1. Printing to the Console:

Python

```
print("Hello, World!")
```

This will print “Hello, World!” to the console.

2. Comments:

Python

```
# This is a single-line comment

"""
This is a multi-line comment.
It can span multiple lines.
"""
```

Comments are used to add explanations or notes to the code. They are not executed and are purely for human readability.

3. Indentation:

Python uses indentation (whitespace) to define blocks of code. This is different from many other languages that use curly braces `{}`. Indentation is crucial for Python and helps maintain readability.

Python

```
if True:
    print("This is indented")
```

4. Variables and Data Types:

Python is dynamically typed, meaning you don't need to specify the data type of a variable explicitly.

Python

```
# Integer
num = 10

# Float
pi = 3.14

# String
name = "Alice"

# Boolean
is_valid = True
```

5. String Concatenation:

Python

```
first_name = "Sandip"
```

```
last_name = "Gupta"
full_name = first_name + " " + last_name
```

6. Getting User Input:

Python

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

7. Conditional Statements:

Python

```
if condition:
    # code if condition is True
elif another_condition:
    # code if another_condition is True
else:
    # code if none of the conditions are True
```

8. Loops:

a. For Loop:

Python

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

b. While Loop:

Python

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

9. Lists:

Python

```
my_list = [1, 2, 3, 4, 5]
```

10. Functions:

Python

```
def greet(name):
    print("Hello,", name)
```

11. Classes and Objects:

Python

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print(f"Hello, my name is {self.name} and I'm {self.age} years
old.")
```

12. Modules and Imports:

Python

```
import math

result = math.sqrt(25)
```

13. Exception Handling:

Python

```
try:
    # code that might raise an exception
except ExceptionType as e:
```

```
    # code to handle the exception
```

14. Working with Files:

Python

```
with open('myfile.txt', 'w') as file:
    file.write('Hello, World!')
```

These are some of the basic elements of Python syntax. They form the foundation upon which you can build more complex programs.

Related posts:

1. Download Python
2. How to run a Python Program
3. Python program to find GCD of two numbers
4. Python Program to find the square root of a number by Newton’s Method
5. Python program to find the exponentiation of a number
6. Python Program to find the maximum from a list of numbers
7. Python Program to perform Linear Search
8. Python Program to perform binary search
9. Python Program to perform selection sort
10. Python Program to perform insertion sort
11. Python program to find first n prime numbers
12. Python program Merge sort

13. NumPy
14. Python library
15. Python Installation and setup
16. Python Variables
17. Python Data Types
18. Python lists
19. Python Creating and Accessing List
20. Python List Manipulation
21. Python Input function
22. Python list slicing
23. Python Class and Object
24. Python find the output programs
25. Python Introduction
26. Python int data type
27. Python float data type
28. Understanding Floating-Point Precision in Python: Avoiding Numerical Computation Errors
29. How to search Python library using command line tool
30. Which python libraries are used to load the dataset ?
31. Why is there no need to mark an int float in a variable in Python ?
32. Does Python have double, short long data types
33. What are High-Level Programming Languages?
34. What are Interpreted Programming Languages?
35. What are General-Purpose Programming Languages?
36. What is a variable in Python?