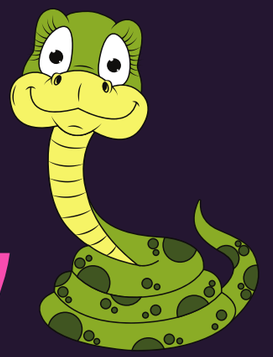
acm
HIT Chapter

Learn.py



Python Decoded.....

Co-ordinator of the program : Prof. Dr. Soumen Paul

Manasija Bhattacharya

What is Python?

Python is a widely used general-purpose, high-level programming language. It was initially designed by **Guido van Rossum** in 1991 and developed by Python Software Foundation. Its code looks similar to plain English, which helps developers quickly understand and write it. Python also has a large community and many libraries, which are like pre-built tools that make coding faster and easier. Whether you're building a website, analyzing data, or creating a game, Python has something for everyone!

Why choose Python?

1 **Powerful and Versatile**

Python is a general-purpose programming language that is widely used for web development, data analysis, artificial intelligence, and more.

2 **Easy to Learn**

With its clean and readable syntax, Python is an excellent choice for beginners to start their coding journey.

3 **Cross-Platform Compatible**

Python code can run on a variety of operating systems, including Windows, macOS, and Linux.

Input: *input()*

The input() function is used to take input from the user. When called, it displays a prompt message (optional) and waits for the user to type something and press Enter. The entered data is always treated as a string (text), but you can convert it to other types like numbers if needed.

Output: *print()*

The print() function displays information on the screen. It can print strings, numbers, variables, or even complex data like lists. You can also use print() to display multiple items by separating them with commas. We can use f-strings too (later).

```
name = input("Enter your name: ")
print("Hello, " + name + “.")
```

Variables

Learn how to declare and assign values to variables in Python, following naming conventions and best practices.

```
name = "Alice"
age = 25
dead = True
print(name)
```

Data Types

Explore the various data types in Python, including integers, floats, strings, and Boolean, and how to work with them.

int: whole numbers
(ex: 5, -10)
float: decimal numbers
(ex: 3.14, -2.5)
String : text
(ex: "hello")
Boolean: True or false

Operations

Discover the different arithmetic, assignment, and comparison operators available in Python and how to use them effectively.

Arithmetic:
+, -, *, /, %
Comparison:
==, !=, >, <
Logical:
and, or, not

CONTROL STATEMENTS

❑ If - Else:

In Python, conditional statements let the code make decisions. The main ones are if, elif, and else.

- If: checks a condition. If it's true, the code inside runs.
- Elif: checks another condition if the first is false.
- Else: runs if none of the above conditions are true.

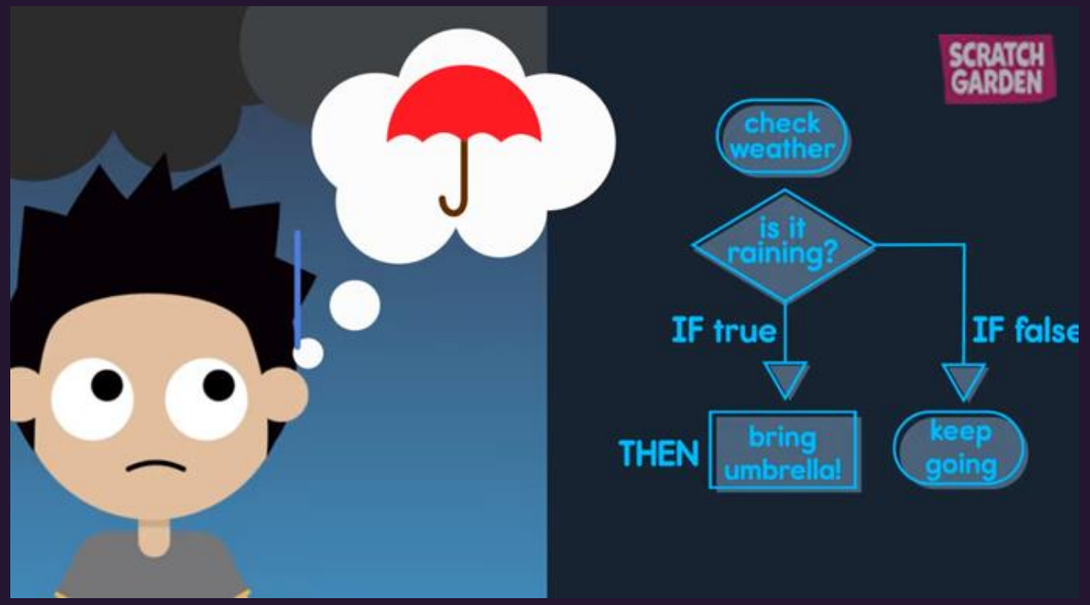
```
age = 17

if age >= 18:
    print("You can vote.")
elif age >= 16:
    print("Almost there!")
else:
    print("Too young to vote.")
```

