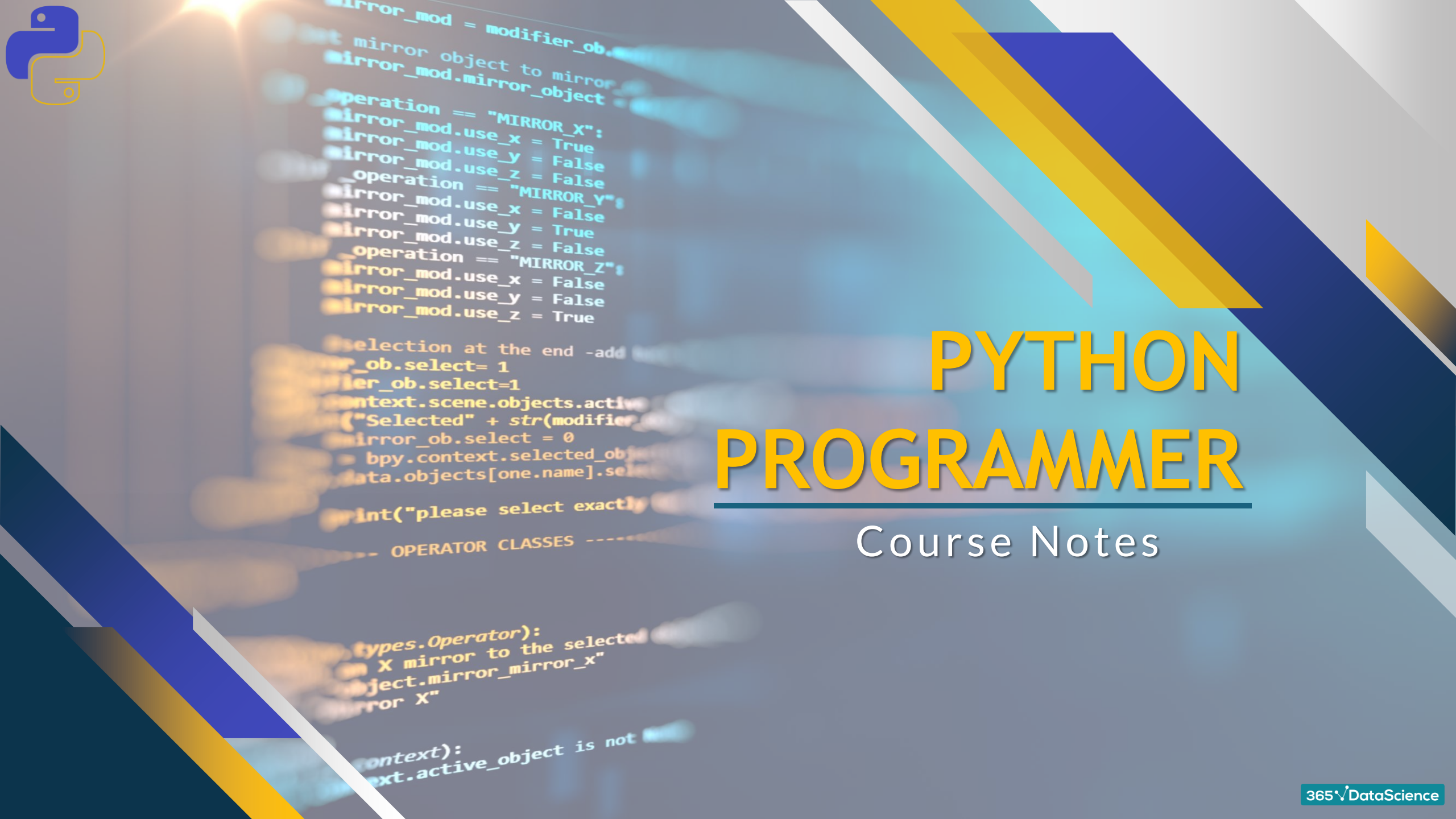


PYTHON PROGRAMMER

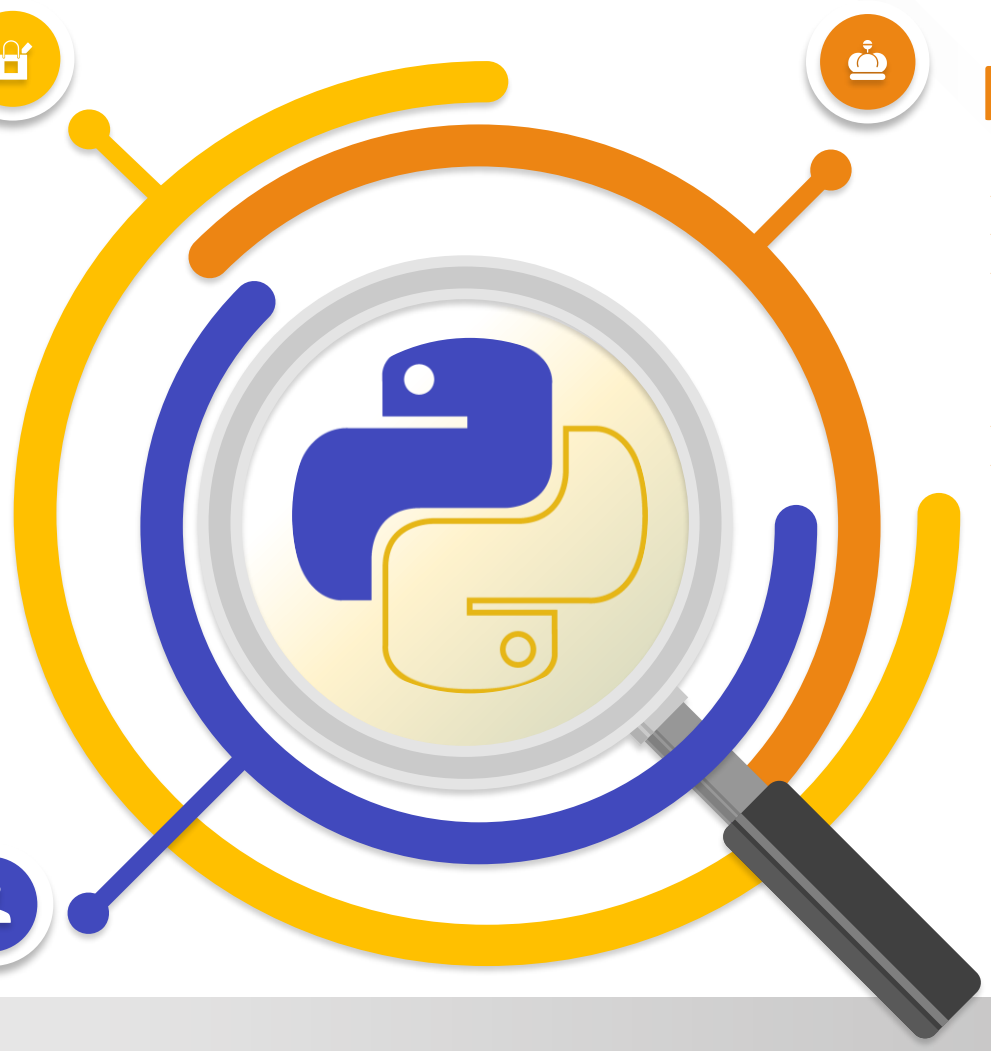
Course Notes

365 DataScience

Why Python?

Python is good for:

- Data Science
- Machine Learning
- Data Visualizations
- Data Applications
- Web Development



More benefits:

- A lot of job opportunities
- Large Python community
- Cross platform
- Completely Free
- A lot of built-in functionalities

Python is used in:

- Academia
- Industry

3 Steps Python Installation

Supported by a vibrant community of open-source contributors, Anaconda Distribution is the tool of choice for solo data scientists and machine learning enthusiast, who want to use Python for scientific computing projects.

