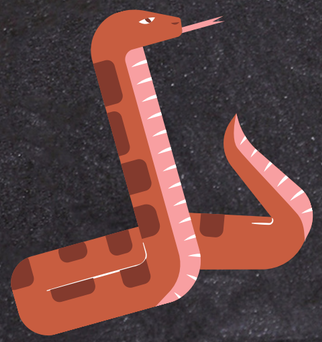
M. Amin HosseinNiya | Ali Rashedi

# Python Programming

## A brief of the content!



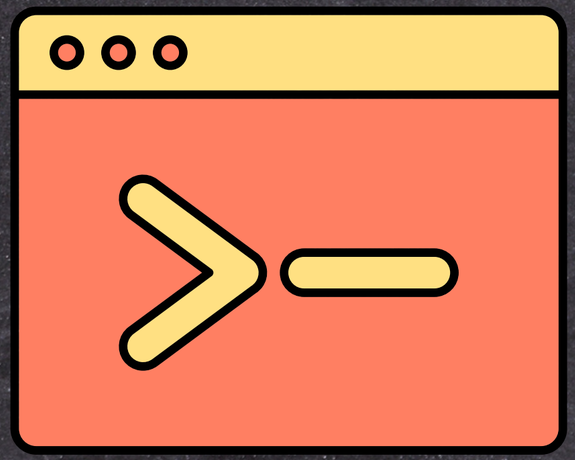
Session 01

# Terminal

In computing, a terminal is a program that allows you to interact with your computer's operating system through a command-line interface. It provides a text-based interface for executing commands and running programs.

**To open the terminal in Windows, you can follow these steps:**

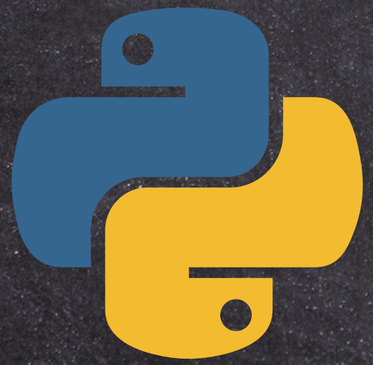
1. Click on the "Start" button in the bottom left corner of your screen.
2. Type "cmd" in the search bar and press Enter. This will open the Command Prompt, which is a basic terminal program that comes with Windows.



M. Amin HosseinNiya | Ali Rashedi

# Python

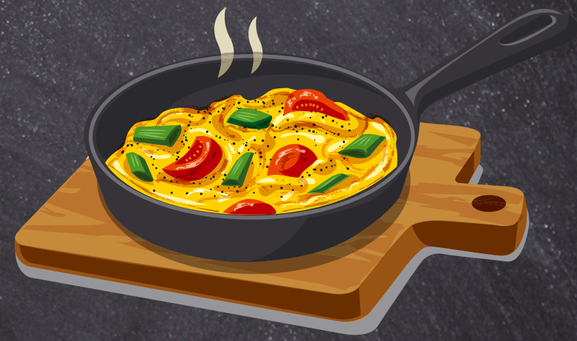
Python is a popular programming language that can be used to write a wide variety of software applications. When working with Python, you can use the terminal to run Python scripts and execute Python commands. This allows you to interact with your Python code and see the output of your programs in real-time.



# Some algorithms

- making a Persian-style omelette

1. Check that you have all necessary ingredients: eggs, tomatoes, onion, salt, pepper, turmeric, and oil.
2. Get a frying pan out of the cabinet and place it on the stove.
3. Pour enough oil into the pan to coat the bottom and heat it over medium heat.
4. While the oil is heating up, dice the tomatoes and onions into small pieces.
5. Once the oil is hot, add the onions to the pan and cook until they become translucent.
6. Add the tomatoes to the pan and cook until they become soft and start to release their juices.
7. Season the mixture with salt, pepper, and a generous amount of turmeric.
8. In a separate bowl, beat the eggs until they are well mixed.
9. Pour the beaten eggs into the pan over the tomato and onion mixture.
10. Cook the omelette over medium-low heat until the eggs are set and the bottom is golden brown.
11. Carefully flip the omelette over and cook the other side until it is golden brown.
12. Slide the omelette onto a plate and enjoy your delicious Persian-style omelette!
13. Turn off the stove and let the pan cool down before washing it.
14. Store any remaining ingredients back in the cabinet and fridge.

