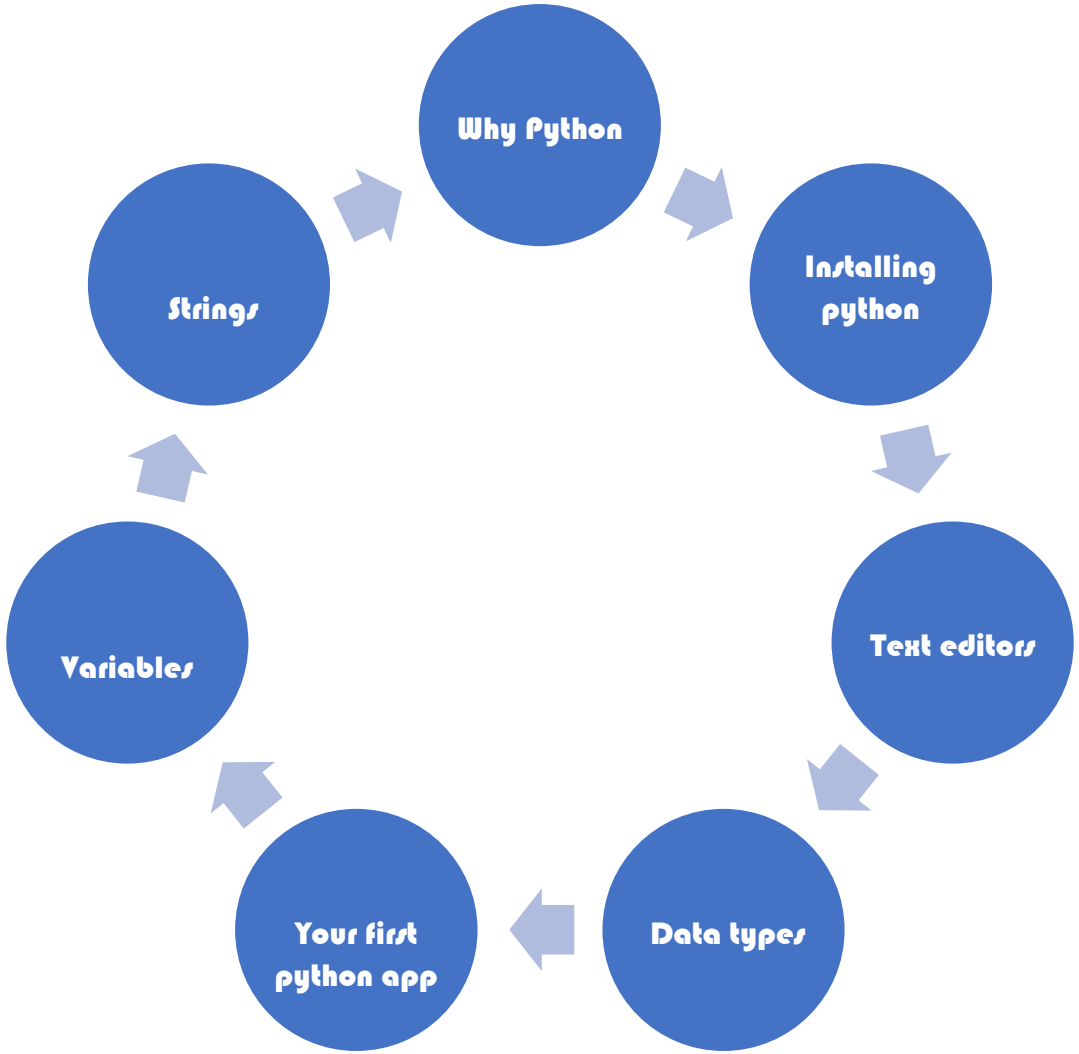
INTRODUCTION TO PYTHON PROGRAMMING

- Lecture one -

PRESENTED BY YOUSSEF KHALIL

Session Agenda



WHY PYTHON ?

- Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

- High level language.

- Supports OOP.

- Python community is very wide.

HOW TO INSTALL PYTHON ?

EMPIRE

- Go to Python official website

Active Python Releases

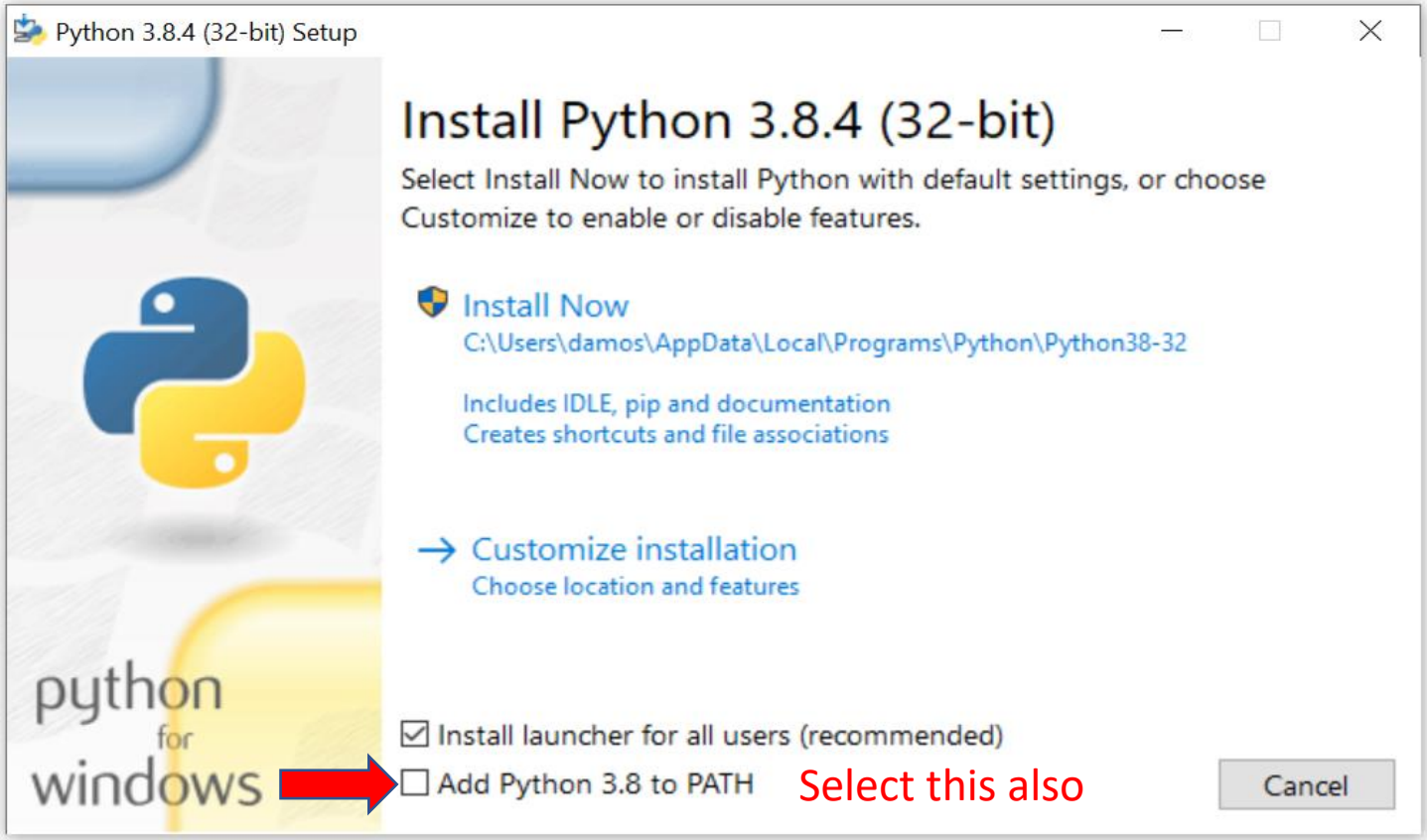
For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.9	bugfix	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
3.6	security	2016-12-23	2021-12-23	PEP 494
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

HOW TO INSTALL PYTHON ?