Step 1

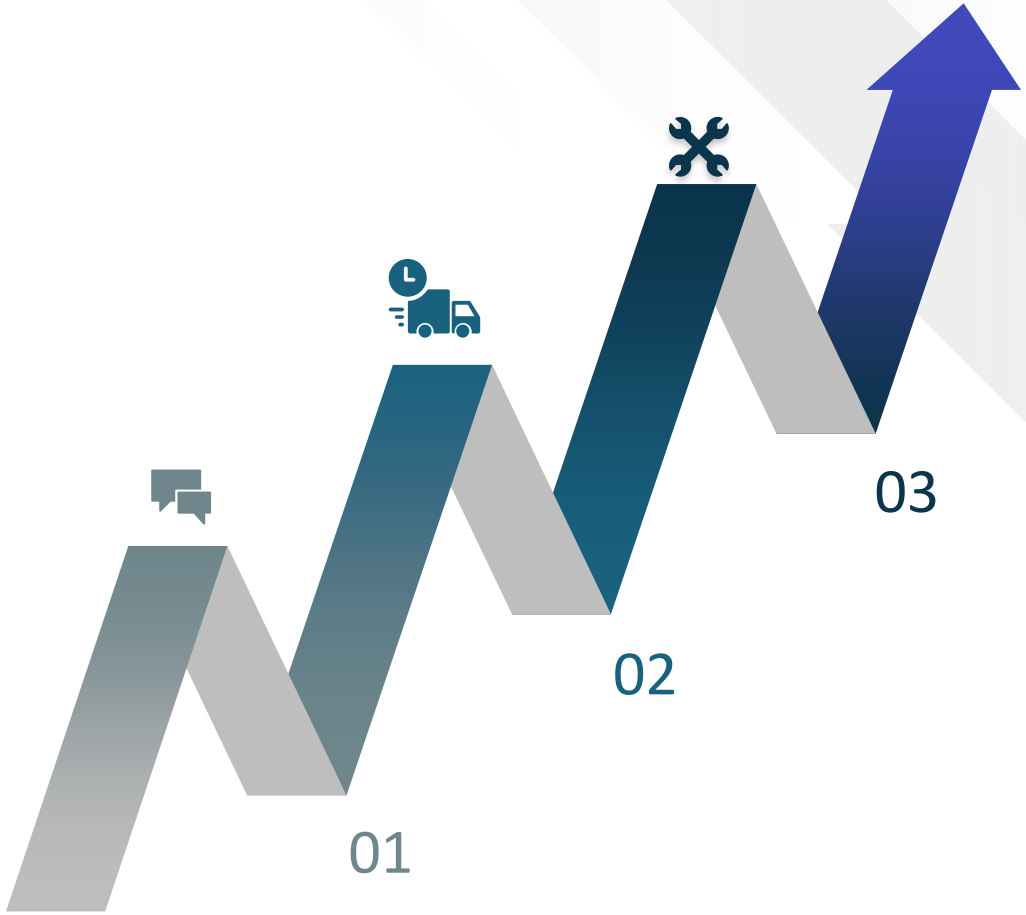
Go to www.anaconda.com

Step 2

Click on *Download*

Step 3

Choose Python 3 version



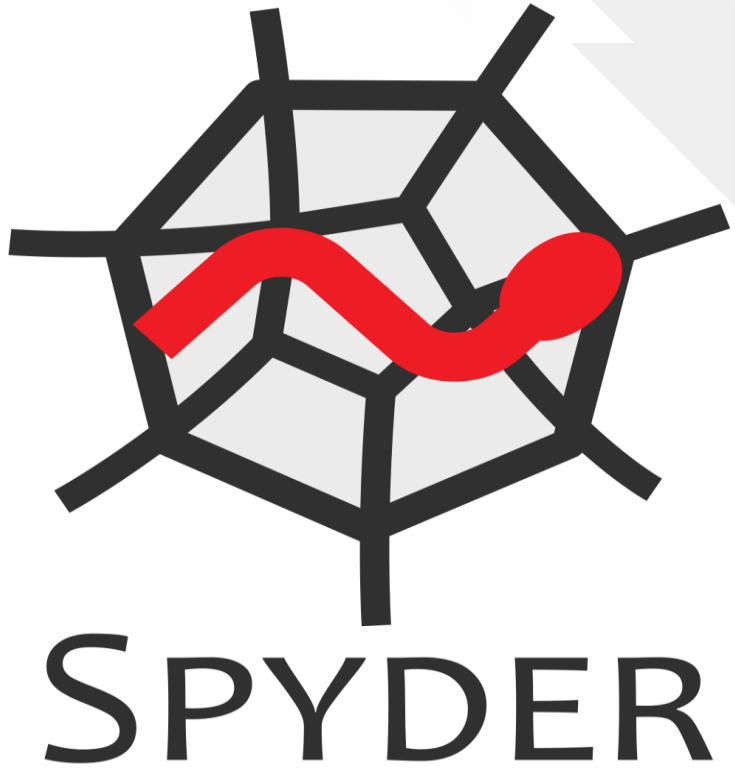
Introducing Spider

IDE

Integrated Development Environment
It has:
- Source code area
- Highlighted syntaxis
- A place for running code

