Coding in Python

First line of code in python

The first line of code to print the string in inverted commans using the print function in python.

```
print("Hello to the world of Python!")
```

Data Types in Python

Python has several built-in data types. Here are a few of them:

- **Integers**:These are whole numbers, either positive, negative, or zero.
- **Floats**:These are decimal numbers.
- **Strings**:These are sequences of characters, such as words or phrases.
- **Boolean**:This is a logical value that can be either True or False.
- **List**:This is a collection of items that can be any data type,including strings,integers,floats,and other lists.

I. Integers

```
x_1 = 5
print(x_1)
```

II. Floats

```
x_2 = 3.14
print(x_2)
```

III. Strings

```
x_3 = "Hello, World!"
print(x_3)
```

IV. Boolean

```
x_4 = True
print(x_4)
```

V. List

```
x_5 = [1, 2, 3, 4, 5]
print(x_5)
```

Operators in Python

1. Arithmetic Operators

Arithmetic operators are used to perform mathematical operations. Here are a few of them:

- **Addition**: `a + b`
- **Subtraction**: `a - b`
- **Multiplication**: `a * b`
- **Division**: `a / b`
- **Modulus**: `a % b` (returns the remainder of the division of a by b)
- **Exponentiation**: `a ** b` (returns a to the power of b)

I. Addition

```
add = 5 + 3
print(add)
```

II. Substraction

```
subtract = 10 - 4
print(subtract)
```

III. Multiplication

```
multiply = 7 * 2
print(multiply)
```

IV. Division

```
divide = 9 / 3
print(divide)
```

V. Modulus

```
modulus = 17 % 5
print(modulus)
```

VI. Exponentiation

```
exponentiation = 2 ** 3
print(exponentiation)
```

2. Comparison Operators

Comparison operators are used to compare values. Here are a few of them:

- **Equal**: `a == b`
- **Not Equal**: `a != b`
- **Greater Than**: `a > b`
- **Less Than**: `a < b`
- **Greater Than or Equal**: `a >= b`
- **Less Than or Equal**: `a <= b`

I. Equal

```
equal = 5 == 5
print(equal)
```

II. Not Equal

```
not_equal = 5 != 3
print(not_equal)
```

III. Greater Than

```
greater_than = 7 > 3
print(greater_than)
```

IV. Less Than

```
less_than = 3 < 7
print(less_than)
```

V. Greater Than or Equal

```
greater_than_or_equal = 7 >= 7
print(greater_than_or_equal)
```

VI. Less Than or Equal

```
less_than_or_equal = 7 <= 7
print(less_than_or_equal)
```

Logical Operators

Logical operators are used to combine conditional statements. Here are a few of them:

- **And**: `a and b` (returns True if both a and b are True)
- **Or**: `a or b` (returns True if either a or b is True)
- **Not**: `not a` (returns True if a is False)

I. And Operator

```
operator_1 = True and True
print(operator_1)
```

II. Or Operator

```
operator_2 = True or False
print(operator_2)
```

III. Not Operator

```
operator_3 = not True
print(operator_3)
```

Assignment Operators

Assignment operators are used to assign values to variables. Here are a few of them:

- **Assign**: `a = b` (assigns the value of b to a)
- **Add and Assign**: `a += b` (adds b to a and assigns the result to a)

- **Subtract and Assign**: `a -= b` (subtracts b from a and assigns the result to a)
- **Multiply and Assign**: `a *= b` (multiplies a by b and assigns the result to a)
- **Divide and Assign**: `a /= b` (divides a by b and assigns the result to a)

I. Assign Operator

```
a = 5
print(a)
```

II. Add and Assign Operator

```
a += 3
print(a)
```

III. Subtract and Assign Operator

```
a -= 2
print(a)
```

IV. Multiply and Assign Operator

```
a *= 2
print(a)
```

V. Divide and Assign Operator

```
a /= 2
print(a)
```

Membership Operators

Membership operators are used to check if a value is present in a sequence. Here are a few of them:

- **In**: `a in b` (returns True if a is present in b)
- **Not In**: `a not in b` (returns True if a is not present in b)

I. In Operator

```
fruits = ['apple', 'banana', 'cherry']
print('apple' in fruits)
```

II. Not In Operator

```
fruits = ['apple', 'banana', 'cherry']
print('grape' not in fruits)
```

Identity Operators

Identity operators are used to check if two variables are the same object. Here are a few of them

- **Is**: `a is b` (returns True if a and b are the same object
- **Is Not**: `a is not b` (returns True if a and b are not the same object)

I. Is Operator

```
a = [1, 2, 3]
b = [1, 2, 3]
print(a is b)
```

II. Is Not Operator

```
a = [1, 2, 3]
b = [1, 2, 3]
print(a is not b)
```

Shift Operators

Shift operators are used to shift the binary representation of numbers. Here are a few of them:

- **Left Shift**: `a << b` (shifts the binary representation of a to the left by b places)
- **Right Shift**: `a >> b` (shifts the binary representation of a to the right by b places)

I. Left Shift Operator

```
a = 10
print(a << 2)
```

II. Right Shift Operator

```
a = 10
print(a >> 2)
```

Variabels in Python

A variable is a name given to a value.It is a way to store and refer to a value in a program.Different type of variable in python are:

- **Integer**:A whole number, like 1, 2, 3, etc.
- **Float**:A decimal number, like 3.14 or -0.5.
- **String**:A sequence of characters, like 'hello' or "hello".
- **Boolean**:A value that is either True or False.
- **List**:A collection of values that can be of any data type, including strings, integers,floats, and other lists.
- **Tuple**:A collection of values that can be of any data type,including strings, integers,floats,and other tuples.
- **Dictionary**:A collection of key-value pairs, where each key is unique and maps to a specific value.
- **Set**: A collection of unique values.