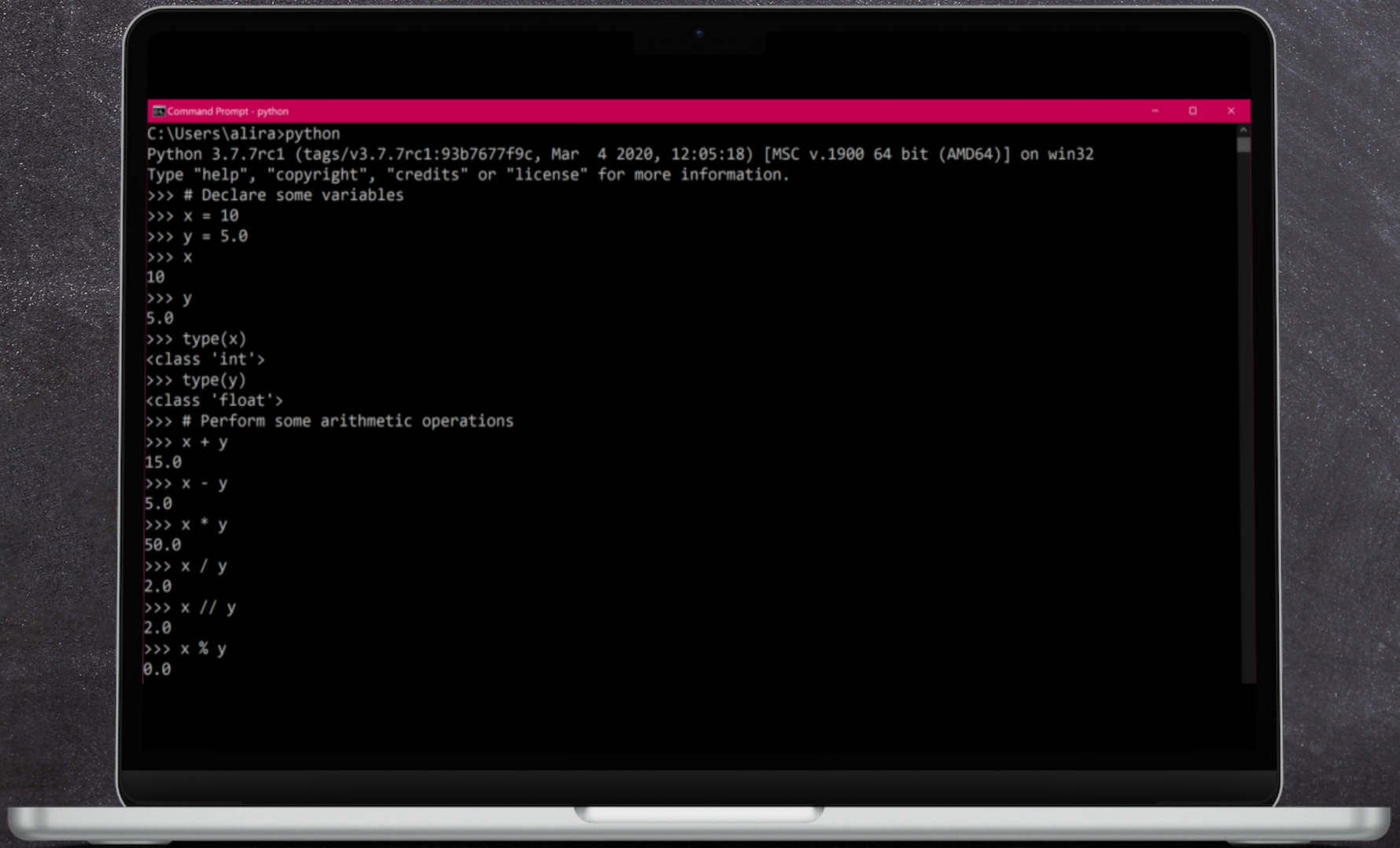


# Some algorithms

- **Algorithm for hair styling with a comb**

1. Determine the desired hairstyle and any necessary styling products.
2. Gather all necessary equipment: a comb or brush, hair tie, and any styling products you wish to use.
3. Starting from the top of your head, use the comb to section off the hair into manageable sections.
4. Comb each section of hair to smooth out any tangles or knots.
5. Use the comb to create the desired shape and volume, directing the hair in the desired direction.
6. If using styling products, apply them as needed to hold the hair in place or provide texture.
7. Repeat steps 3-6 for each section of hair until the entire head of hair has been styled.
8. If necessary, use a hair tie or other accessory to secure any sections of hair that need to be held in place.
9. Check the hair for any stray strands or uneven areas, and use the comb to make any necessary adjustments.
10. Finish by applying any final touches, such as hairspray or shine serum, to complete the hairstyle.

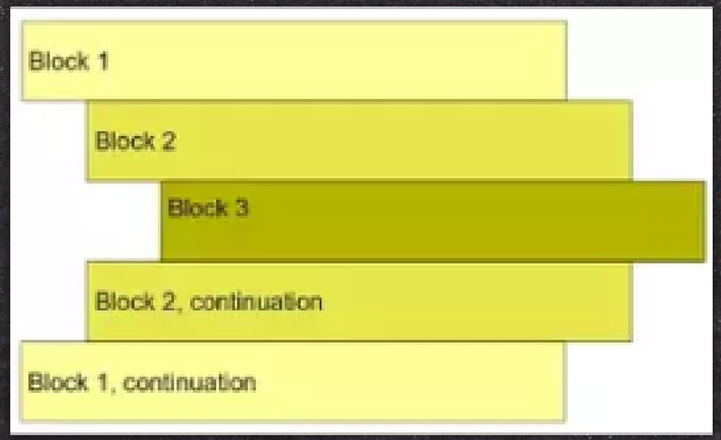
# Working with variables and some simple operators in the terminal:



# Indentation in Python

In Python, indentation is used to group statements together and define the scope of control structures. It's important to use consistent indentation throughout your code, typically four spaces per level. If you don't use proper indentation, you will get a syntax error.

در پایتون از فرورفتگی (فاصله‌ی ابتدای متن از اول خط) برای گروه بندی عبارات با هم و تعریف محدوده ساختارهای کنترلی (شرطی، حلقه و ...) استفاده می شود. مهم است که از فرورفتگی ثابت در سراسر کد خود استفاده کنید، معمولاً چهار فاصله در هر سطح. اگر از فرورفتگی مناسب استفاده نکنید، با یک خطای سینتکس مواجه خواهید شد.




main.py

```
1  if x > 0:
2      print("x is positive")
3  else:
4      print("x is not positive")
```

# Variable names in python

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _ )
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- A variable name cannot be any of the Python keywords.

# Reserved Words

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| False | class | return | is | finally | None | in | as | raise |
| if | for | lambda | continue | True | def | except | elif | import |
| from | while | nonlocal | and | del | global | break | try | pass |
| not | with | or | yield | assert | else | | | |

# Variable Types

- Text Type : str
- Numeric Types: int, float
- Boolean Type: bool

- You can get the data type of any object by using the type() function.
- You can change type of variables.

```
>>> a = 5
>>> b = 'ali'
>>> type(a)
<class 'int'>
>>> type(b)
<class 'str'>
>>> float(a)
5.0
```

# Calculate Pizza Area

pizza_area.py

```python
# we want to calculate pizza area

# first we have to input Radius of pizza
radius = input(' lotfan shoa e pizza ra vared konid: ')

# change type of input to float
radius = float(radius)

# define pi number
pi = 3.1415

# calculate the area
area = (radius ** 2) * pi

# output the resault
print(' The area of the pizza is : ', area, ' cm^2')
```

# Calculate average of scores

```python
# we want to calculate average of some scores for lessons
# first we have to input score for each course
riazi = input(' lotfan nomre riazi ra vared konid: ')
fizik = input(' lotfan nomre fizik ra vared konid: ')
zist = input(' lotfan nomre zist ra vared konid: ')
zaban = input(' lotfan nomre zaban ra vared konid: ')

# define numbers of courses
numbers = 4

# change type of input to float
riazi = float(riazi)
fizik = float(fizik)
zist = float(zist)
zaban = float(zaban)

# define sum of numbers
sum_scores = riazi + fizik + zist + zaban

# calculate the average
average = sum_scores / numbers

# output the resault
print(' The average of the scores is : ', average)
```

# Boolean variables

Boolean variables are variables that can only have two possible values: true or false. In programming, Boolean variables are often used to represent conditions or states that can be either true or false. In many programming languages including Python, the Boolean data type is represented by the keywords "True" and "False".

# Logical Operators

Logical operators are used in programming to combine two or more conditions that result in a Boolean value (true or false). There are three basic logical operators:

1. AND: This operator returns true only if all of the conditions being compared are true. For example, if we have two conditions A and B, `A and B` is true only if both A and B are true.

2. OR: This operator returns true if at least one of the conditions being compared is true. For example, if we have two conditions A and B, `A or B` is true if either A or B is true.

3. NOT: This operator returns the opposite of the condition being compared. If the condition is true, the NOT operator returns false, and if the condition is false, the NOT operator returns true.

Logical operators are commonly used in control structures, such as if-else statements and loops (which we discuss them next session), to determine the flow of the program based on multiple conditions.

