

Date : 6 june 2024

Day : Thursday

BASICS OF PYTHON

✓ Introduction to Python Programming :

You will begin your journey into Python programming, which is one of the most popular programming languages in the field of data science. Python is known for its versatility and power, making it a suitable choice for various data science tasks, such as data cleaning, preprocessing, analysis, and modeling. Learning Python will provide you with access to a wide range of libraries and tools that are specifically designed for data science, which can make your work with data much more efficient and effective.

Python is a good choice for beginners in data science for several reasons.

- Python has a simple and easy-to-learn syntax that is designed to be readable and resembles the English language. This makes it easy for beginners to write code and understand what is happening in their code.
- Python has a vast array of libraries and tools that are specifically designed for data science, including NumPy, Pandas, Matplotlib, and Scikit-learn. These libraries are essential for data manipulation, visualization, and machine learning tasks.
- Python has a large open source community that provides support, tutorials, and documentation, making it easier for beginners to learn and use Python for data science tasks.
- Python is a flexible language that can be used for a wide range of data science tasks and can handle large datasets.

Overall, Python's simplicity, vast array of libraries, open source community, and flexibility make it an excellent choice for beginners in data science.

Where do we practice coding?

IDE

An Integrated Development Environment (IDE) is a software application that provides programmers with a comprehensive set of tools and features to write, debug, and test their code in a single interface. IDEs often include a text editor, a debugger, a compiler, and other features like autocomplete, syntax highlighting, and code formatting. IDEs are designed to make the coding process more efficient and productive by providing all the necessary tools and features in one place, which can save time and effort for developers. IDEs are commonly used for programming languages like Python, Java, C++, and many more.

Few examples of IDE:

- PyCharm
- Spyder
- IDLE
- Jupyter
- Visual Studio Code (VSCode):
- Google Colab

Numeric Data Types in Python :

We will be focusing on two main numeric data for the time being:

Integers: Integers (or ints) are whole numbers without decimal points. In Python, integers can be positive, negative, or zero. They can be created by writing a number without a decimal point or by using the `int()` function.

Floats: Floats are numbers with decimal points. In Python, floats can be positive, negative, or zero. They can be created by writing a number with a decimal point, by using scientific notation (e.g. `1.23e-4`), or by using the `float()` function.

Booleans: Boolean is a data type in Python that can have two values: True or False, used for logical operations.

Python also supports other numeric data types, such as complex numbers, `bool` for boolean values (True or False) and `decimal` for fixed-precision decimal values. Additionally, Python provides a rich set of operators and functions for performing arithmetic operations and other manipulations on numeric data.

Basic Arithmetic

Let's explore some basic arithmetic operations. Run the code cells to see the answers

```
#Hello World program  
print("Hello World")
```

 Hello World

```
#addition  
5+4
```

 9


```
#subtraction  
6-3
```

 3

```
#multiplication  
9*6
```

 54

```
#float division  
10/3
```

 3.3333333333333335

```
#float division  
25/5
```

 5.0

```
#floor division  
35//2
```

 17

```
#modulus  
33%2
```

 1

```
#exponential  
3**3
```

 27

```
(5+3)+5*4+2*3
```

 34

✓ BODMAS

operator precedence in python follows the **BODMAS** rule for arithmetic expressions. The precedence of operators is listed below in a high to low manner.

Firstly, parantheses will be evaluated, then exponentiation and so on.

B – Bracket

O – Order

D – Division

M – Multiplication

A – Addition

S – Subtraction

BODMAS nostalgia (BODMAS :- Bracket, Order, Division, Multiplication, Addition, Substraction)

```
(5 + 2) * 3 ** 2 - 1
```

```
=73 * 2 - 1
```

```
=7*9-1
```

```
=63-1
```

```
=62
```

Activity 1 - Score Board Operator

An operator working behind the scoreboard of a inter cohort AlmaBetter cricket tournament, is responsible for updating the scores and points of each team. However, the operator is currently facing a challenge. He has been tasked with updating the total number of points gained by Team London, but he does not possess the necessary programming skills to complete this task. According to the tournament's rules, teams are awarded the following points based on the outcome of a match:

wins: 3 points

draws: 1 point

losses: 0 points

Team London has played 9 matches in this tournament. They won 6 matches, lost 2 matches and drew 1. The operator is in need of assistance to calculate the total number of points earned by Team London. As a python expert adept with knowledge of integer, floats and boolean, you can help the operator by writing a solution for the following problem.

What would your approach be?

```
#1. Define your fixed values:
wins = 3
draws = 1
losses = 0
#2. Calculate the total points gained by Team London
london_points = 6 * wins + 2*losses + 1*draws
# 3. Print the variable london_points
print(london_points)
```

 19

```
ait = 4*losses+1*draws+7*wins
print(ait)
```

 22

✓ What is a Variable?

VARIABLES are entities which help us store information and retrieve it later.

A variable with a fixed name can store information of nature like numeric, textual, boolean etc.

A Python variable is a reserved memory location to store values. In other words, a variable in a python program gives data to the computer for processing.