I. Integer Variable

An integer is a whole number that can be positive, negative, or zero, without any fractional or decimal components.

```
a = 10
print(a)
```

II. Float Variable

A float is a number that has a fractional or decimal component.

```
a = 10.5
print(a)
```

III. String Variable

A string is a sequence of characters, such as a word or phrase, enclosed in quotes.

```
a = 'hello'
print(a)
```

IV. Boolean Variable

A boolean is a value that can be either True or False.

```
a = True
print(a)
```

V. List Variable

A list is a collection of values that can be of any data type, including strings, integers, floats, and other lists.

```
a = [1, 2, 3, 'hello', 4.5]
print(a)
```

VI. Tuple Variable

A tuple is a collection of values that can be of any data type, including strings, integers, floats, and other tuples. Tuples are immutable, meaning they cannot be changed after they ar created.

```
a = (1, 2, 3, 'hello', 4.5)
print(a)
```

VII. Dictionary Variable

A dictionary is a collection of key-value pairs, where each key is unique and maps to a specific value.

```
a = {'name': 'John', 'age': 30, 'city': 'New York}
print(a)
```

VIII. Set Variable

A set is an unordered collection of unique elements.

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

Loops in Python

A loop is a control structure that allows you to execute a block of code repeatedly for a specified number of times. There are two types of loops in Python:

- for loop
- while loop

I. For Loop

A for loop is used to iterate over a sequence (such as a list,tuple, or string) and execute a block of code for each item in the sequence.

```
fruits = ['apple', 'banana', 'cherry']
for fruit in fruits:
    print(fruit)
```

II. While Loop

A while loop is used to execute a block of code repeatedly while a certain condition is true.

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

While Loop with a brake

A while loop can be used with a break statement to exit the loop when a certain condition is met.

```
i = 0
while i < 5:
    print(i)
    if i == 3:
        break
    i += 1
```

For Loop with a condition

A for loop can be used with a condition to iterate over a sequence and execute a block of cod

```
fruits = ['apple', 'banana', 'cherry']
for i, fruit in enumerate(fruits):
    if fruit == 'banana':
        print(fruit)
```

Conditional Statements in Python

Conditional statements are used to execute different blocks of code based on certain conditions.There are three types of conditional statements in Python:

- if statement
- elif statement
- else statement

I. If Statement

An if statement is used to execute a block of code if a certain condition is true.

```
x = 5
if x > 10:
    print("x is greater than 10")
```

II. Elif Statement

An elif statement is used to check another condition if the first condition is false.

```
x = 5
if x > 10:
    print("x is greater than 10")
    elif x == 5:
        print("x is equal to 5")
```

III. Else Statement

An else statement is used to execute a block of code if all the conditions in the if and elif statements are false.

```
x = 5
if x > 10:
    print("x is greater than 10")
    else:
        print("x is less than or equal to 10")
```

Accessing Data Elements in Python

In Python, you can access data elements in a list, tuple, or dictionary using their index or key.

```
my_list = [1, 2, 3, 4, 5]
print(my_list[0])
```

```
food=["pizza","pasta","burger","noodles","rice"]
# now we simply relocate the new value to the valudes in the dataset
food[1]="briyani"
```

```
# Access the elements from dictionary
person={"name":"khawar","age":25,"country":"pakistan"}
print(person)
print(person["name"])
print(person["country"])
```

Nested Loops in Python

A nested loop is a loop inside another loop. The inner loop will execute for each iteration of the outer loop.

```
# nested While loop
x=0
y=0

while x<5:
    y=0
    while y<5:
        print(x,y)
        y=y+1
    x=x+1

# nusted for loop
x=0
y=0

for x in range(5):
    for y in range(5):
        print(x,y)
```

Function in Python

A function in Python is a method that can be called multiple times from different parts of the code.

- Functions are used to group a set of instructions together to perform a specific task.
- Functions can take arguments and return values.
- Functions can be reused in the code.
- Functions can be used to make the code more readable and maintainable.

I. Defining a Function

```
# defining a function
def greet(name):
    print("Hello, " + name + "!")
    greet("John")
```

II. Function Arguments

```
# function arguments
def greet(name, age):
    print("Hello, " + name + "! You are " + str(age) + " years
    greet("John", 30)
```

III. Function Return Values

```
# function return values
def greet(name):
    return "Hello, " + name + "!"
    print(greet("John"))
```

IV. Function with Multiple Return Values

```
# function with multiple return values
def greet(name, age):
    return "Hello, " + name + "!", age
    print(greet("John", 30))
```

V. Function with Default Argument Values

```
# function with default argument values
def greet(name, age=30):
    return "Hello, " + name + "! You are " + str(age) + " years
    print(greet("John"))
```

VI. Function with Variable Number of Arguments

```
# function with variable number of arguments
def greet(*names):
    for name in names:
        print("Hello, " + name + "!")
        greet("John", "Jane", "Bob")
```

Input Function in Python

An input function in Python is a built-in function (`input()`) that reads user input from the console as a string, optionally displaying a prompt to guide the user.

```
# input function
name = input("Please enter your name: ")
print("Hello, " + name + "!")
```

```python
# input function with default value
name = input("Please enter your name: ")
if name == "":
    name = "John"
    print("Hello, " + name + "!")
```

```python
age_1=25            # khawar age
age_2=30            # zeeshan age

if age_1>age_2:
    print(name_1,"is older")
else:
    print(name_2,"is older")
```