# Logical Operators

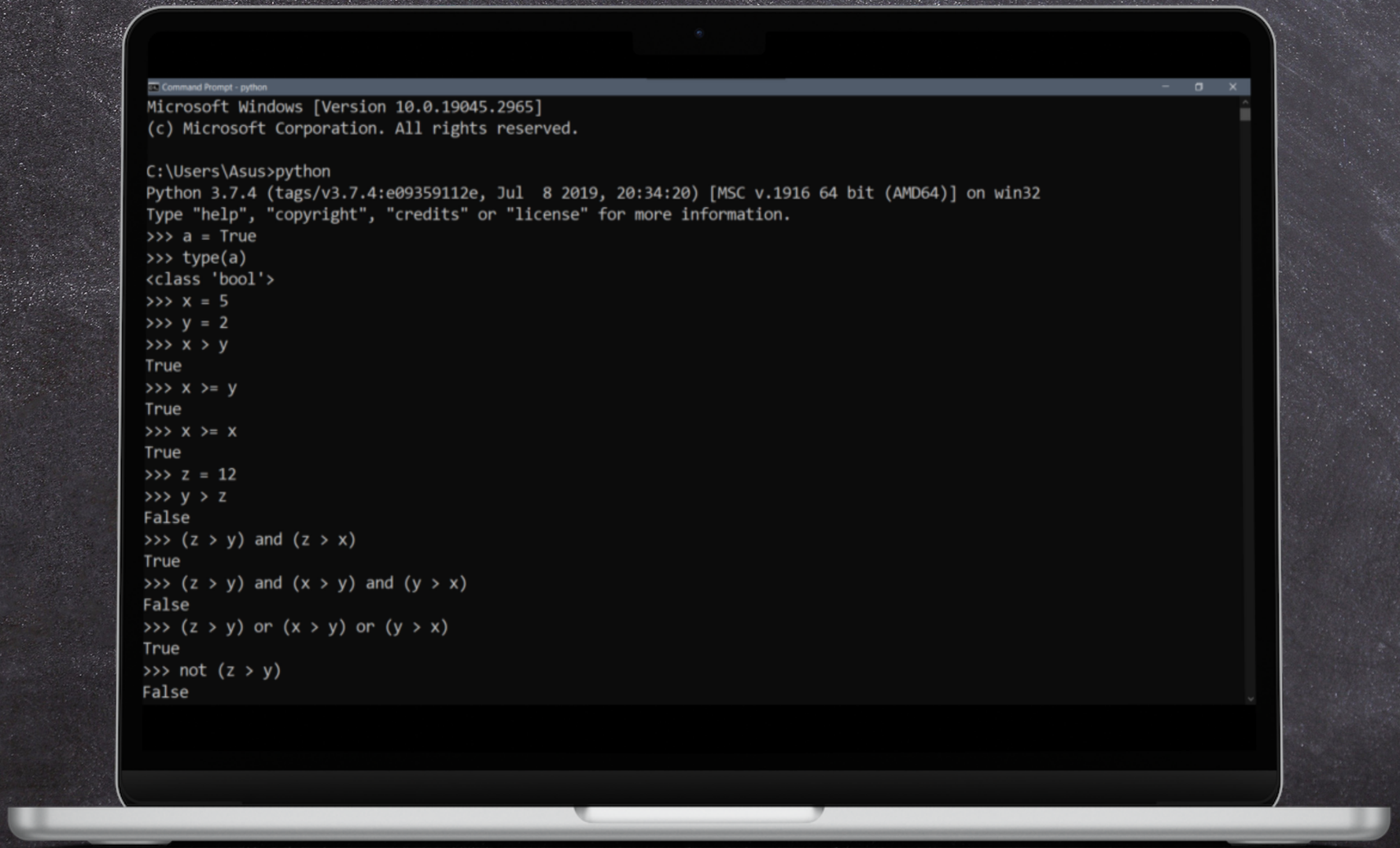
How <or> and <and> operators work.

| x | y | x or y |
|---|---|---|
| + | + | + |
| + | - | + |
| - | + | + |
| - | - | - |

| x | y | x and y |
|---|---|---|
| + | + | + |
| + | - | - |
| - | + | - |
| - | - | - |

Note that it's not necessary to give two operands to <or> and <and> operators. Point is that only if all of conditions (regardless of number of them) are true, then <and> returns true, and even if one of the conditions (again regardless of number of them) is true, then <or> returns true.

# Working with Boolean variables and logical operators in the terminal:

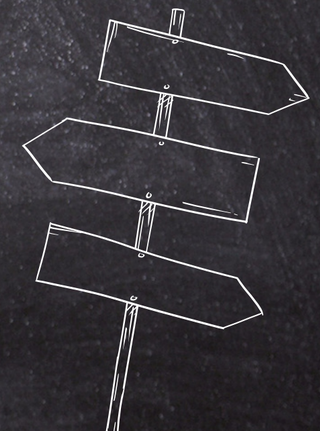


# Conditional statements

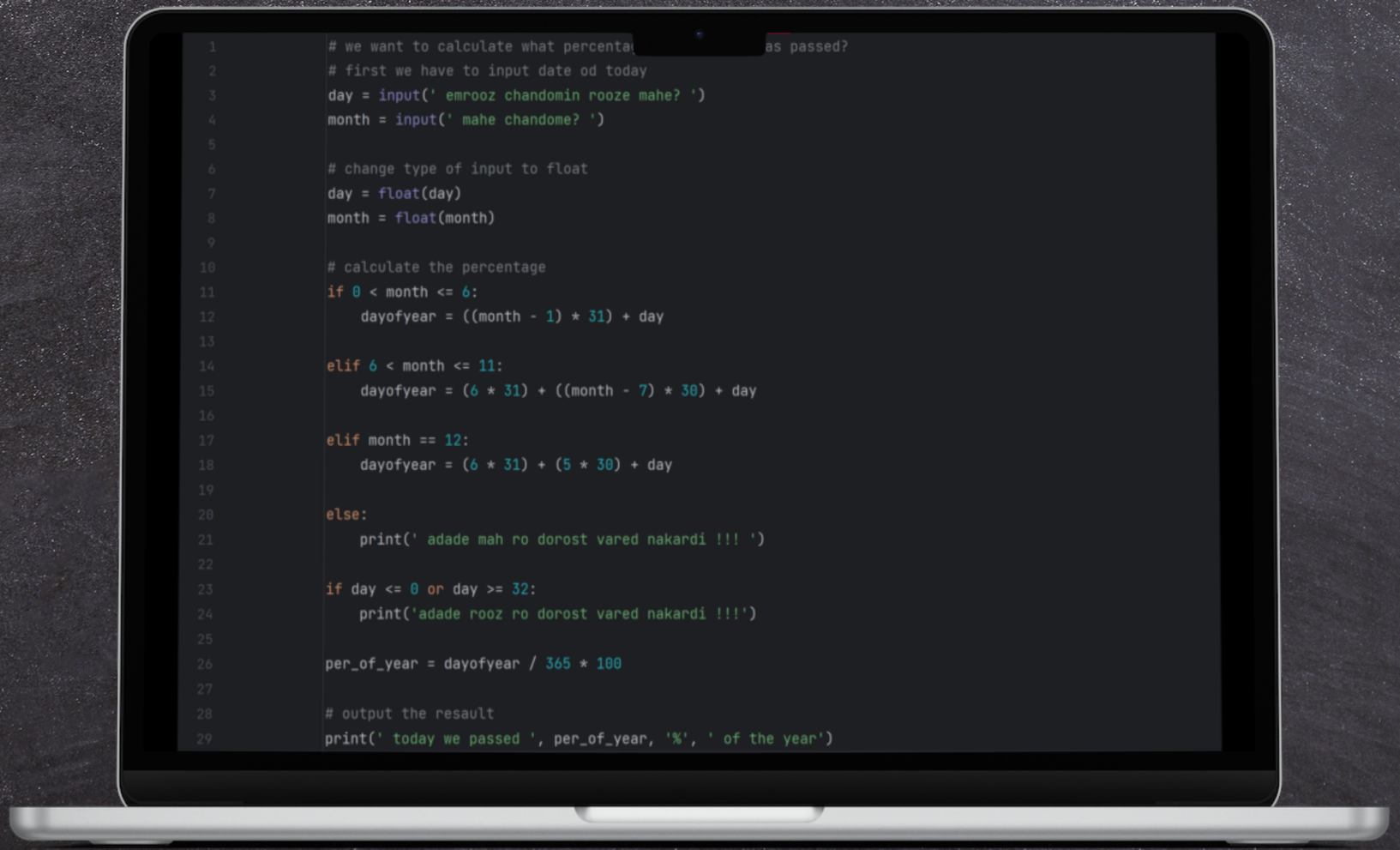
if - elif - else...

