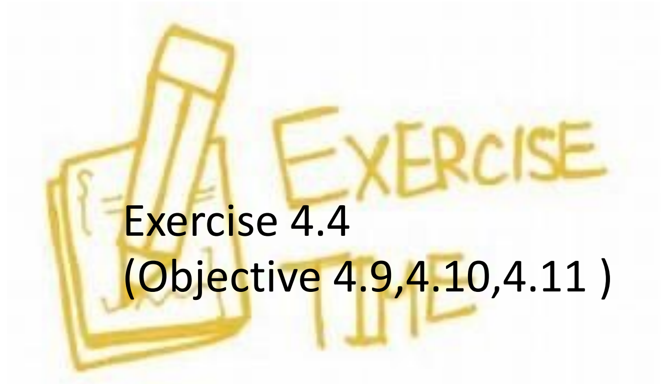
**`str[start:end]`**

```
str[0:1] stores ‘P’            ([0:1] picks 0)
str[14:] stores ’3!’         ([14:] picks 14,15)
str[:2] stores ’Pr’            ([:2] picks 0,1)
str[0:5:2] stores ‘Por’   ([0:5:2] picks 0,2,4)
```

## Use repetition operator * to duplicate strings

```
>>> s1 = 'Welcome’

>>> s2 = 3 * s1
>>> s2
WelcomeWelcomeWelcome

>>> s3 = s1 * 3
>>> s3
WelcomeWelcomeWelcome
```

## Practice



Exercise 4.4
(Objective 4.9,4.10,4.11 )

Suppose that s1 and s2 are two strings, given as follows:

```
s1 = "Welcome to Python"
s2 = "to"
```

What are the results of the following expressions?

```
a. s1[4]
b. s1[-4]
c. s2[1:]
d. s2*4
```

## All Data are Objects in Python

- An object is an entity.
- Each object has an **id** and a **type**.
- Objects of the same kind have the same **type**.
- Use the **id** function and **type** function to get object information

```
>>> n = 3  # n is an integer
>>> id(n)
505408904
>>> type(n)
<class ’int’>
>>> f = 4.0  # f is a float
>>> id(f)
26647120
>>> type(f)
<class ’float’>
```

```
>>> s = "Welcome" # s is a string
>>> id(s)
36201472
>>> type(s)
<class ’str’>
```

**id** and **type** rarely used in programming, but good pedagogical tools for understanding objects.

## Methods do Operations on a Specific Object

- Methods can only be invoked from  a specific object.

```
str=’Welcome’

str.lower() # Returns a new string in lowercase
```

## str Object Methods: Converting Strings

| Method | Description |
|---|---|
| capitalize() | Returns a copy of this string with only the first character capitalized. |
| lower() | Returns a copy of this string with all letters converted to lowercase. |
| upper() | Returns a copy of this string with all letters converted to uppercase. |
| title() | Returns a copy of this string with the first letter capitalized in each word. |
| swapcase() | Returns a copy of this string in which lowercase letters are converted to uppe and uppercase to lowercase. |
| replace(old, new) | Returns a new string that replaces all the occurrence of the old string with a n string. |
| replace(old, new, n) | Returns a new string that replaces up to n number of the occurrence of the old with a new string. |

```
>>> s = "Welcome"
>>> s1 = s.lower() # Invoke the lower method
>>> s1
'welcome'
>>> s2 = s.upper() # Invoke the upper method
>>> s2
'WELCOME’
>>> s2 = s.replace(‘e’,’E’) # Invoke the replace method
>>> s2
‘WElcomE'
```

## str Object Methods: Striping Whitespace Characters

| **Method** | **Description** |
|---|---|
| lstrip() | Returns a string with the leading whitespace characters removed. |
| rstrip() | Returns a string with the trailing whitespace characters removed. |
| strip() | Returns a string with the starting and trailing whitespace characters removed. |

```
>>> s = "\t Welcome \n"
>>> s1 = s.strip() # Invoke the strip method
>>> s1
'Welcome’
```

