Book

Python for Data Science

Sourav Agarwal
Founder, Datahat

Simplified Data Science

Index

7. Object Oriented Programming approach (2 weeks)
    a. Defining class as a blueprint
    b. User defined classes
    c. Encapsulation
    d. Abstraction
    e. Inheritance
    f. Polymorphism
8. General purpose libraries in Python (1 week)
    a. Math
    b. Random
9. Python Libraries for Data Science (4 week)
    a. Numpy
    b. Pandas
    c. Matplotlib
    d. Sklearn
    e. Seaborn
    f. Scipy
10. Interview preparation guide (2 weeks)
11. Advanced modules (2 weeks)
    a. Productionizing using Python
        i. API generation
    b. Dashboarding
        i. Streamlit

Datahat Tip: Grow by 1% each day

Acknowledgement

I, would like to sincerely thank my family who helped me manage time for this book by taking care of my primary needs

I would like to thank my friends who motivated me into writing another tutorial book on Python

Reading Rainbow Tip: “It’s impossible, until it’s done”

Python Installation and Setup

What is Python??

Python is a high-level, general purpose Object Oriented Programming language that emphasizes code readability and dynamic typing

Prerequisites

1. Technical Requirements
    a. Computer (Laptop or Desktop): At least 4 GB RAM and 4GB Storage
    b. Internet Connectivity
2. Learning Plan
    a. Dedicated learning hours (at least 4 hours per week)
    b. Following the lectures regularly
    c. Sincerely attempting the assignments
    d. Discussing

Python IDLE

Step 1: Check if you have **Python pre-installed**

Windows: Open Powershell

MacOS or Linux: Open Terminal

```
c:\> python –version
```

```
$python –version
```

Note: If you are a Linux user (such as Ubuntu), you would have python pre installed

[If you have an updated Python version (Python 2), then install an upgraded version]