Conditional statements in Python are used to execute a block of code only if a certain condition is met. The most commonly used conditional statements in Python are the if statement, the elif statement, and the else statement.

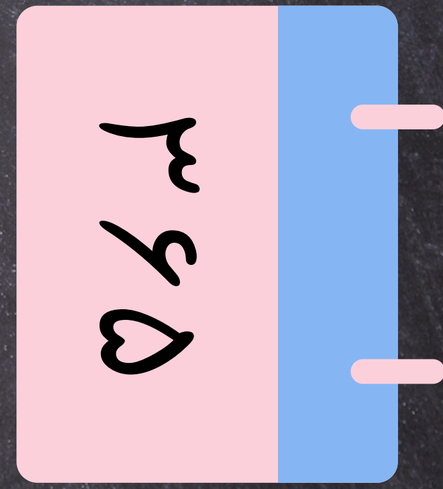
- The if statement is used to execute a block of code if a certain condition is true.
- The elif statement is used to check for additional conditions if the previous condition(s) were not true.
- The else statement is used to execute a block of code if none of the previous conditions were true.



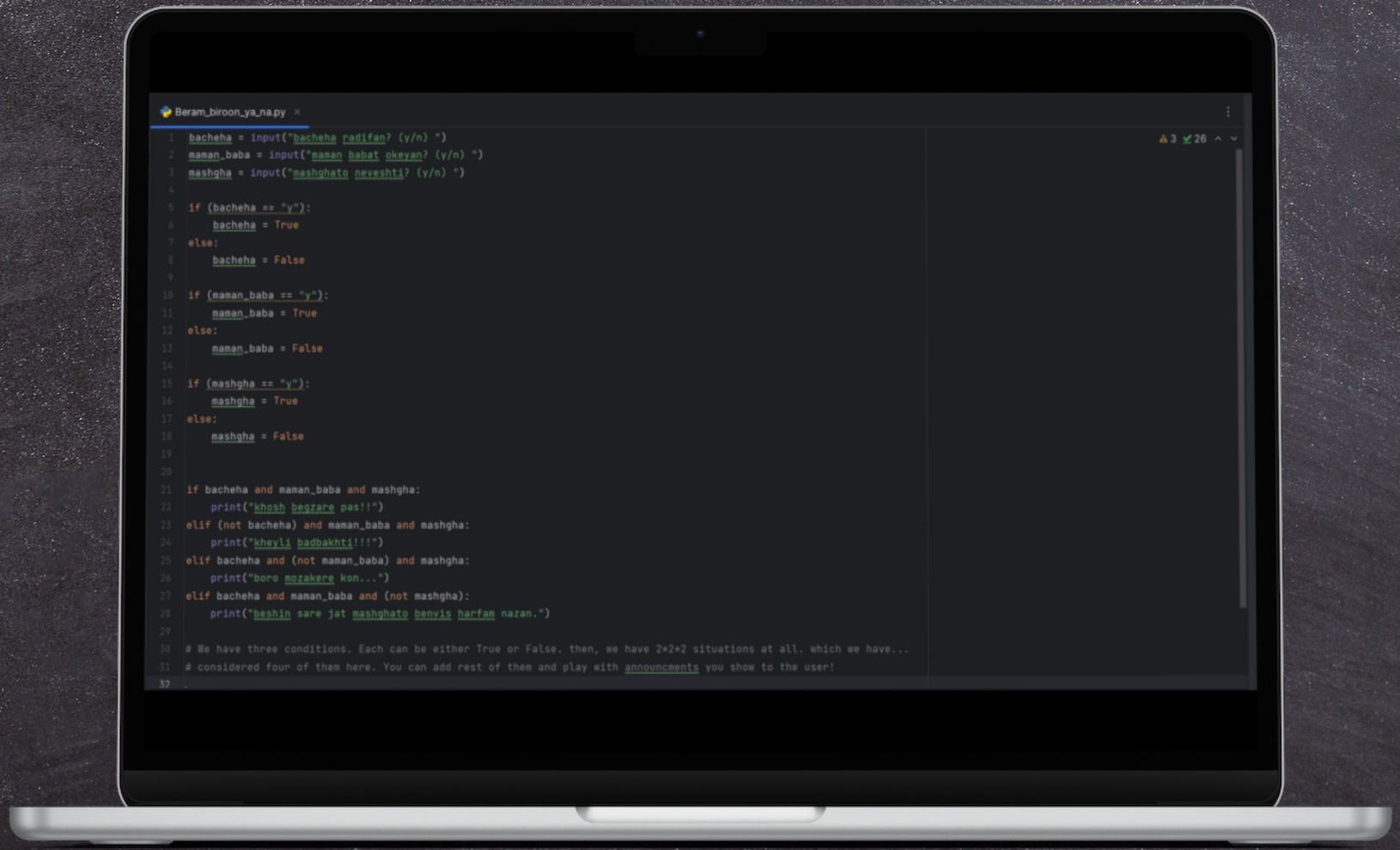
# What percentage of the year has passed?



```python
# we want to calculate what percenta[redacted]as passed?
# first we have to input date od today
day = input(' emrooz chandomin rooze mahe? ')
month = input(' mahe chandome? ')

# change type of input to float
day = float(day)
month = float(month)

# calculate the percentage
if 0 < month <= 6:
    dayofyear = ((month - 1) * 31) + day

elif 6 < month <= 11:
    dayofyear = (6 * 31) + ((month - 7) * 30) + day

elif month == 12:
    dayofyear = (6 * 31) + (5 * 30) + day

else:
    print(' adade mah ro dorost vared nakardi !!! ')

if day <= 0 or day >= 32:
    print('adade rooz ro dorost vared nakardi !!!')

per_of_year = dayofyear / 365 * 100

# output the resault
print(' today we passed ', per_of_year, '%', ' of the year')
```