❑ Ternary:

Short version of if – else.

```
age = 17
can_vote = "You can vote." if age >= 18 else "Too young to vote."
```



LOOPS

Loops in Python allow you to repeat a block of code multiple times, making it easy to handle tasks that require repetition.

Before

```
print(1)
print(2)
print(3)
print(4)
print(5)
```

After

```
for i in range(1, 6):
    print(i)
```

code

```
1  a = 1
2  while a < 10:
3   print (a)
4   a += 2
```

output

variables

Loops:

For loop

```
for i in range(5):
 print(i)
```

While loop

```
i = 0
while i < 5:
    print(i)
    i += 1
```

Do while loop

```
i = 0
while True:
print(i)
i += 1
if i >= 5:
break
```

For each loop

```
names = ["Alice", "Bob", "Charlie"]

for name in names:
    print(name)
```

Data Structures:

List: []

In Python, a **list** is a built-in data structure that is used to store an ordered collection of items.
Lists are mutable, meaning that their contents can be changed after the list has been created.

```
fruits = ["apple", "banana", "cherry"]
print(fruits)
fruits[1] = "blueberry“
print(fruits)
```

Tuple: ()

An ordered, immutable (unchangeable) collection of items. Tuples also allow duplicates and are defined with parentheses ()

```
colors = ("red", "green", "blue")
print(colors)
```

Set: {}

An unordered collection of unique items. Sets don’t allow duplicates and are defined with curly braces {}

```
num = {1, 2, 2, 3, 4, 5}
print(num)
```

Dictionary: {}

An unordered, mutable (changeable) collection of key-value pairs in Python. Each key must be unique, and it maps to a specific value. Dictionaries are defined with curly braces {} and colons : to separate keys and values.

```
person = { "name": "Alice", "age": 25,
"city": "New York" }
print(person)
```

Function:

Function is a reusable block of code that performs a specific task. Functions help organize code, making it easier to read, maintain, and reuse.

Types of Functions:

- **Built-in Functions:** Functions that are already available in Python, such as print(), len(), and type().

```
print(len("Hello")
```

- **User-defined Functions:** Functions that you create to perform specific tasks using def.

```
def greet(name):
    return f"Hello, {name}!”

print(greet(“Prantik"))
```

Python Libraries and Frameworks

1 **Standard Library**

Python standard library is simply great.

Ex: Math, Random, Datetime

2 **Popular Libraries**

Explore widely-used libraries like NumPy, Pandas, and Matplotlib for data analysis and visualization.

3 **Web Frameworks**

Learn about web development frameworks such as Django and Flask, which simplify the process of building web applications.

4 **Data Science Frameworks**

Understand how to use TensorFlow and Scikit-learn for machine learning and artificial intelligence projects.

Object-Oriented Programming (OOP) in Python

Classes

A blueprint for creating objects. It defines a set of attributes and methods that the objects created from it will have.

Objects

An instance of a class. It’s created from a class and can have data (attributes) and behaviors (methods).

Inheritance

A mechanism where a new class (child) derives properties and methods from an existing class (parent), allowing for code reuse.

Encapsulation

A way to restrict access to certain parts of an object, typically by making attributes private. This ensures that data can only be accessed and modified through specific methods.

```python
class Car:
    def __init__(self, brand, color):
        self.brand = brand
        self.color = color

    def start(self):
        return f"{self.brand} car is starting."

my_car = Car("Toyota", "Red")

print(f"My car is a {my_car.color} {my_car.brand}.")
print(my_car.start())
```

Conclusion and Next Steps

Congratulations! You have completed the "Learn.py" Python workshop. With the knowledge and skills you've gained, you're now equipped to tackle more advanced Python projects and explore the many applications of this versatile programming language.

Remember, the journey of learning never ends. Continue to practice, experiment, and explore the wealth of Python resources available online and in the community. Happy coding!