Spyder

Contains:
- File editor
- IPython Console
- Variable explorer, File explorer and Help box



Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It offers a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

Beyond its many built-in features, its abilities can be extended even further via its plugin system and API. Furthermore, Spyder can also be used as a PyQt5 extension library, allowing developers to build upon its functionality and embed its components, such as the interactive console, in their own PyQt software

Printing in a Nutshell

In Jupyter and Spyder, the last line in the input field is returned as an output (if there is anything to display).
Therefore to display an operation we can either write as the last line in the input field or use the print function.

Python Syntax

Forward Slash
Always gives a decimal number called Floating Point Number (Float)

2 Forward Slashes
Always gives an integer value and discards the decimal part

2 Asterisks
To the power of

Single Quotation Marks
Creates a string

Clear
Clears the console.

Whenever we are dealing with the *print ( )* function, we can force the display of the operations

(Word)
Name error ⇢ always indicate the string of characters with single or double quotation marks

end=' '
Defines an end character for a string .
By default the end character is a new line.

If we are printing a string, we've got some some extra characters to help us

Backward Slash
Escapes the following character.

Backward Slash + t
Indents the text, Tab command

Backward Slash + n
New line

Variables: General Rules

Asking the computer for more space in the main memory

Variables can't start with number.

1x=3

01

We can't have space in a variable's name.

! However, we can have underscores.

new_variable = 3

02

We can change variable's values.

z = x + y

z = x - y

03

Giving the variables sensible names is a good practice.

area=pi*radius**2

04

A variable can be set equal to a string. (String concatenation)