EMPIRE

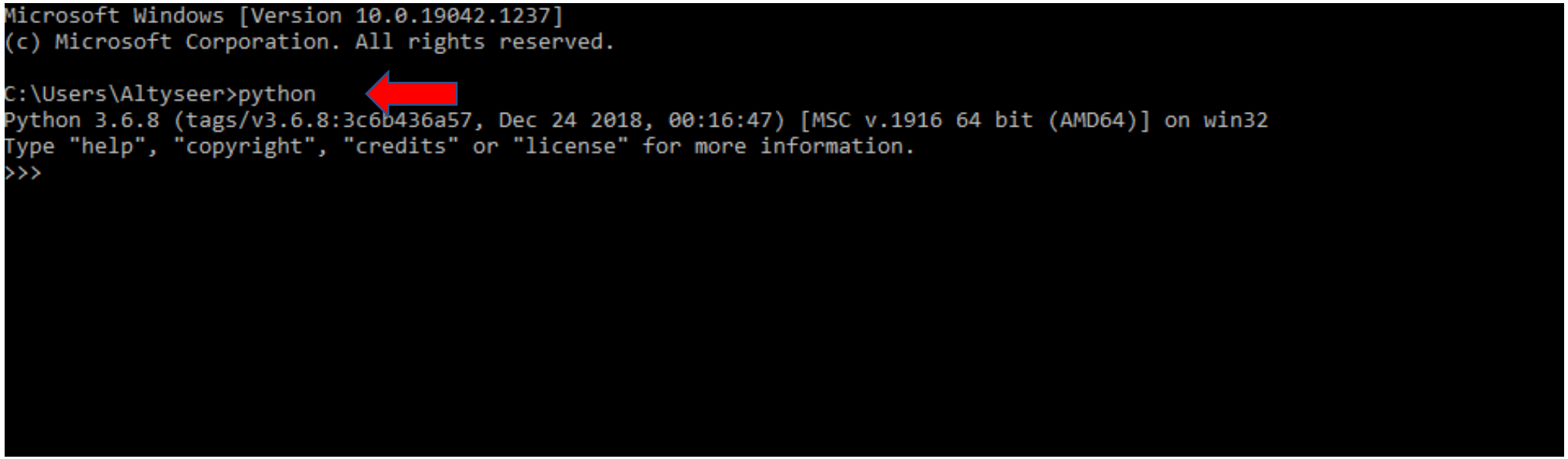
- Run the file you just downloaded



HOW TO INSTALL PYTHON ?

EMPIRE

- Open the command prompt and write python if you got the python version like 3.9.1 so python had been downloaded successfully.

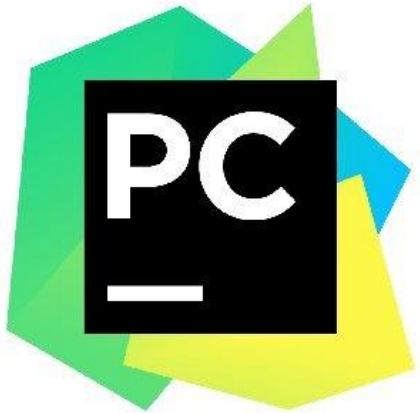


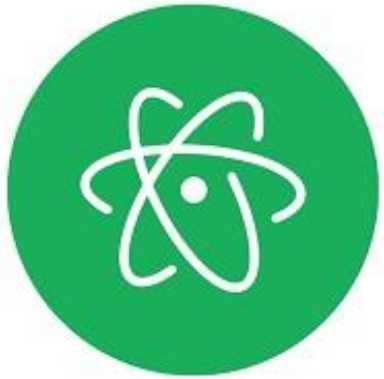
HOW TO INSTALL TEXT EDITOR ?