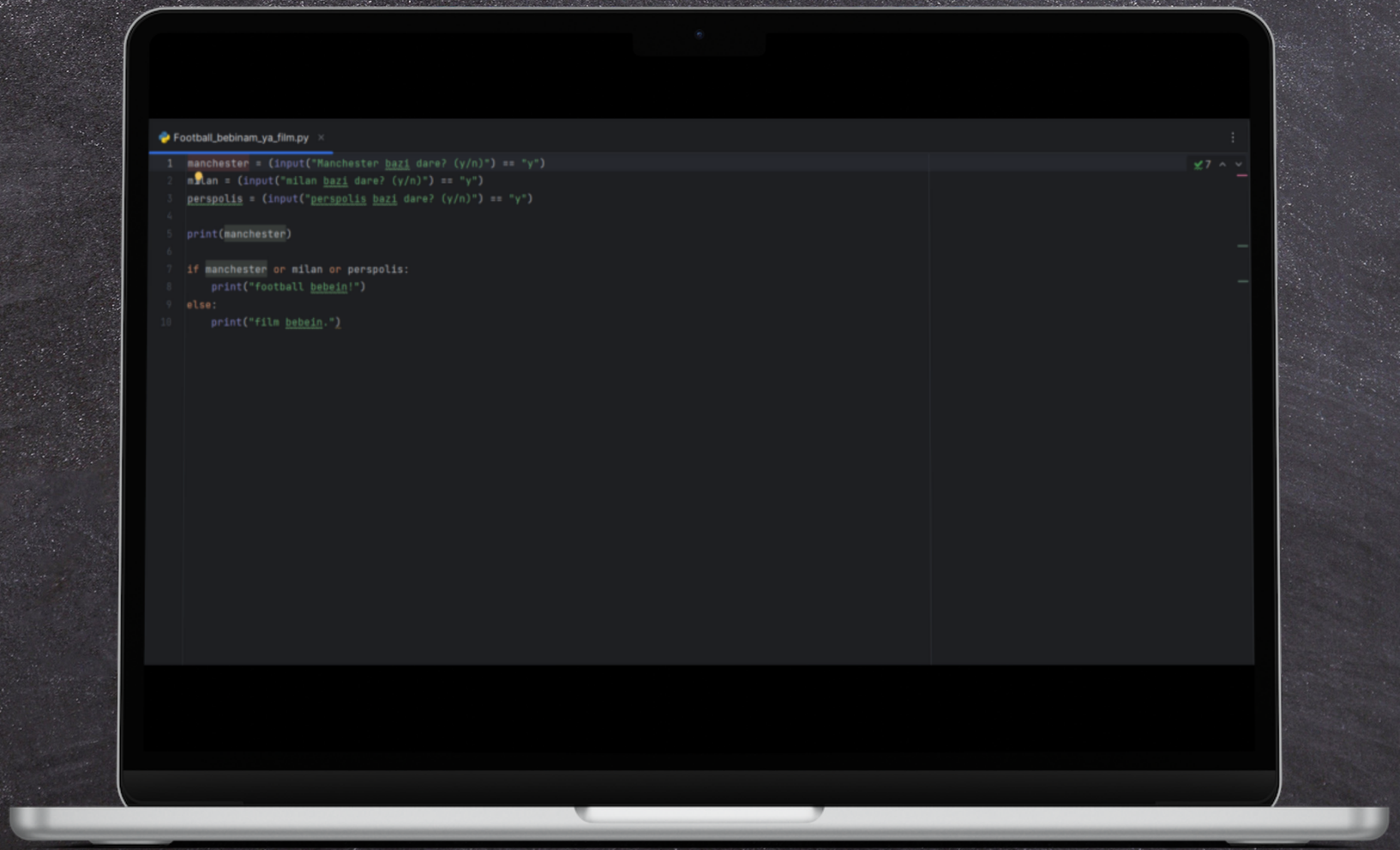
# To go or not to go???





One of my hearts says go, one of my hearts says don't go...

# Film or Foorball???





I believe this code technically can generate both answers, so I wonder why I've watched three movies ever.  two of them  were about football. Third one is forgotten by now. Might better you not use it as your daily planner.