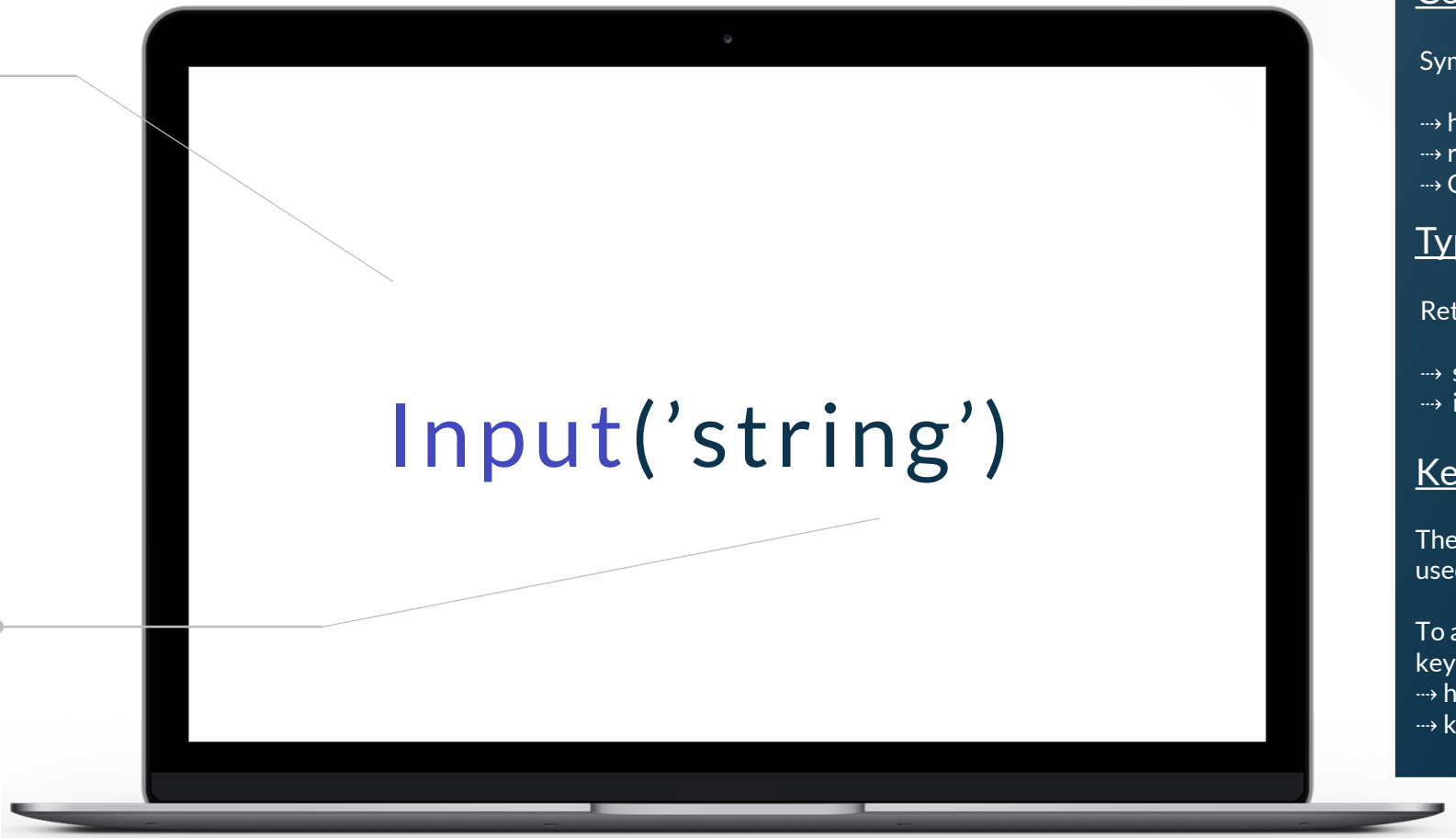
phrase_1 + phrase_2

05

Strings: Input Function

Allows you to get user's input

The user is requested to make an input



If we press "Enter" ⇢ The input function has finished running

NB:

Comment/Uncomment:

Symbol: #

⇢ highlight the area
⇢ right click
⇢ Comment/Uncomment

Type:

Returns the type of a value

⇢ str = string
⇢ int = integer

Keywords:

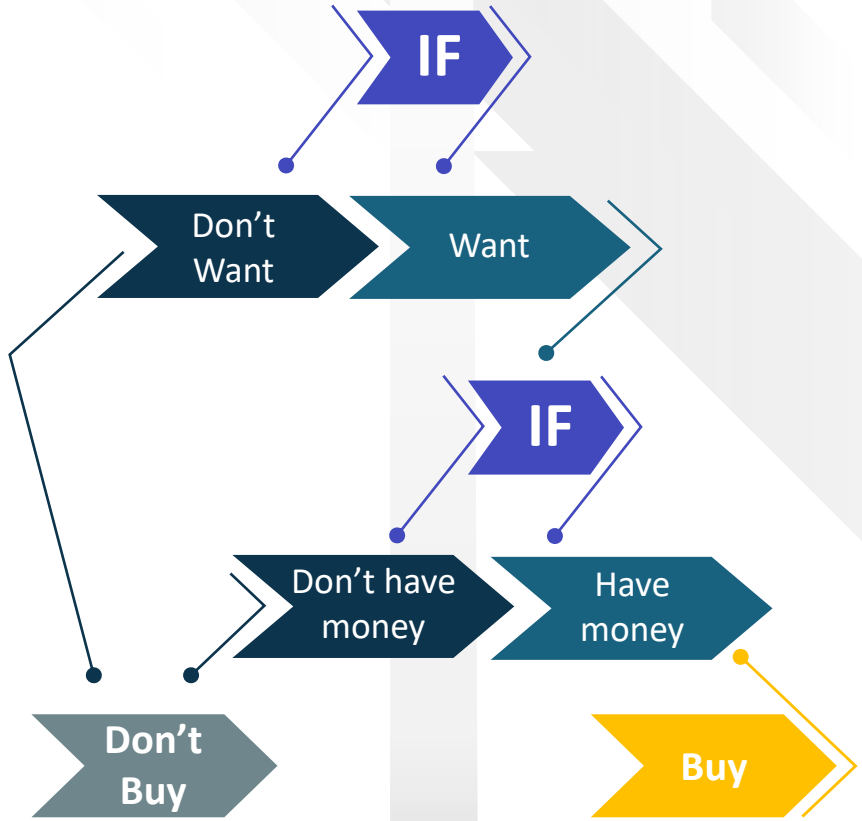
They are not allowed to be used as variables names.