EMPIRE

- There are many text editors we use to make our coding easier like Vs code, PyCharm, atom and many other Editors.

I prefer Vs code or PyCharm. Press on the image and you will go to the website to download.





USEFUL SOURCE

Here I will add Useful YouTube Videos to help you to install python if you prefer videos

Video in Arabic

Video in English

DATA TYPES

Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`

Getting Started With Python

3.1. Using Python as a Calculator

Let's try some simple Python commands. Start the interpreter and wait for the primary prompt, `>>>`. (It shouldn't take long.)

3.1.1. Numbers ¶

The interpreter acts as a simple calculator: you can type an expression at it and it will write the value. Expression syntax is straightforward: the operators `+`, `-`, `*` and `/` work just like in most other languages (for example, Pascal or C); parentheses (`()`) can be used for grouping. For example:

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5  # division always returns a floating point number
1.6
```

Getting Started With Python


```
>>> 17 / 3  # classic division returns a float
5.666666666666667
>>>
>>> 17 // 3  # floor division discards the fractional part
5
>>> 17 % 3  # the % operator returns the remainder of the division
2
>>> 5 * 3 + 2  # floored quotient * divisor + remainder
17
```

With Python, it is possible to use the `**` operator to calculate powers [1]:

```
>>> 5 ** 2  # 5 squared
25
>>> 2 ** 7  # 2 to the power of 7
128
```

The equal sign (`=`) is used to assign a value to a variable. Afterwards, no result is displayed before the next interactive prompt:

Variables in Python

Example

```
x = 5
y = "John"
print(x)
print(y)
```

Example

```
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)
```

CASTING AND TYPE FUNCTION

EMPIRE

Casting

If you want to specify the data type of a variable, this can be done with casting.

Example

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

Get the Type

You can get the data type of a variable with the `type()` function.

Example

```
x = 5
y = "John"
print(type(x))
print(type(y))
```

PYTHON IS CASE SENSITIVE

Case-Sensitive

Variable names are case-sensitive.

Example

This will create two variables:

```
a = 4
A = "Sally"
#A will not overwrite a
```

NOTE: the variable a is not the same A. The variable ball is not the same as Ball

TRY TO DO SOME CALCULATIONS USING VARIABLES


```
>>> tax = 12.5 / 100
>>> price = 100.50
>>> price * tax
12.5625
>>> price + _
113.0625
>>> round(_, 2)
113.06
```

IT'S YOUR TURN NOW !!

MATHEMATICAL OPERATIONS IN PYTHON

Operator	Description	Syntax
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when first operand is divided by the second	x % y
**	Power : Returns first raised to power second	x ** y

USEFUL LINK FOR MATHEMATICAL OPERATIONS

STRINGS

EMPIRE

Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.

`'hello'` is the same as `"hello"`.

You can display a string literal with the `print()` function:

Example

```
print("Hello")
print('Hello')
```

STRINGS

EMPIRE

Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

Example

```
a = "Hello"
print(a)
```

STRINGS

EMPIRE

Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

Example

```
a = "Hello"
print(a)
```

STRINGS ARE ARRAYS

Strings are Arrays

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.

However, Python does not have a character data type, a single character is simply a string with a length of 1.

Square brackets can be used to access elements of the string.

Example

Get the character at position 1 (remember that the first character has the position 0):

```
a = "Hello, World!"
print(a[1])
```

STRINGS ARE ARRAYS

String Length

To get the length of a string, use the `len()` function.

Example

The `len()` function returns the length of a string:

```
a = "Hello, World!"
print(len(a))
```

STRINGS ARE ARRAYS

Looping Through a String

Since strings are arrays, we can loop through the characters in a string, with a `for` loop.

Example

Loop through the letters in the word "banana":

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

STRINGS ARE ARRAYS

Check String

To check if a certain phrase or character is present in a string, we can use the keyword `in`.

Example

Check if "free" is present in the following text:

