GKTCS INNOVATIONS

Control Structure

Conditional Statements (Decision Making)

GKTCS INNOVATIONS

- Python language provide the following conditional (Decision making) statements.

If statements

Else statements

Elif statements

Nested if statements

If Statement

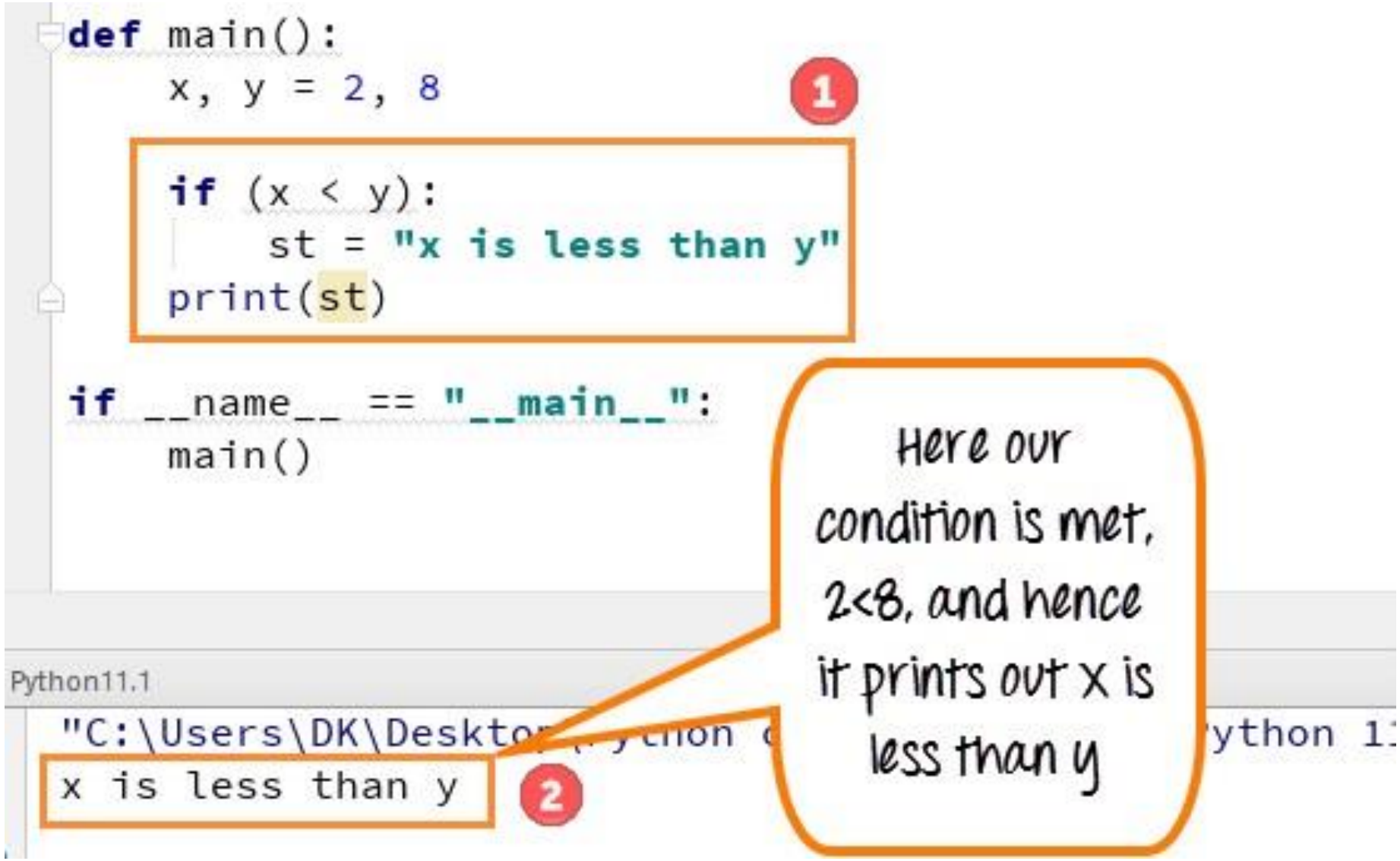
GKTCS INNOVATIONS

- In Python, If Statement is used for decision making.
- It will run the body of code only when IF statement is true.
- When you want to justify one condition while the other condition is not true, then you use "if statement".

Syntax:

```
if expression
    Statement
else
    Statement
```

Let see an **example-**



What happen when "if condition" does not meet

GKTCS INNOVATIONS

- In this step, we will see what happens when your "if condition" does not meet.