To access the list with the keywords, type in the console:
⇢ help ()
⇢ keywords

Conditionals: Boolean Expressions

Conditionals are what allow computers to make decisions based on the value of variables.
Python can check weather a statement is true or false.

==	Checking for equality
!=	Not equal to
<=	Less than or equal to
>=	Grater than or equal to



Conditionals in real life Example

Logical operators

Logical operators chain Boolean expressions together to give a particular result

Or

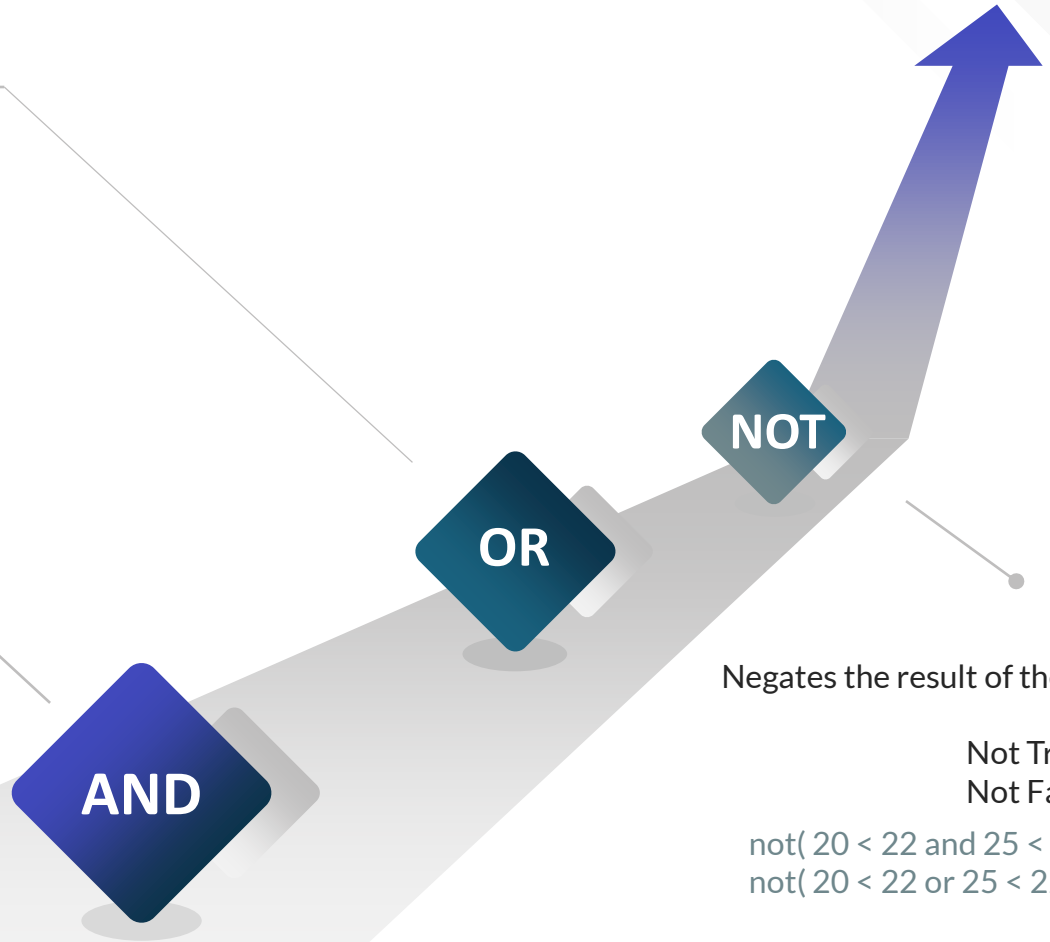
If only one variable satisfies the condition
⇢ True