** To install an upgraded version of **Python in Linux**

Red Hat, CentOS, or Fedora

```
dnf install python3 python3-devel
```

Debian or Ubuntu

```
apt-get install python3 python3-dev
```

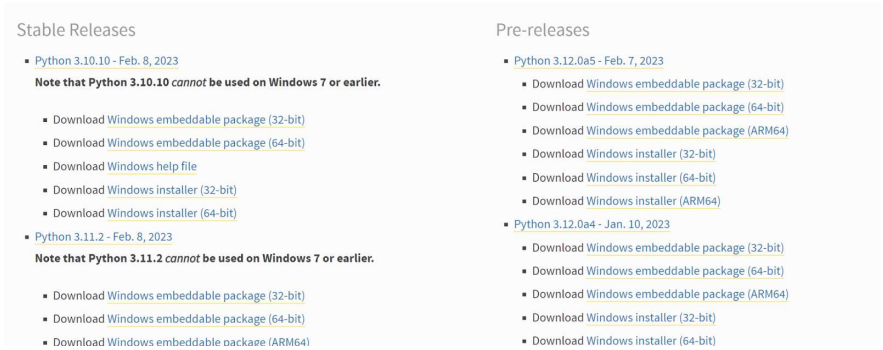
Gentoo

```
emerge dev-lang/python
```

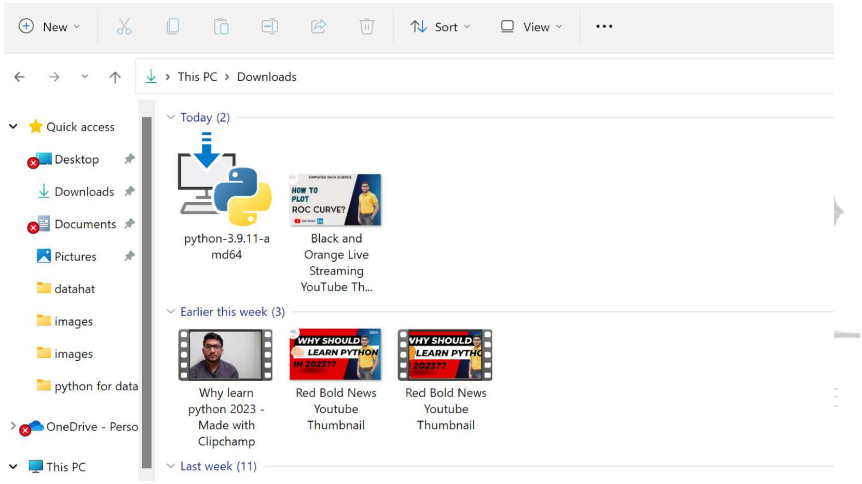
Arch Linux

```
pacman -S python3
```

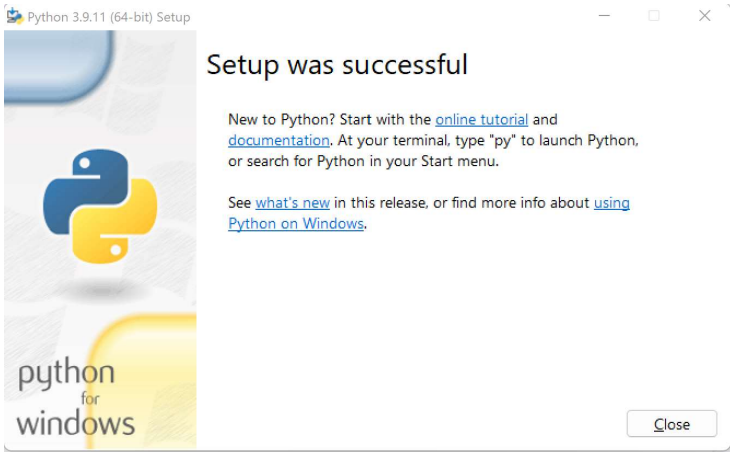
Step 2: visit the link and download the installer for Windows and MacOS



Step 3: execute or run the installer



Step 3: Finish the setup and check the Python in Windows Powershell or Terminal as in step 1



Step 4: Open IDLE: [**IDLE** is an Integrated and learning environment created with **Python]**

Similarly For MacOS

Use the installer for MacOS and follow the guide

For Linux

[If Python is not pre-installed]


```
$ sudo apt-get install python3.8
```

To install IDLE in Linux

```
$ sudo apt install idle              [On Debian/Ubuntu for Python2]
$ sudo apt-get install idle3         [On Debian/Ubuntu for Python3]
$ sudo yum install python3-tools     [On CentOS/RHEL and Fedora]
```

VS Code

Step 1: visit the link

Step 2: Follow the setup

Step 3: Setting up Python for VS Code

Step 4: Executing your first Python Program

[Online Python Compiler: https://www.programiz.com/python-programming/online-compiler/]

6. Basics of Programming

Chapter 1: Writing the First Program in Python

"Firsts Are Always Special"

I want all of you to pay close attention and follow along in your computer

We are going to answer 3 questions

- How to write a Python Program

first_program.py > ...

```
1   my_name = "Sourav Agarwal"
2   my_message = '''I am glad to be able to contribute back to the Python
3   Excited to be here talking to you and hope to help you in your Python
4
5   #display
6   print(my_name)
7   print(my_message)
8
9   #error  [Interpreter based during run time]
10  print("\nhello")
11  print("welcome to datahat")
```

- How to execute a python program

  $python <*filename.py*>

```
PS C:\Users\datahat\datahat2.0> python .\first_program.py
Sourav Agarwal
I am glad to be able to contribute back to the Python Community
Excited to be here talking to you and hope to help you in your Python Journey

hello
welcome to datahat
PS C:\Users\datahat\datahat2.0>
```

- How to Debug

Python is interpreted during Runtime

Debug using these 3 steps

```
PS C:\Users\datahat\datahat2.0> python .\first_program.py
Traceback (most recent call last):
  File "C:\Users\datahat\datahat2.0\first_program.py", line 6, in <module>
    prnt(my_name)
NameError: name 'prnt' is not defined
PS C:\Users\datahat\datahat2.0>
```

- Look for the line number in the error message (colored in Yellow: here line 6)
- Check for the error message (colored in pink)
- Go to the line in code and correct

Here line no 6 is erroneous

Chapter 2: Basic Components of Python Program (Scope | Indentation | Keywords | Identifier)

Now, let us look at the 4 primary components of a Python Program

- **Scope:**

  Scope identifies the region in which a particular variable/identifier is valid. For example, if variable “var1” is defined with a value of 4 and “var1” is assigned a value “hello” within the scope of a function func(),

  Then var1 would have a value “hello” within the scope of the function func(); used outside the scope of this function, it would have the earlier value of 4

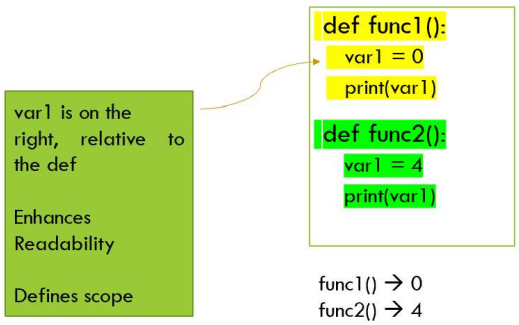
  Understanding from the analogy, let us say we have 2 files named “datahat” in our computer, now 1 of the files is present in directory1 while the other in directory2

  The values in the two files are different. So the data retrieved would depend on the scope or the region from which the file is accessed

  Scope came about because early programming languages (like BASIC) only had global names. With this kind of name, any part of the program could modify any variable at any time, so maintaining and debugging large programs could become a real nightmare. To work with global names, you’d need to keep all the code in mind at the same time to know what the value of a given name is at any time. This was an important side-effect of not having scopes.