# Presented by

Ali Rashedi

alirashedi2014@gmail.com

ali-rashedi

M. Amin HosseinNiya

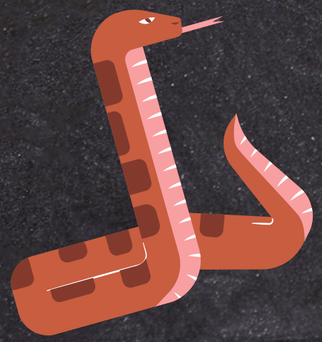
m.amin.hosseinniya@gmail.com

mohammadamin-hosseinniya

M. Amin HosseinNiya | Ali Rashedi

# Python Programming

## A brief of the content!



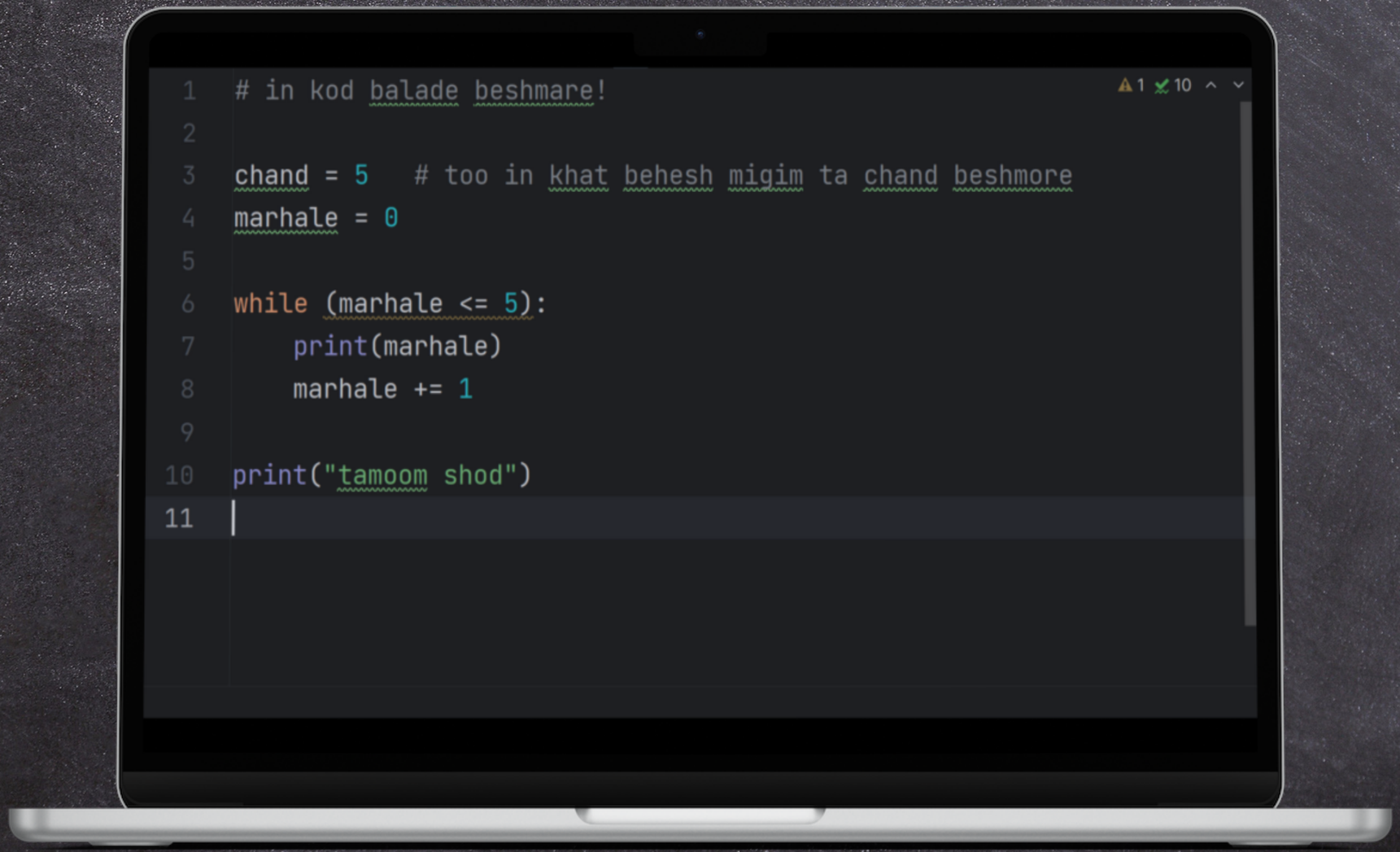
Session 02

M. Amin HosseinNiya | Ali Rashedi

# While loop

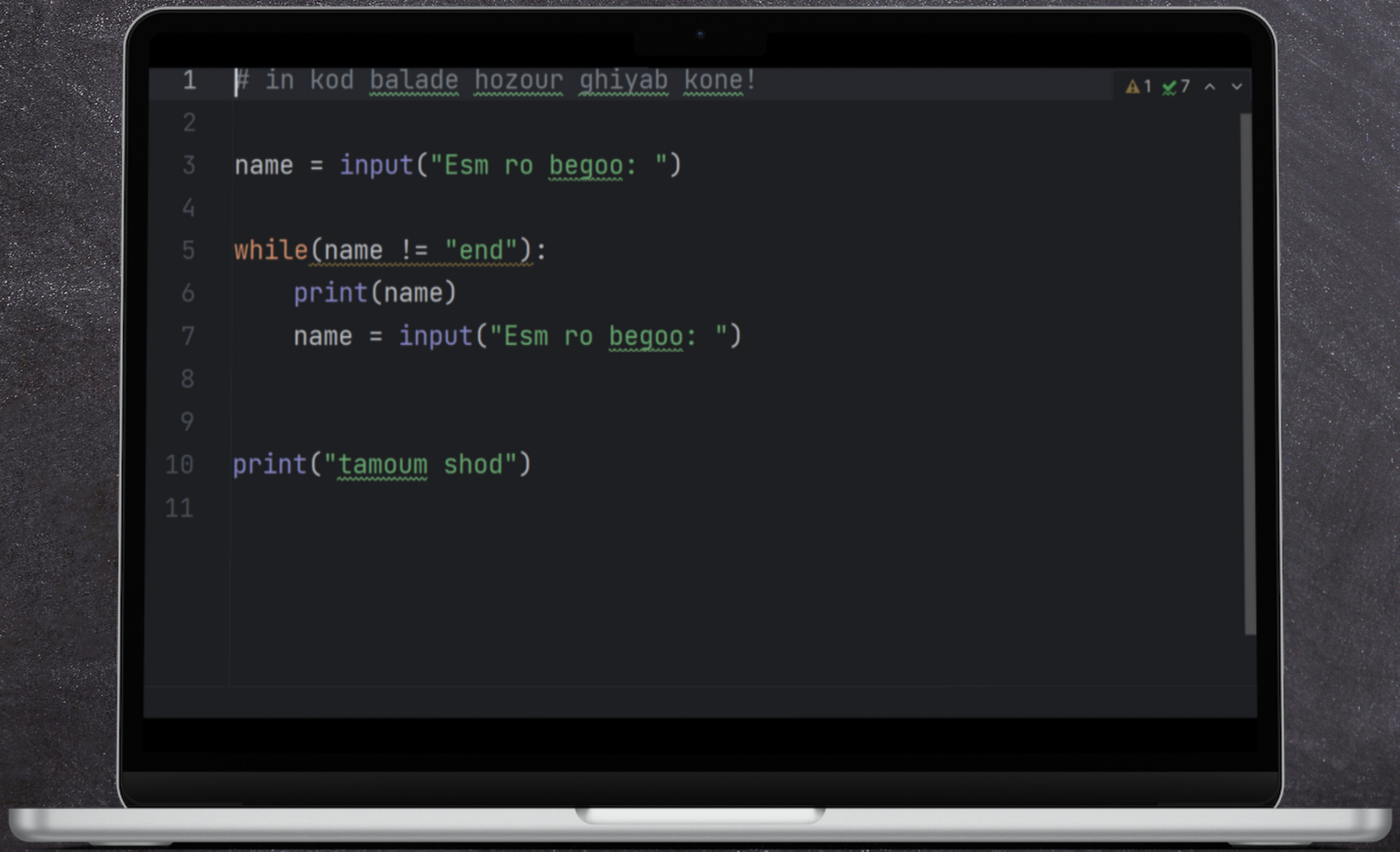
A 'While' loop is a type of loop in Python that allows you to repeat a block of code while a certain condition is true.