## str Object Methods: Testing Strings

| **Method** | **Description** |
|---|---|
| isalnum() | Returns True if all characters in this string are alphanumeric and there is at least one character. |
| isalpha() | Returns True if all characters in this string are alphabetic and there is at least one character. |
| isdigit() | Returns True if this string contains only number characters. |
| isidentifier() | Returns True if this string is a Python identifier. |
| islower() | Returns True if all characters in this string are lowercase letters and there is at least one character. |
| isupper() | Returns True if all characters in this string are uppercase letters and there is at least one character. |
| isspace() | Returns True if this string contains only whitespace characters. |

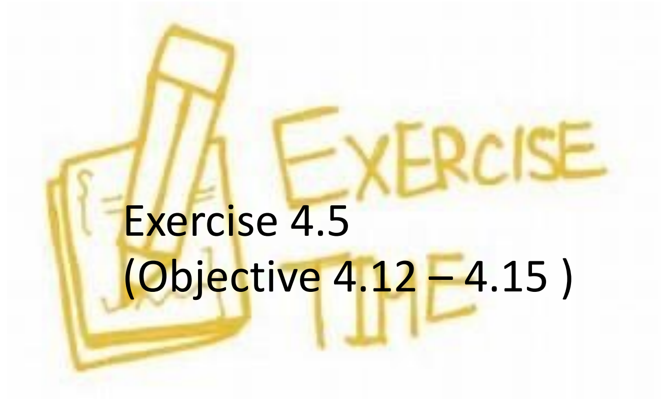
```
>>> s = "\t Welcome \n"
>>> s.isalpha()
False
>>> s.isdigit()
False
>>> s.strip().isalpha()
True
>>> s.strip().islower()
False
```

## str Object Methods: Searching for Substrings

| Method | Description |
|---|---|
| endswith(s1) | Returns True if the string ends with the substring s1. |
| startswith(s1) | Returns True if the string starts with the substring s1. |
| find(s1) | Returns the lowest index where s1 starts in this string, or -1 if s1 is not found in this string. |
| rfind(s1) | Returns the highest index where s1 starts in this string, or -1 if s1 is not found in this string. |
| count(subtring) | Returns the number of non-overlapping occurrences of this substring. |

```
>>> s = "\t Welcome \n"
>>> s.count('com')
1
>>> 'abab'.count('ab’)
2
>>> s.startswith('wel')
False
>>> s.startswith('Wel')
False
>>> s.find('Wel')
2
>>> s.strip().find('Wel')
0
```

# Practice



Exercise 4.5
(Objective 4.12 – 4.15 )

# A print game with “Programming is fun”

**ONE time?**

```
print("Programming is fun!");
```

**TWO times?**

```
print("Programming is fun!");
print("Programming is fun!");
```

**THREE times?**

```
print("Programming is fun!");
print("Programming is fun!");
print("Programming is fun!");
```

**ONE HUNDRED times?**

100 times

```
print("Programming is fun!");
print("Programming is fun!");
print("Programming is fun!");
print("Programming is fun!");
print("Programming is fun!");
print("Programming is fun!");

…

…

…
print("Programming is fun!");
print("Programming is fun!");
print("Programming is fun!");
```

