ABDELGHAFOR'S VIRTUAL INTERNSHIP

# PYTHON PROGRAM

SESSION (2)

PREPARED BY : MARK KOSTANTINE

# AGENDA OVERVIEW

03

CONDITIONS

10

WHILE LOOPS

14

FOR LOOPS

19

TASKS

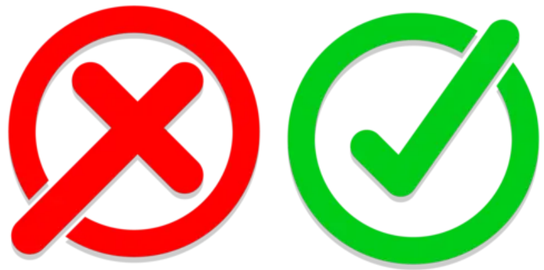
22

QUESTIONS

# CONDITIONS

Python supports the usual logical conditions from mathematics:

- **Equals:** a == b
- **Not Equals:** a != b
- **Less than:** a < b
- **Less than or equal to:** a <= b
- **Greater than:** a > b
- **Greater than or equal to:** a >= b



# IF STATEMENT

These conditions can be used in several ways, most commonly in **"if statements"** and loops
An **"if statement"** is written by using the if keyword

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

# SHORT HAND IF

If you have only one statement to execute, you can put it on the same line as the if statement

```
if a > b: print("a is greater than b")
```

# ELIF STATEMENT

The **elif** keyword is Python's way of saying "if the previous conditions were not true, then try this condition"

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

# ELSE STATEMENT

The **else** keyword catches anything which isn't caught by the preceding conditions

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

# IF WITH LOGICAL OPERATORS

Python supports the usual logical conditions from mathematics:

## 01

### AND

The **and** keyword is a logical operator, and is used to combine conditional statements

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

## 02

### OR

The **or** keyword is a logical operator, and is used to combine conditional statements

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

## 03

### NOT

The **not** keyword is a logical operator, and is used to reverse the result of the conditional statement

```
a = 33
b = 200
if not a > b:
  print("a is NOT greater than b")
```

# NESTED IF

You can have if statements inside if statements, this is called nested if statements

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

# CONDITIONS EXERCISE

Print **"Hello World"** if a is greater than b

# WHILE LOOPS

With the while loop we can execute a set of statements as long as a condition is true

```
i = 1
while i < 6:
  print(i)
  i += 1
```

**Note:** remember to increment i, or else the loop will continue forever.

# BREAK / CONTINUE STATEMNETS

- With the **break** statement we can stop the loop even if the while condition is true

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

- With the **continue** statement we can stop the current iteration, and continue with the next

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

# ELSE STATEMENT

With the else statement we can run a block of code once when the condition no longer is true

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

# WHILE LOOP EXERCISE

Print **i** as long as **i** is less than 10

# FOR LOOPS

- A **for** loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string)
- This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.
- With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

**Note:** The for loop does not require an indexing variable to set beforehand

- Even **strings** are iterable objects, they contain a sequence of characters

```
for x in "banana":
  print(x)
```

# FOR LOOPS

- With the **break** statement we can stop the loop before it has looped through all the items :

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

# RANGE FUNCTION

- To loop through a set of code a specified number of times, we can use the **range()** function,
- The **range()** function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

```
for x in range(6):
  print(x)
```

```
for x in range(2, 6):
  print(x)
```

```
for x in range(2, 30, 3):
  print(x)
```

# ELSE IN FOR LOOP

- The **else** keyword in a for loop specifies a block of code to be executed when the loop is finished

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

**Note:** The else block will **NOT** be executed if the loop is stopped by a break statement.

```
for x in range(6):
  if x == 3: break
  print(x)
else:
  print("Finally finished!")
```

# NESTED LOOPS

- A nested loop is a loop inside a loop.
- The " inner loop " will be executed one time for each iteration of the " outer loop " :

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

# FOR LOOP EXERCISE

Loop through the items in the fruits list

```
fruits = ["apple", "banana", "cherry"]
    x   fruits
  print(x)
```

# TASKS

**Beginner Level :**

- Conditions: Given a variable age = 15, write a Python program that prints whether the person is a child, teenager, or adult.

- While Loop: Create a program that starts with a variable number = 10. Use a while loop to subtract 1 from number until it becomes 0, printing each value of number.

- For Loop: Write a program that prints the multiplication table (1 to 10) for the number 5.

# TASKS

**Intermediate Level :**

- Conditions: Given three variables a = 10, b = 25, and c = 15, write a Python program that prints the largest number among them without using the max() function.

- While Loop: Start with a variable attempts = 0. Use a while loop that continues to increment attempts until it reaches 3, and print "Attempt number X" each time, where X is the current attempt number.

- For Loop: Create a program that calculates and prints the factorial of 6 using a for loop.

# TASKS

## Advanced Level :

- Conditions: Given a variable year = 2024, write a program that determines if it is a leap year. The program should check if the year is divisible by 4 but not divisible by 100 unless it is also divisible by 400.

- While Loop: Start with two variables a = 0 and b = 1. Use a while loop to generate and print the first 10 numbers of the Fibonacci sequence.

- For Loop: Write a Python program that takes the list numbers = [2, 3, 4, 5, 6, 7, 8, 9, 10] and outputs a new list with only the prime numbers from the original list using nested for loops.

# ANY QUESTIONS ?

PYTHON PROGRAM

# THANK YOU

UPCOMING NEXT WEEK : SESSION (3)