Is 15 < 22 or 22 < 22 ⇢ True

And

If both conditions are satisfied
⇢ True

is 15 < 22 and 20 < 22 ⇢ True



Not

Negates the result of the condition

Not True ⇢ False
Not False ⇢ True

not(20 < 22 and 25 < 22) ⇢ True
not(20 < 22 or 25 < 22) ⇢ False

If Statement

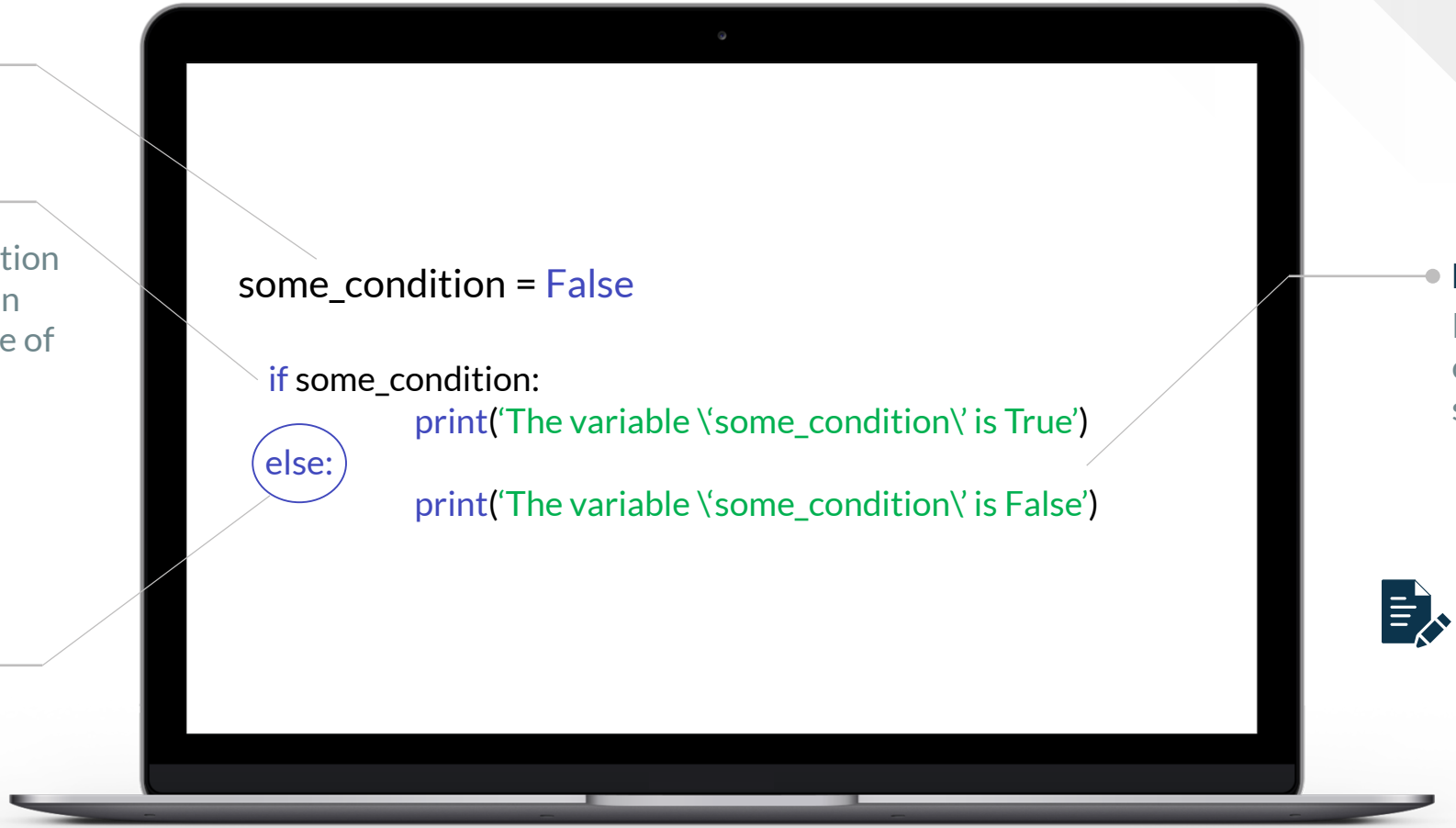
Python supports the usual logical conditions from mathematics.
These conditions can be used in several ways, most commonly in "if statements" and loops.

Our variable

If statement

Meaning: If this condition evaluates to True, then print the following line of code.

If, else and *print* are colored in blue, because they are keywords.