- **Indentation**

SCOPE & INDENTATION



Defines the scope and enhances readability

- **Keywords**

Keywords are predefined and reserved words in any programming language including Python. We cannot use keywords s an identifier

Here are the list of keywords in Python 3

[**Python 3 has 33 keywords | Python2 had 30**]

```
List of all keywords in python 3
['False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert', 'async', 'await
', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally'
, 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not'
, 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
PS C:\Users\datahat\datahat2.0>
```

- **Identifier**

Identifiers are the names we give to identify a variable, function, class, module or objects

Rules for writing Identifiers

[Python is case sensitive]

1. Identifiers can be a combination of letters, digits and underscore (_) : var1, Var_, _12, _var_234

2. An identifier cannot start with a digit: 1var (invalid)
3. We cannot use special characters except _ : var$, var8_# (invalid)
4. Keywords are not allowed as identifiers : name, assert (invalid)

Assignment Chapter 1 & 2

- Modify the above Python program to print your name along with a custom message of your choice
- Select the invalid identifier (Multiple options could be invalid)
    - var_name
    - VarName
    - My_first_name123
    - _yourname_23
    - 123_age
- Select the valid identifiers (Multiple options could be valid)
    - hello_
    - name
    - name_
    - True

Chapter 3: Input/Output in Python

Accepting User Input and displaying Output

In Python 3, there is **input()**

Python3

```
# Taking input from the user
name = input("Enter your name: ")

# Output
print("Hello, " + name)
print(type(name))
```

Syntax: input(<prompt>)

- Taking integer input
    - Using the input function, type cast the value to integer

Python3

```
# Taking input from the user as integer
num = int(input("Enter a number: "))

add = num + 1

# Output
print(add)
```

- Multiple inputs
    - Using **map()** and **split()**

Python3

```
a, b, c = map(int, input("Enter the Numbers : ").split())
print("The Numbers are : ",end = " ")
print(a, b, c)
```

- Input for sequences (List, Tuple)

```
[2]  1 L = list(map(int, input("Enter list elements:").split()))
     2 S = set(map(int, input("Enter Set Elements: ").split()))
     3 T = tuple(map(float, input("Enter tuple elements: ").split()))
     4
     5 print(type(L), type(S), type(T))
     6
     7 print("List: ", L)
     8 print("Set: ", S)
     9 print("Tuple: ", T)
```

```
Enter list elements:5 3 60 10 -1 8
Enter Set Elements: 62 -0 92 8 1 11 14
Enter tuple elements: 7.0 -2 7.9
<class 'list'> <class 'set'> <class 'tuple'>
List:  [5, 3, 60, 10, -1, 8]
Set:  {0, 1, 8, 11, 14, 92, 62}
Tuple:  (7.0, -2.0, 7.9)
```

Output using **print()**

Syntax: **print(value(s), sep=' ', end='\n', file=file, flush=flush)**

Parameters

- **value(s):** any value(s) passed for output would be converted to string type
- **sep**: (optional) corresponds to the separator between the values passed for output. Default → ' '

```
[7]   1 print(L, S)
      2 print(L, S, sep="|")

      [5, 3, 60, 10, -1, 8] {0, 1, 8, 11, 14, 92, 62}
      [5, 3, 60, 10, -1, 8]|{0, 1, 8, 11, 14, 92, 62}
```

- end: (optional) Specify what to print at the end. Default → '\n'
- file: (optional) File object with a write method
- flush: (optional) A boolean, specifying if the output is flushed (True) or buffered (False). Default → False

Format Output

- Using String % operator

  print("Vishesh: %2d, Portal: %.2f" %(1,06.33)

  https://realpython.com/python-modulo-string-formatting/#use-the-modulo-operator-for-string-formatting-in-python

  Format function

  https://www.geeksforgeeks.org/python-string-format-method/

Chapter 4: Operators in Python

There are 7 types of operators in Python

- Arithmetic
- Comparison
- Assignment
- Logical
- Membership
- Identity
- Bitwise

Arithmetic operators are used for numerical computations

These include

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Comparison Operators are used for comparing between the operands

Whether an operand is greater than the other, equal to ,etc

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Assignment Operators are used to assign values to a variable

There are 2 categories of assignment operators➙ Direct assignment | Assignment with computation

= is the assignment operator

When combined with any arithmetic operator it becomes assignment with computation

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3

Logical operators denote AND, OR, NOT operations

[Not is an unary operator]

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Membership operators check the presence or absence in the variable

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Identity Operators

Used to determine if the operand is or is not another operand or value

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Bitwise operators

Operator	Name	Description	Example
&	AND	Sets each bit to 1 if both bits are 1	x & y
|	OR	Sets each bit to 1 if one of two bits is 1	x | y
^	XOR	Sets each bit to 1 if only one of two bits is 1	x ^ y
~	NOT	Inverts all the bits	~x
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off	x << 2
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off	x >> 2