Python11.1.py

```
1   #
2   # Example file for working with conditional statement
3   #
4   def main():
5       x, y = 8, 4
6
7       if (x < y):
8           st = "x is less than y"
9       print(st)
10
11  if __name__ == "__main__":
12      main()
13
14
```

It shows the error because it does not match our "if condition" (i.e x<y)

Run Python11.1

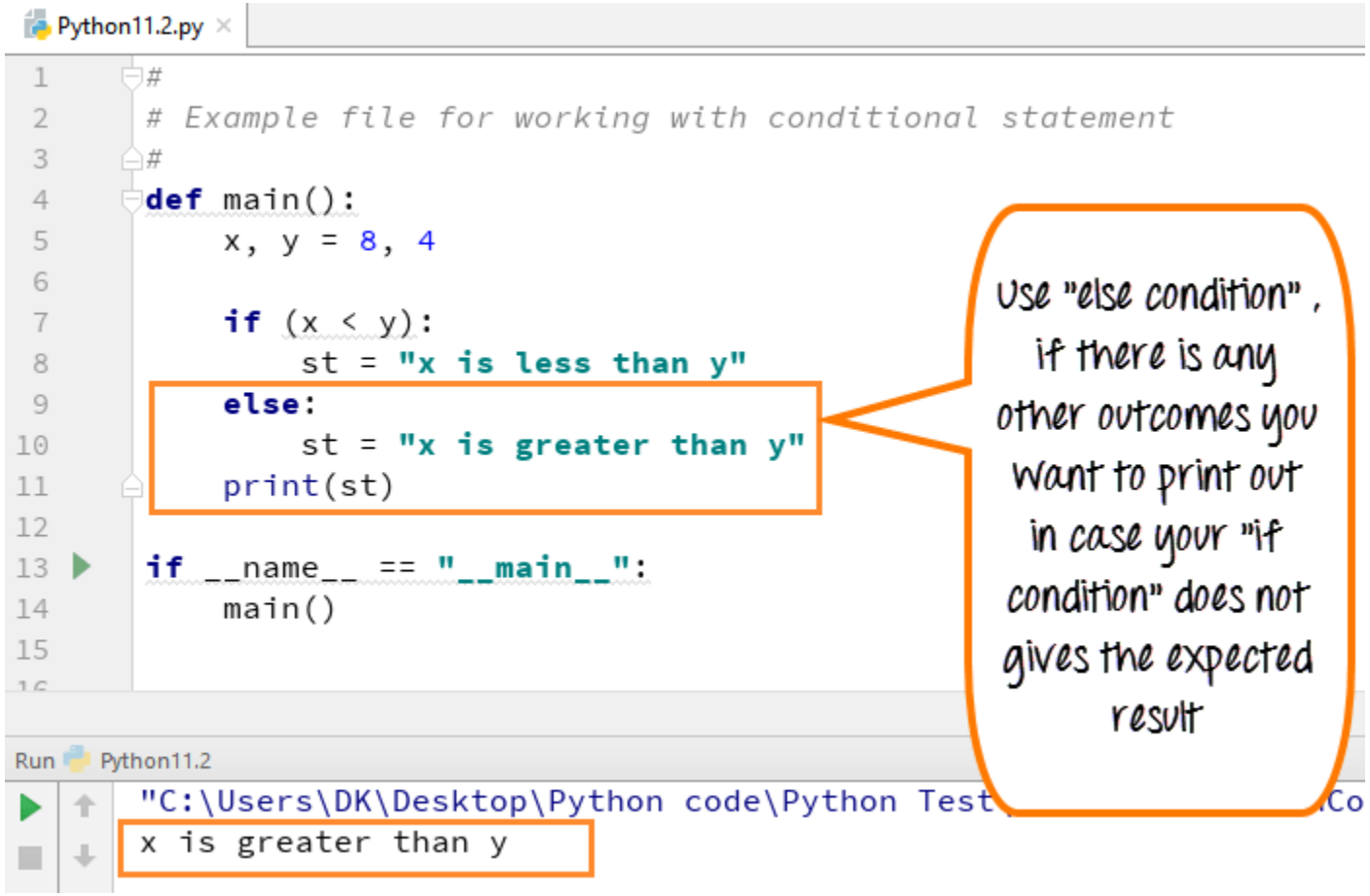
```
"C:\Users\DK\Desktop\Python code\Python Test\Python 11\PythonCode11\venv\S
Traceback (most recent call last):
  File "C:/Python Code/PythonCode11/Python11.1.py", line 12, in <module>
    main()
  File "C:/Python Code/PythonCode11/Python11.1.py", line 9, in main
    print(st)
UnboundLocalError: local variable 'st' referenced before assignment
```

- Code Line 5: We define two variables x, y = 8, 4
- Code Line 7: The if Statement checks for condition x<y which is **False** in this case
- Code Line 8: The variable st is **NOT** set to "x is less than y."
- Code Line 9: The line print st - is trying to print the value of a variable that was never declared. Hence, we get an error.