**Repeatedly typing is tedious!!!**

## Ask a question to control the repetition of execution of a code block

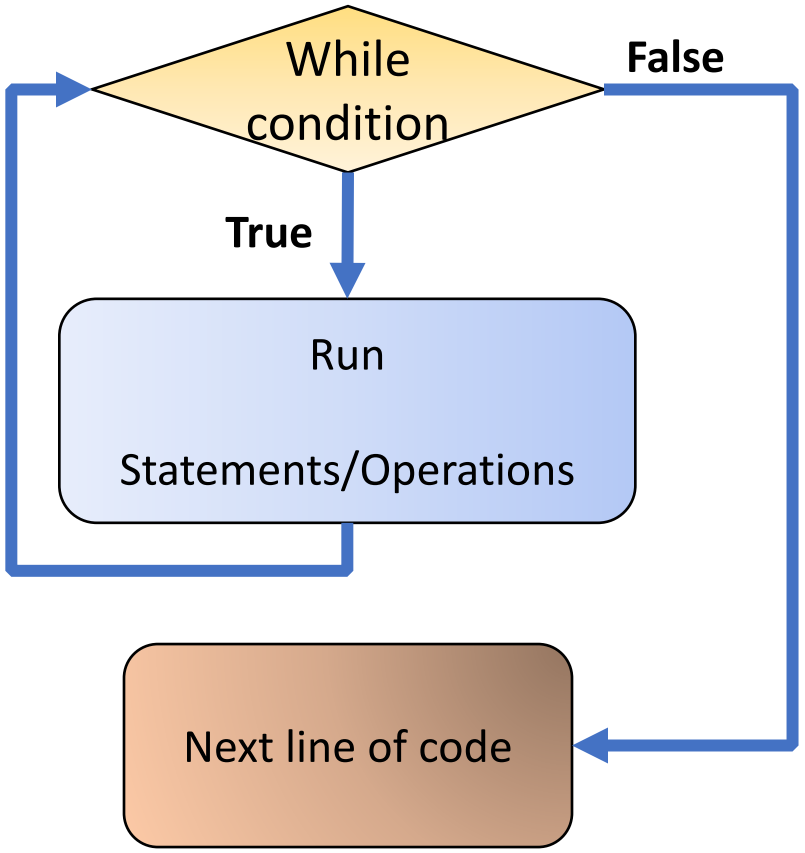
Print “Programming is fun” 100 times

```
Step 1: count=0 # record how many times we have printed the message

while count<100, do step 2 and step 3
    Step 2: Print “Programming is fun”
    Step 3: Increase count by 1
```

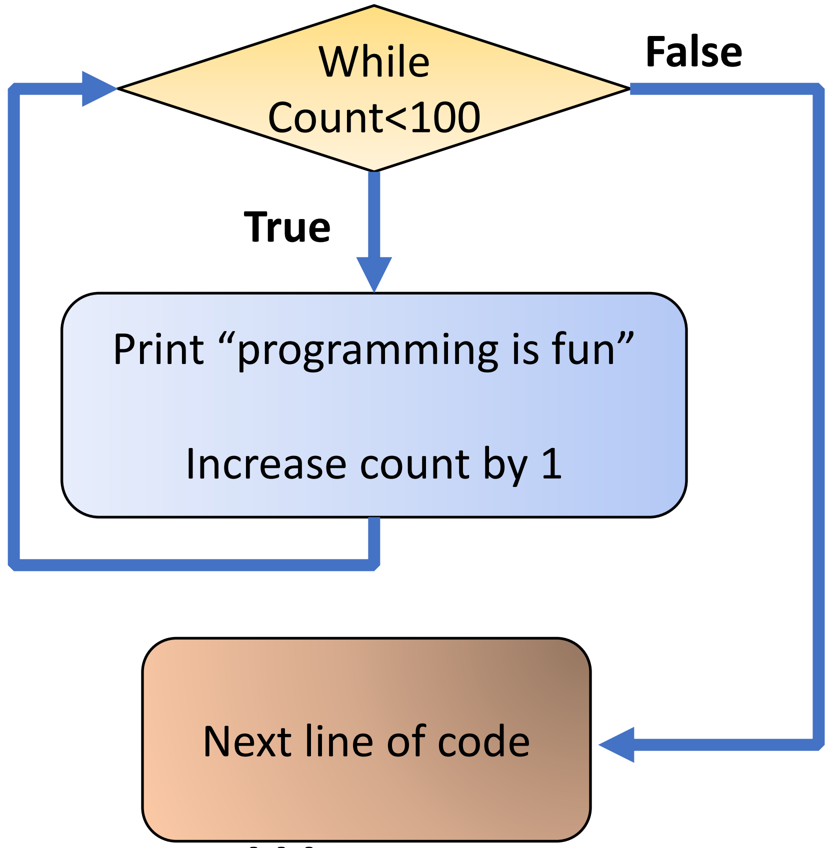
# While loop – repeat executing a code block

```
while <condition> :
    <statement>
    <statement>
    <statement>
<Next line of code>
```