# While loop in action




While_example1_Beshmar.py file

# While loop in action

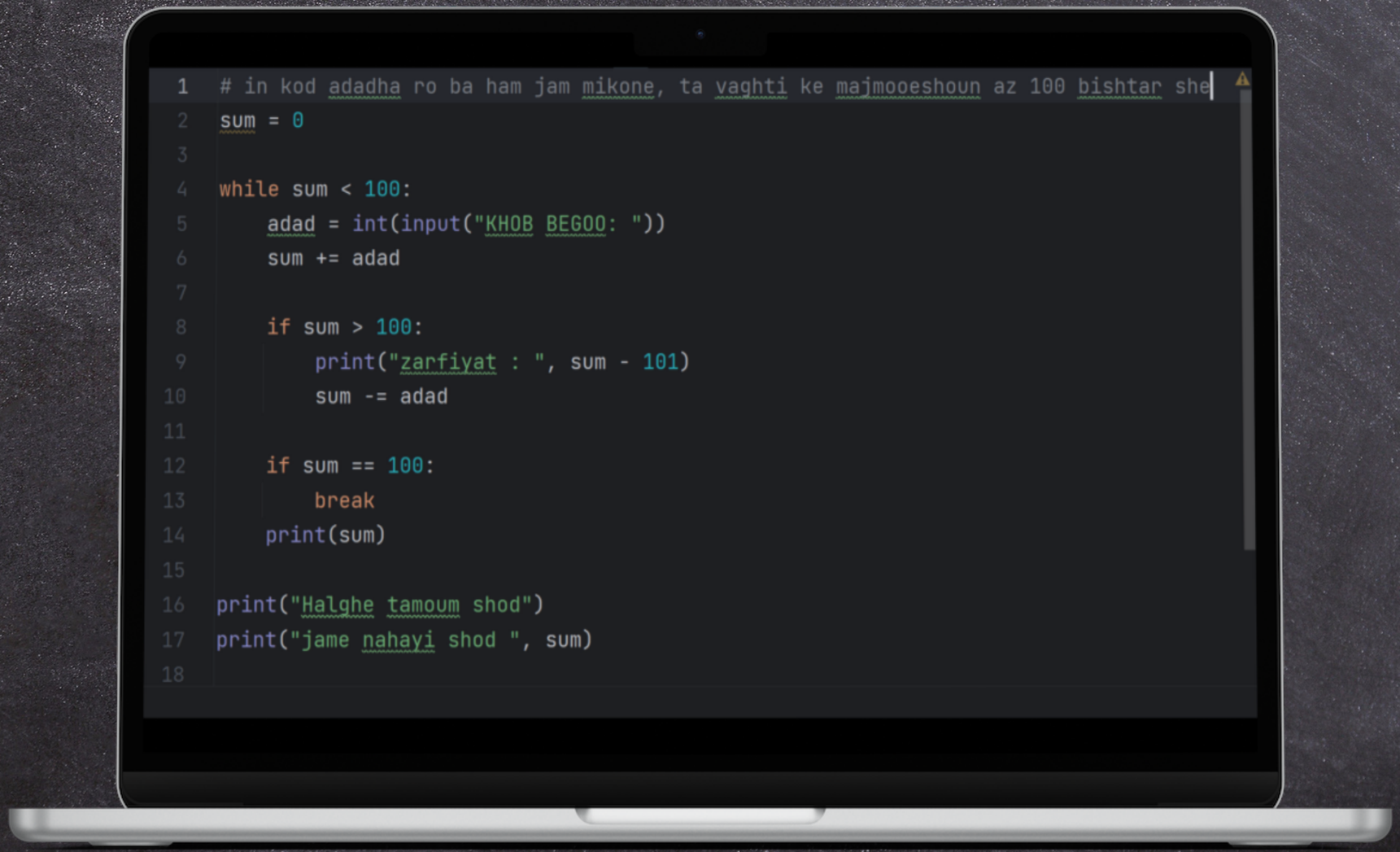



While_example2_HozourGhiyab.py file

# Break statement!

A 'break' statement is a way to exit a loop prematurely, even if the loop condition is still true.

# Break statement in action



```python
# in kod adadha ro ba ham jam mikone, ta vaghti ke majmooeshoun az 100 bishtar she
sum = 0

while sum < 100:
    adad = int(input("KHOB BEGOO: "))
    sum += adad

    if sum > 100:
        print("zarfiyat : ", sum - 101)
        sum -= adad

    if sum == 100:
        break
    print(sum)

print("Halghe tamoum shod")
print("jame nahayi shod ", sum)
```

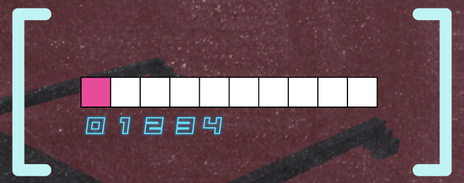
While_example3_JamZadan.py

M. Amin HosseinNiya | Ali Rashedi

# Liiiist

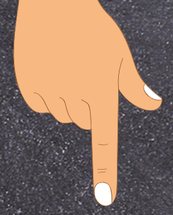
Lists are one of the fundamental data structures in Python. They are used to store and organize collections of items. Here are some key characteristics of lists:

- Ordered: Lists maintain the order of elements as they are added.
- Mutable: Elements in a list can be modified, added, or removed.
- Heterogeneous: Lists can contain different data types, such as numbers, strings, or even other lists.



```python
fruits = ["apple", "banana", "orange", "grape"]
```

In this example, we have a list called "fruits" that contains four elements. The order of the elements is preserved, and we can access each element by its position in the list.



# Indexing in list

In Python, elements in a list can be accessed using indexes. The index represents the position of an element in the list. It's important to note that indexing in Python starts from 0. Here's an example to illustrate list indexing:

NEXT ➡

```python
fruits = ["apple", "banana", "orange", "grape"]

# Accessing elements using positive indexes
print(fruits[0])   # Output: "apple"
print(fruits[2])   # Output: "orange"

# Accessing elements using negative indexes
print(fruits[-1])  # Output: "grape"
print(fruits[-3])  # Output: "banana"
```

In this example, we have a list called "fruits." By specifying the index in square brackets, we can access individual elements in the list. Positive indexes count from the beginning of the list, while negative indexes count from the end.

List indexing allows us to retrieve and manipulate specific elements within a list, enabling us to perform various operations and computations on our data.

# Basic List Operations

append():

- Syntax: list.append(element)
- Adds an element to the end of the list.
- Example:

```python
numbers = [1, 2, 3]
numbers.append(4)
print(numbers)  # Output: [1, 2, 3, 4]
```

pop():

- Syntax: list.pop()
- Removes and returns the last element from the list.
- Example:

```python
numbers = [1, 2, 3, 4]
last_number = numbers.pop()
print(last_number)  # Output: 4
print(numbers)      # Output: [1, 2, 3]
```

# Basic List Operations

remove():

- Syntax: list.remove(element)
- Removes the first occurrence of the specified element from the list.
- Example:

```python
fruits = ["apple", "banana", "orange", "banana"]
fruits.remove("banana")
print(fruits)  # Output: ["apple", "orange", "banana"]
```

sort():

- Syntax: list.sort()
- Sorts the elements of the list in ascending order.
- Example:

```python
numbers = [4, 2, 1, 3]
numbers.sort()
print(numbers)  # Output: [1, 2, 3, 4]
```

# Basic List Operations

len():

- Syntax: len(list)
- Returns the number of elements in the list.
- Example:

```python
fruits = ["apple", "banana", "orange"]
length = len(fruits)
print(length)  # Output: 3
```

sum():

- Syntax: sum(list)
- Calculates the sum of all the elements in the list (if they are numeric).
- Example:

```python
numbers = [1, 2, 3, 4]
total = sum(numbers)
print(total)  # Output: 10
```

| Method | Description |
|---|---|
| append() | Adds an element at the end of the list |
| clear() | Removes all the elements from the list |
| copy() | Returns a copy of the list |
| count() | Returns the number of elements with the specified value |
| extend() | Add the elements of a list (or any iterable), to the end of the current list |
| index() | Returns the index of the first element with the specified value |
| insert() | Adds an element at the specified position |
| pop() | Removes the element at the specified position |
| remove() | Removes the item with the specified value |
| reverse() | Reverses the order of the list |
| sort() | Sorts the list |

M. Amin HosseinNiya | Ali Rashedi

# Fooooooor

For loops in Python are a powerful construct that allows us to iterate over a sequence of elements. They are used when we want to perform a set of actions repeatedly for each item in the sequence. The general syntax of a for loop in Python is as follows:

python

```python
fruits = ["apple", "banana", "orange"]

for fruit in fruits:
    print(fruit)
```

# Break and Continue

Python provides two useful statements, "break" and "continue", to control the flow of execution within loops.