```
txt = "The best things in life are free!"
print("free" in txt)
```

STRINGS ARE ARRAYS

Use it in an `if` statement:

Example

Print only if "free" is present:

```
txt = "The best things in life are free!"
if "free" in txt:
  print("Yes, 'free' is present.")
```

STRINGS ARE ARRAYS

Check if NOT

To check if a certain phrase or character is NOT present in a string, we can use the keyword `not in`.

Example

Check if "expensive" is NOT present in the following text:

```
txt = "The best things in life are free!"
print("expensive" not in txt)
```

STRINGS ARE ARRAYS

Slicing

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

Example

Get the characters from position 2 to position 5 (not included):

```
b = "Hello, World!"
print(b[2:5])
```

STRINGS ARE ARRAYS

Slice From the Start

By leaving out the start index, the range will start at the first character:

Example

Get the characters from the start to position 5 (not included):

```
b = "Hello, World!"
print(b[:5])
```

STRINGS ARE ARRAYS

Slice To the End

By leaving out the *end* index, the range will go to the end:

Example

Get the characters from position 2, and all the way to the end:

```
b = "Hello, World!"
print(b[2:])
```

MODIFY STRINGS

Upper Case

Example

The `upper()` method returns the string in upper case:

```
a = "Hello, World!"
print(a.upper())
```

MODIFY STRINGS

Lower Case

Example

The `lower()` method returns the string in lower case:

```
a = "Hello, World!"
print(a.lower())
```

MODIFY STRINGS

Remove Whitespace

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

Example

The `strip()` method removes any whitespace from the beginning or the end:

```
a = " Hello, World! "
print(a.strip()) # returns "Hello, World!"
```

MODIFY STRINGS

Replace String

Example

The `replace()` method replaces a string with another string:

```
a = "Hello, World!"
print(a.replace("H", "J"))
```

MODIFY STRINGS

Split String

The `split()` method returns a list where the text between the specified separator becomes the list items.

Example

The `split()` method splits the string into substrings if it finds instances of the separator:

```
a = "Hello, World!"
print(a.split(",")) # returns ['Hello', ' World!']
```

CONCATENATE STRINGS

EMPIRE

String Concatenation

To concatenate, or combine, two strings you can use the + operator.

Example

Merge variable `a` with variable `b` into variable `c` :

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

CONCATENATE STRINGS

Example

To add a space between them, add a `" "`:

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

CONCATENATE STRINGS

String Format

As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:

Example

```
age = 36
txt = "My name is John, I am " + age
print(txt)
```

But we can combine strings and numbers by using the `format()` method!

The `format()` method takes the passed arguments, formats them, and places them in the string where the placeholders `{}` are:

CONCATENATE STRINGS

Example

Use the `format()` method to insert numbers into strings:

```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
```

Try it Yourself »

The format() method takes unlimited number of arguments, and are placed into the respective placeholders:

CONCATENATE STRINGS

Example

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```

CONCATENATE STRINGS

You can use index numbers `{0}` to be sure the arguments are placed in the correct placeholders:

Example

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

STRINGS METHODS

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdecimal()	Returns True if all characters in the string are decimals

FOR MORE STRINGS METHODS

STRINGS METHODS

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdecimal()	Returns True if all characters in the string are decimals

FOR MORE STRINGS METHODS

EXERCISE TIME

STRING SPLIT AND JOIN

WHAT IS YOUR NAME

USEFUL MATERIALS

EMPIRE

PYTHON OFFICIAL TUTORIAL

W3SCHOOL

GEEKSFORGEEKS

EL ZERO WEB SCHOOL

THE END

EMPIRE

AND THAT BRINGS US TO THE END.

I'D LIKE TO THANK YOU FOR YOUR TIME AND ATTENTION TODAY.

By: Youssef M. Khalil

01151751949