Else statement

Meaning: In every other case, print something else

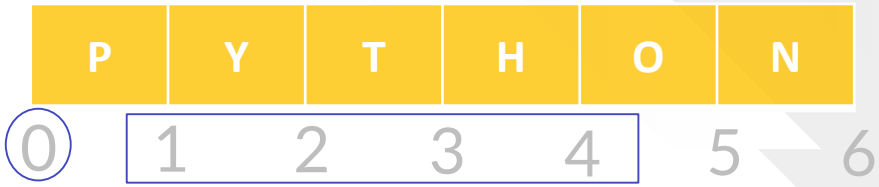
NB: You can use *elif* statement if you want to check more than one condition

More on strings

This is how string indexing from a coding perspective looks like.

Our variable: my_string

⇢ It points to a space in the main memory.
⇢ That space contains the string *Python*
⇢ Each of *Python*'s letters has its own unique space
⇢ We can access these letters by using the index numbers ⇢



- **len(my_string)**

  Shows the length of the string
  Python = 6 (It has 6 letters)

- **my_string[0]**

  Meaning: From this string, please give me the first character
  Python = P

- **my_string[1:4]**

  Meaning: Give me access to group of letters that start at the first index to the fourth
  Python = yth

- **my_string.upper ( )**

  Capitalizes all letters. Lower makes all letters lower case.

NB:
1. Python is a zero-indexing program language

This means that Python starts counting from 0 and that's why if we type [1] – Python will give us the letter Y.

2. In Python:
'n' == 'N'
False

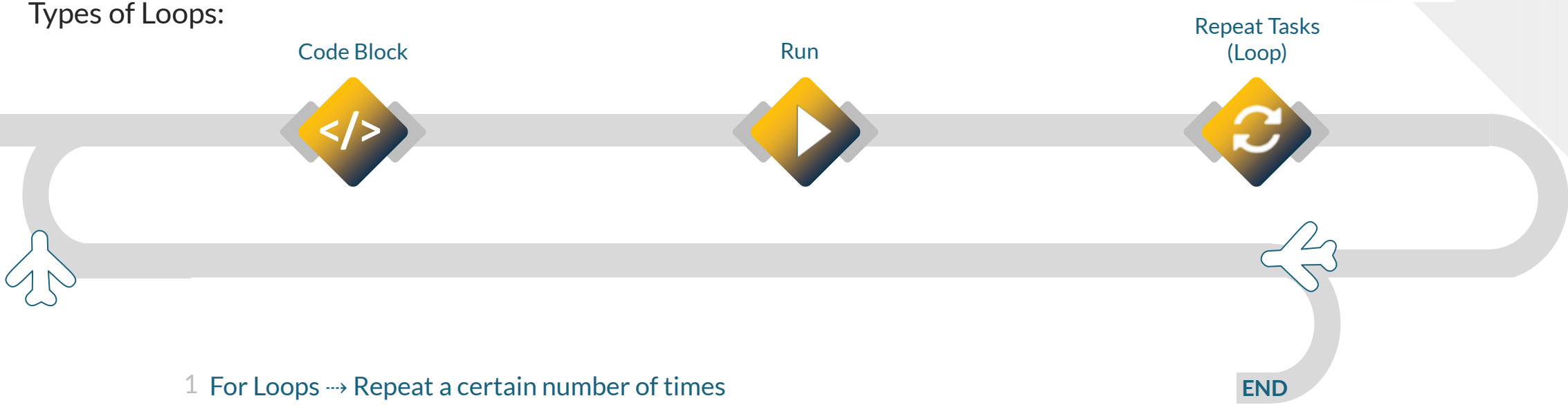
Lists and Loops

Computers are good at doing repetitive tasks quickly

List Example:

```
my_list = [ 34, 76, 58]
```

Types of Loops:



1. For Loops ⇢ Repeat a certain number of times
2. While Loops ⇢ Repeat based on a condition (True/False)

NB:
Incrementing values:
Changing the value of a variable

Common List Methods

Append

It appends the desired value at the end of the list

*It could be useful in For Loops

Remove

Removes values from the list

Reverse

It reverses the order of the list

Extend

Similar to Append. We can add a whole new list to the current list.

Index

This method shows the index of the value

```
my_list.
```

If you write the name of the variable (e.g. my_list) and then type . (dot) all of the methods that are available will appear.

Modolo Operator

It's denoted by the % sign

11 % 2

Result: 1

Modulus gives you the remainder of the result of the division