**Loop Body**



```
1 count=0

2 while count<100:
3     print("Programming is fun")
4     count=count+1
```



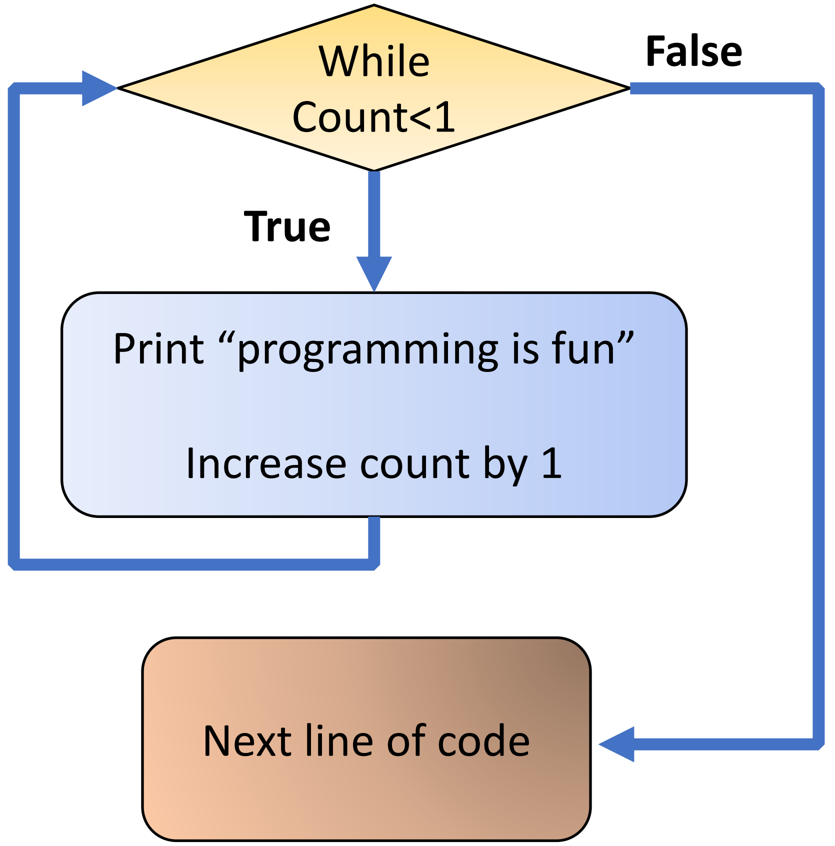
"Programming is fun" is printed ONE HUNDRED times!!!

count=1

Execution order of lines : 1 2 3 4 2 3 4 2 3 4 …. 2

count=0 count=3 count=100

```
1 count=0

2 while count<1:
3     print("Programming is fun")
4     count=count+1
```