How to use "else statement"

GKTCS INNOVATIONS

- The "else statement" is usually used when you have to judge one statement on the basis of other.
- If one condition goes wrong, then there should be another condition that should justify the statement or logic.



- Code Line 5: We define two variables x, y = 8, 4
- Code Line 7: The if Statement checks for condition x<y which is **False** in this case
- Code Line 9: The flow of program control goes to else condition
- Code Line 10: The variable st is set to "x is **greater** than y."
- Code Line 11: The line print st will output the value of variable st which is "x is greater than y"

When "else condition" does not work

GKTCS INNOVATIONS

- There might be many instances when your "else condition" won't give you the desired result.
- It will print out the wrong result as there is a mistake in program logic.
- In most cases, this happens when you have to justify more than two statement or condition in a program.

Python11.3.py

```
1   #
2   # Example file for working with conditional statement
3   #
4   def main():
5       x, y = 8, 8
6
7       if (x < y):
8           st = "x is less than y"
9       else:
10          st = "x is greater than y"
11      print(st)
12
13
14  if __name__ == "__main__":
15      main()
```

Oops ! .... Now both #numbers over here are same, still it prints out "x is greater than y"

Run Python11.2

```
"C:\Users\DK\Desktop\Python  \Python Test\Python 11\Pyt
x is greater than y
```

How to use "elif" statement

GKTCS INNOVATIONS

- To correct the previous error made by "else condition", we can use **"elif"** statement.
- By using "**elif**" condition, you are telling the program to print out the third condition or possibility when the other condition goes wrong or incorrect.

Python11.3.py

```
1   #
2   # Example file for working with conditional statement
3   #
4   def main():
5       x, y = 8, 8
6
7       if (x < y):
8           st = "x is less than y"
9
10      elif (x == y):
11          st = "x is same as y"
12
13      else:
14          st = "x is greater than y"
15      print(st)
16
17
18  if __name__ == "__main__":
19      main()
20
```

Run Python11.3

```
"C:\Users\DK\Desktop ython
x is same as y
```

When both co-ordinates(8,8) are same we have used "elif condition" to print out, "x is same as y"

- Code Line 5: We define two variables x, y = 8, 8
- Code Line 7: The if Statement checks for condition x<y which is **False** in this case
- Code Line 10: The flow of program control goes to the elseif condition. It checks whether x==y which is true
- Code Line 11: The variable st is set to "x is **same as** y."
- Code Line 15: The **flow of program control exits the if Statement (it will not get to the else Statement).**
- And print the variable st. The output is "x is same as y" which is correct

How to execute conditional statement with minimal code

GKTCS INNOVATIONS

Syntax:

A If B else C

Example:

```
1   def main():
2       x, y = 10, 8
3       st = "x is less than y" if (x < y) else "x is greater than or equal to y"
4       print(st)
5
6   if __name__ == "__main__":
7       main()
8
9
10
11
```

Python gives freedom to use minimal code to execute your conditional statement

```
Run  Python11.4
"C:\Users\DK\Desktop\Python code\Python Test\Python 11\PythonCode11\venv\Scripts\py
x is greater than or equal to y
```

- Code Line 2: We define two variables x, y = 10, 8
- Code Line 3: Variable st is set to "x is less than y "if x<y or else it is set to "x is greater than or equal to y".
- In this x>y variable st is set to **"x is greater than or equal to y."**
- Code Line 4: Prints the value of st and gives the correct output

Nested If Statement

GKTCS INNOVATIONS

When there is an if statement (or if..else or if..elif..else) is present inside another if statement (or if..else or if..elif..else) then this is calling the **nesting of control statements**.

Here we have a if statement inside another if..else statement block. Nesting control statements makes us to check multiple conditions.

```
num = -99
if num > 0
        print(“positive number”)
else:
        print(“negative number”)
if -99<=num:
        print(“two digit negative number”)
```

Output:

```
negative number
two digit negative number
```