The "break" statement is used to prematurely exit the loop when a certain condition is met. It completely terminates the loop and continues execution with the next statement outside of the loop.

On the other hand, the "continue" statement is used to skip the rest of the current iteration and move on to the next iteration of the loop.

python

```python
fruits = ["apple", "banana", "orange"]

for fruit in fruits:
    if fruit == "banana":
        break
    print(fruit)
```

python

```python
fruits = ["apple", "banana", "orange"]

for fruit in fruits:
    if fruit == "banana":
        continue
    print(fruit)
```

# Presented by

Ali Rashedi

alirashedi2014@gmail.com

ali-rashedi

M. Amin HosseinNiya

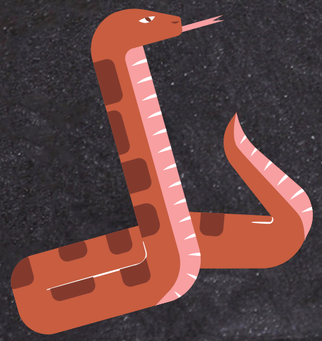
m.amin.hosseinniya@gmail.com

mohammadamin-hosseinniya

M. Amin HosseinNiya | Ali Rashedi

# Python Programming

## A brief of the content!



Session 03

M. Amin HosseinNiya | Ali Rashedi

# Functions

A function is a block of organized, reusable code that performs a specific task. It takes in input, performs some operations, and produces an output. Functions allow us to break down complex programs into smaller, manageable pieces. This promotes code reusability and enhances the overall structure of our code.

# Defining a Function

To define a function in Python, we use the def keyword, followed by the function name, parentheses, and a colon. Let's take a look at the basic syntax:

python Copy code

```python
def function_name(parameters):
    # Code block
    # Perform operations
    # Return a value (optional)
```

# Calling a Function]

Once we have defined a function, we can call it from other parts of our code. To call a function, we simply write the function name followed by parentheses. Let's see an example:

python Copy code

```python
# Function definition
def greet():
    print("Hello, world!")

# Function call
greet()
```

# Arguments

Functions can also accept parameters, which are input values passed to the function. Parameters allow us to make our functions more flexible and versatile. Here's an example:

python Copy code

```python
# Function definition with parameters
def greet(name):
    print("Hello, " + name + "!")

# Function call with arguments
greet("Alice")
greet("Bob")
```

```
Hello, Alice!
Hello, Bob!
```

# Return Statement

Sometimes, we want our functions to produce an output or a result. We can achieve this using the return statement. The return statement allows us to specify the value that the function should return back to the caller. Let's see an example:

python Copy code

```python
# Function definition with return statement
def add_numbers(a, b):
    return a + b

# Function call and storing the result
result = add_numbers(5, 3)
print("The sum is:", result)
```

```
The sum is: 8
```

# Modules

In Python, a module is a file containing Python definitions and statements that can be imported into other Python scripts or modules. This allows you to reuse code and organize your code into logical units. Python has a large standard library of modules that provide a wide range of functionality. You can also create your own modules to encapsulate related functionality and make it easier to reuse in other projects. To use a module in your Python code, you can import it using the import statement and access the functions and variables defined in the module using the dot notation.

# How to use Modules

1

```python
import random

random_number = random.randint(1, 10)
print("Random number: ", random_number)

```

Import the entire library and call the functions you need using a dot notation

2

```python
import random as rnd

random_number = rnd.randint(1, 10)
print("Random number: ", random_number)

```

Import the entire library with a short nickname!

3

```python
from random import randint

random_number = randint(1, 10)
print("Random number: ", random_number)

```

Import only the one function you need.

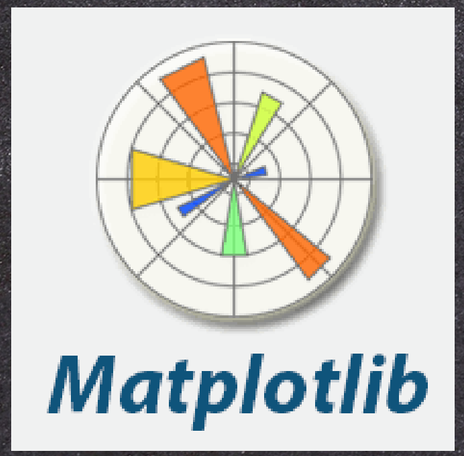
# To install libraries you don't have already...

First of all, don't forget to search for installing the library in hand. This will usually lead you to install that package very easily. However, this is a general guide to install new libraries:

You can use a package manager such as `pip`. Here are the steps to install a library using `pip`:

1. Open a command prompt or terminal window.
2. Type `pip install <library_name>` and press Enter. Replace `<library_name>` with the name of the library you want to install.
3. Wait for `pip` to download and install the library and its dependencies. This may take a few minutes depending on the size of the library and your internet connection speed.
4. Once the installation is complete, you can import the library in your Python code using the `import` statement.

# Two well-known libraries:




To visualize almost anything


Numerical Python!

M. Amin HosseinNiya | Ali Rashedi

# String things

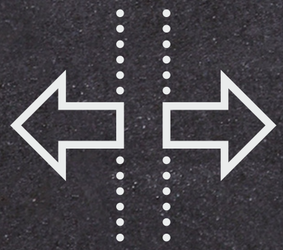
# String methods

.strip()

Aa

.upper() and .lower()

.replace()



.split("something")

something.join(a list)

**And a lot more more methods, waiting you to discover them!**

# Escape Characters!

| Code | Description |
|---|---|
| \' | Single quotation |
| \\ | Backslash |
| \n | New Line |
| \r | Carriage return |
| \t | Tab |
| \b | Backspace |
| \f | Form feed |
| \ooo | Octal equivalent |
| \xhhh | Hexadecimal equivalent |

And of course, more to discover.

# Presented by

Ali Rashedi

alirashedi2014@gmail.com

ali-rashedi

M. Amin HosseinNiya

m.amin.hosseinniya@gmail.com

mohammadamin-hosseinniya