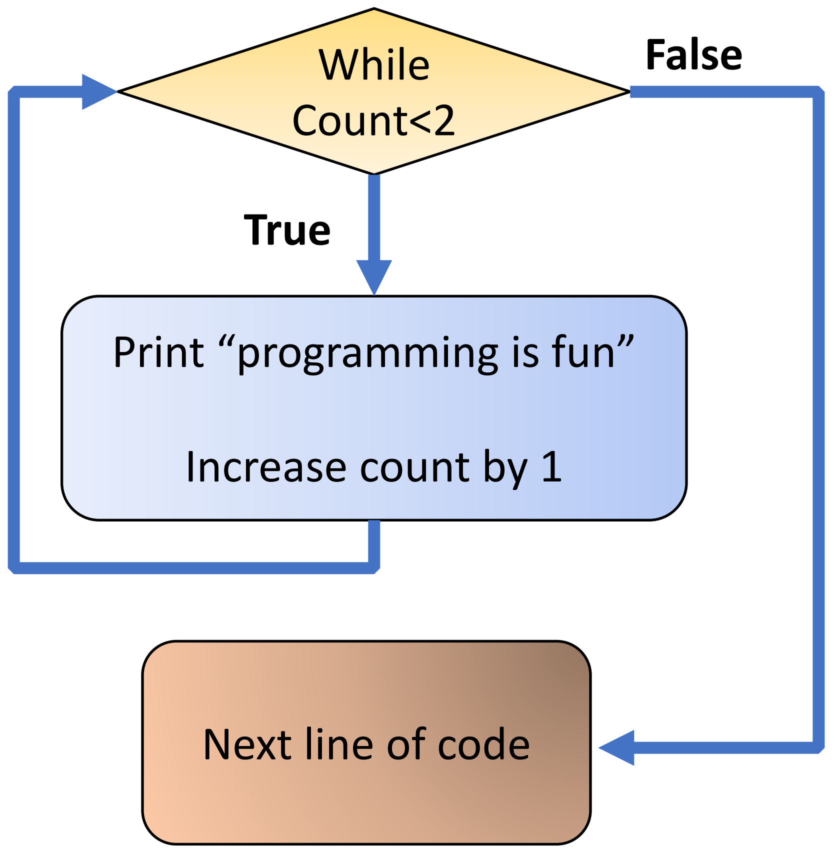


“Programming is fun” is printed ONE time!!!

Execution order of lines : 1 2 3 4 2

count=0

```
1 count=0

2 while count<2:
3     print("Programming is fun")
4     count=count+1
```



“Programming is fun” is printed TWO times!!!

count=1

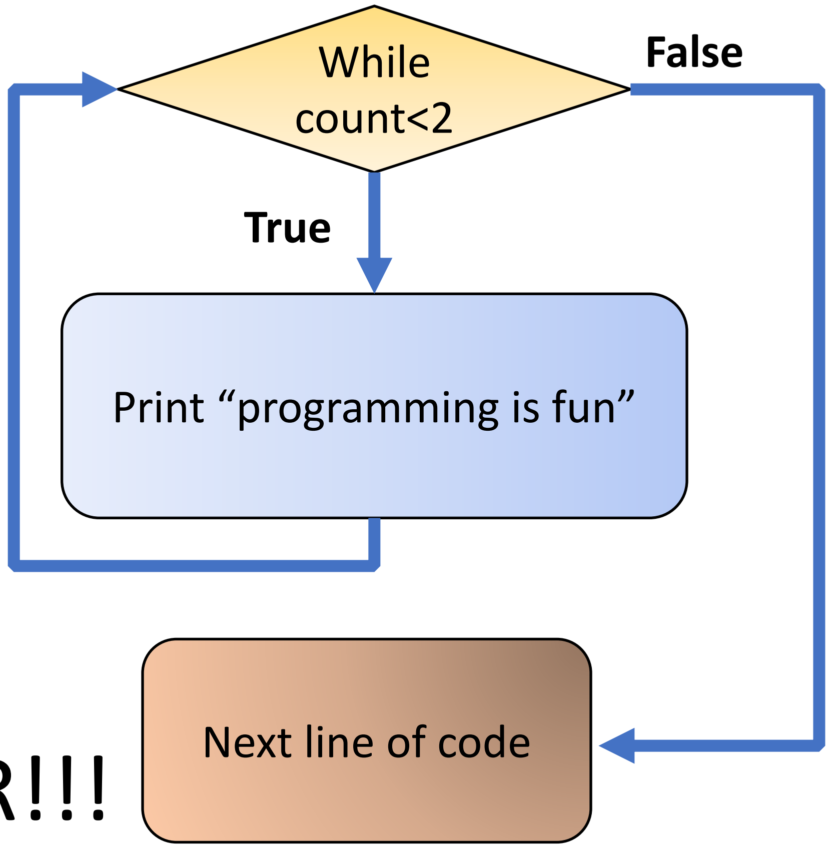
Execution order of lines : 1 2 3 4 2 3 4 2

count=0

count=2

# Infinite Loop happens if loop never stops

```
1 count=0

2 while count<2:

3     print("Programming is fun")
```



"Programming is fun" is printed FOREVER!!!