The type of data contained in a variable can be changed at user's will.

```
y=6.4
```

```
y
```

 6.4

✓ Basic Arithmetic operations

```
# Addition
x=6
y=7
z = x+y
```

```

print(z)

# Multiplication
m = x*y

print(m) # Print the variable z
type(m) # Get the data type of variable z

# Division
d = x/y

print(d) # Print the variable z
type(d) # Get the data type of variable z

# Subtraction
z = x-y

# Use the in-built print function to print the variable
print(z)

```

Waittt! Shouldn't it be 0.75?? Well, The reason we get this result is because we are using "floor" division. The // operator (two forward slashes) is the mathematical equivalent of doing `[0.75]` which returns the greatest integer less than or equal to 0.75

```

# Modulo operator
y=5
x=3
z = y%x
print(z)



# Using powers and exponents
z = x**y # We did not even need to store it in another variable nor use print command
print(z)

```

Rules for naming a variable in Python

- Variables names must start with a letter or an underscore like `_product` , `product_`
- The remainder of your variable name may consist of letters, numbers and underscores.
- `spacy1`, `pyThon`, `machine_learning` are some valid variable names
- Names are case sensitive.
- `case_sensitive`, `CASE_SENSITIVE`, and `Case_Sensitive` are each a different variable.
- It's a best practice to create a variable that should be coherent with the context you are working on. Let's say you are working with fruit data; the variable name should be `"fruit"` or `"fruit_mango."`

Introduction to Booleans

Booleans, or true  /false  values, are an essential part of programming, and Python is no exception!

In real life, we often make decisions based on whether something is true or false. For example, if it's raining outside, we might decide to take an umbrella. In programming, we use Booleans to make decisions too! We can write code that checks if something is true or false and then does something based on the answer.

For instance, think about a login page on a website. When a user enters their username and password, the program needs to check whether the information is correct or not. If it's correct, the program will let the user log in, and if it's incorrect, the program will deny access. This is just one example of how Booleans are used in real-world applications.

Comparison Operators

- These operators will allow us to compare variables and output a Boolean value (True or False).
- If you have any sort of background in Math, these operators should be very straight forward.

- First we'll present a table of the comparison operators and then work through some examples:
- In the table below, Assume a=3 and b=4.

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

✓ PRACTICE QUESTIONS

You're creating a program to manage a zoo's animal population. Declare a variable `lion_population` with an initial value of 10. The zoo welcomes 5 new lion cubs. Update the `lion_population` variable and print the total lion population.

```
lion_population = 10
new_cubs = 5
updated_population = lion_population + new_cubs
print(updated_population)
```

➞ 15

You're developing a weather monitoring system. Declare a variable `temperature` with a value of 25.5 degrees Celsius. Due to a sudden heatwave, the temperature increases by 8 degrees. Update and print the new temperature.

```
temperature = 25.5
increase = 8
new_temperature = temperature + increase
print(new_temperature)
```

➞ 33.5

A science experiment involves tracking the growth of a plant. Declare a variable `plant_height` with an initial value of 15 centimeters. Over a week, the plant grows 2.5 centimeters taller each day. Update and print the final height of the plant after the week.

```
plant_height = 15
increaseperday = 2.5
final_height = plant_height + 2.5*7
final_height = plant_height + 2.5*7
print(final_height)
```

➞ 32.5

You're designing a space mission trajectory. Declare variables `initial_velocity` and `acceleration` with values 3000 meters per second and 500 meters per second squared respectively. Calculate and print the final velocity after 10 seconds.

```
initial_velocity = 3000
acceleration = 500
time = 10
final_velocity = initial_velocity + acceleration*time
print(final_velocity)
```

➞ 8000


A group of friends is sharing a pizza. Declare a variable `pizza_slices` with a value of 8. Each friend wants to have an equal number of slices, and there are 5 friends. Calculate and print the maximum number of slices each friend can have without cutting the pizza.

```
pizza_slices = 8
friends = 5
8//5
```

➞ 1

You're modeling the movement of a pendulum. Declare a variable `pendulum_length` with a value of 1.2 meters. Calculate and print the period of oscillation (time taken for one complete swing) using the formula ($T = 2\pi \sqrt{\frac{L}{g}}$), where (π) is pi (approximately 3.14159) and (g) is the acceleration due to gravity (approximately (9.81) meters per second squared).

```
pendulum_length = 1.2
pi = 3.14159
g = 9.81
x = 2*pi
y = (pendulum_length/g)**0.5
time = x*y
print(time)
```

 2.197534089590265

A software company is tracking the number of bugs in their codebase. Declare a variable `bug_count` with an initial value of 100. After a round of debugging, 35 bugs are fixed. Update and print the new `bug_count`.

```
bug_count = 100
fixed = 35
new = 100-35
print(new)
```

 65

You're building a game where players collect gems. Declare a variable `gem_count` with an initial value of 50. Each time a player finds a gem, 5 gems are added to their collection. Update and print the new `gem_count`.

```
gem_count = 50
add_gems = 5
new_count = (gem_count + add_gems)*5
print (new_count)
```

 275


A fitness tracker is monitoring a user's heart rate variability (HRV). Declare a variable `hrv_index` with an initial value of 80. After a relaxation session, the user's HRV improves by 10 points. Update and print the new `hrv_index`.

```
hrv_index = 80
improvement = 10
new_hrv = hrv_index + 10
print (new_hrv)
```

 90

You're simulating the growth of a bacterial colony. Declare a variable `bacteria_count` with an initial value of 5000. Over a day, the colony doubles in size every 4 hours. Update and print the new `bacteria_count`.

```
bacteria_count = 5000
hours = 24/4
updated_count = (bacteria_count**2)*6
print (updated_count)
```

 150000000

Start coding or [generate](#) with AI.