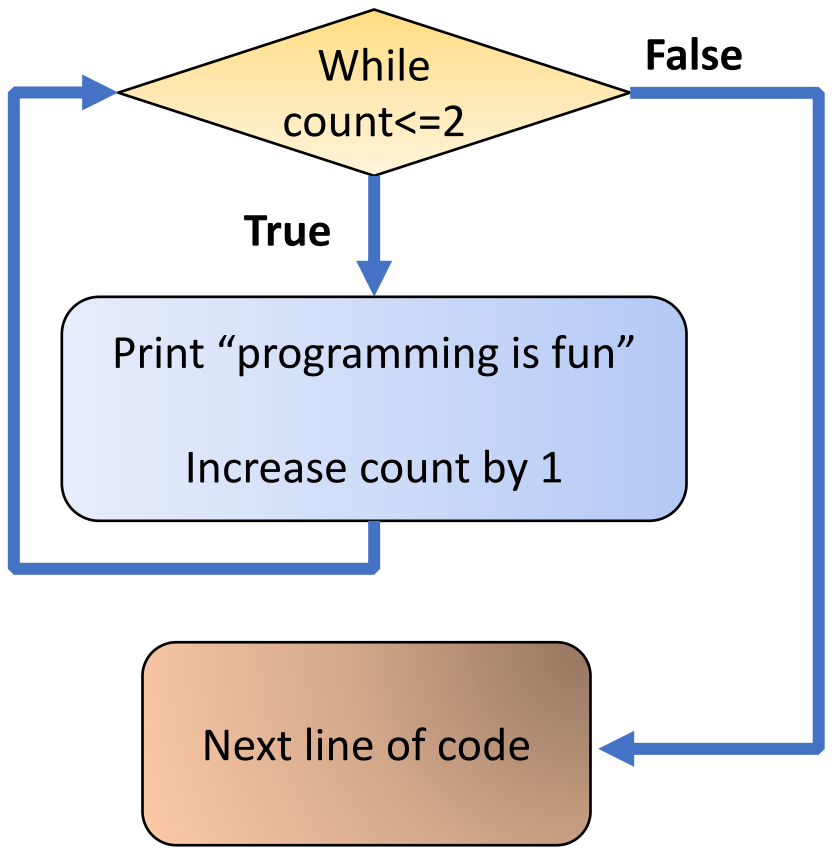
count=0

Execution order of lines : 1 2 3 4 2 3 4 2 ......................

count=0 count=0

# Practice 1

```
1 count=0

2 while count<=2:
3     print("Programming is fun")
4     count=count+1
```

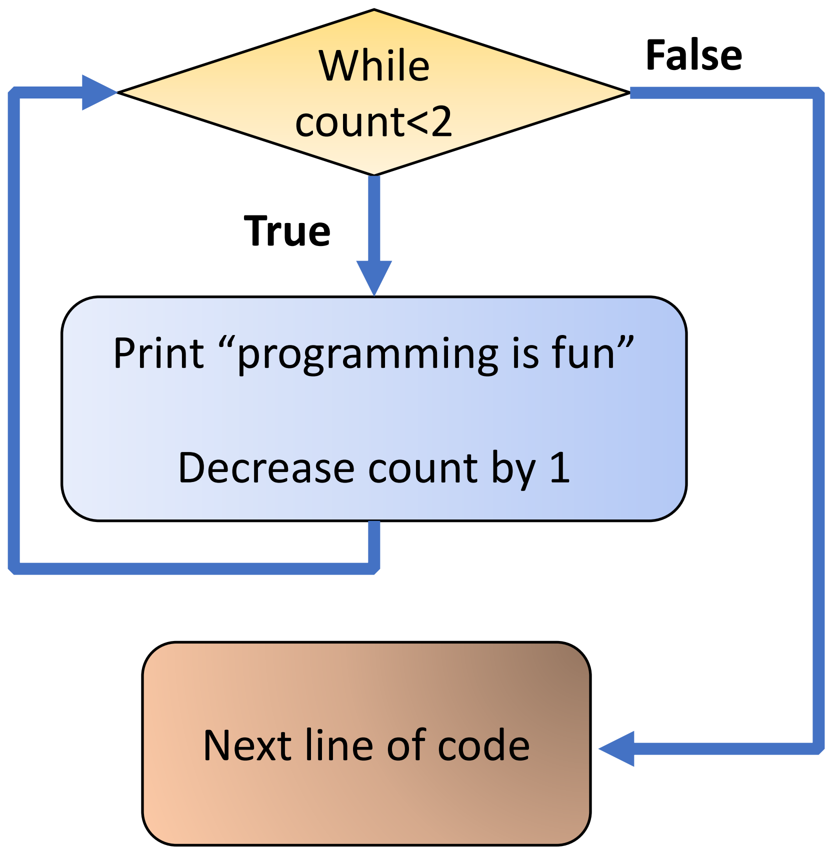


What is the output of program?

Write down the execution order of lines.

## Practice 2

```
1 count=0

2 while count<2:
3     print("Programming is fun")
4     count=count-1
```

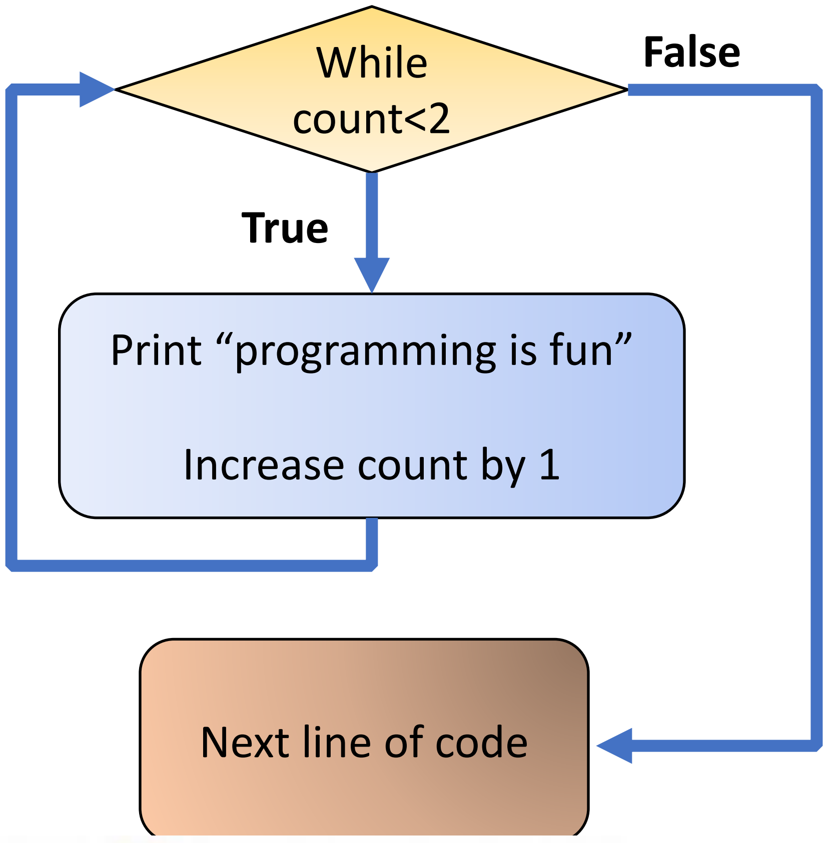


Write down the execution order of lines.

What is the output of program?

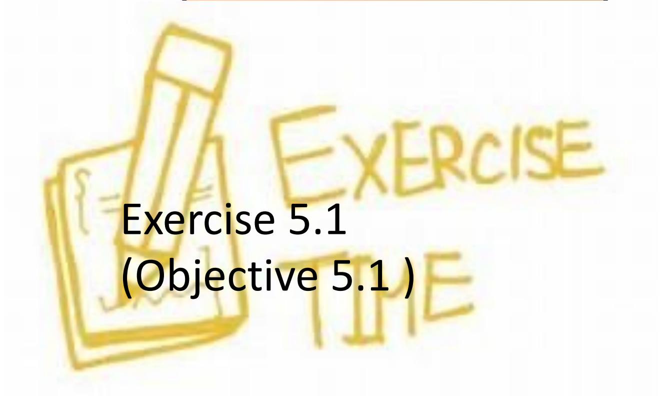
# Practice 3

```
1 count=1

2 while count<2:

3     print("Programming is fun")

4     count=count+1
```



Write down the execution order of lines

What is the output of program?



Exercise 5.1
(Objective 5.1 )