Python_Notes_Week_03

Type Casting: Type casting, or type conversion, refers to converting a variable from one data type to another in Python. For example, if you have a variable containing a string number, such as "27", you may need to convert it to an integer before performing arithmetic operations. Otherwise, Python will interpret "27" as a string and add it to other strings instead of performing arithmetic.

```
a =   2       # Integer
b = " 27 "    # String
c = int (b)
sum = a + c
print (sum)
```

Types of Type Casting in Python

https://github.com/ibraheem-02/Python_Programs/blob/main/Type_Casting.ipynb

Python provides two types of type casting:

1. **Explicit Type Casting**: Where the programmer manually converts one data type into another.
2. **Implicit Type Casting**: Where Python automatically converts one data type to another to prevent data loss or errors.

Explicit Type Casting: Explicit conversion requires you to use built-in Python functions to convert a value from one type to another manually. When you specify explicit type casting, you have full control over the data type you want.

Here's an example of explicit type casting, where both a and b are converted from strings to integers before being added:

```
a = "2"     #String

b = "20"    #String
print(int(a)+ int(b))
```

In this example, a and b are explicitly converted into integers using the int() function, making the addition work as expected.

The explicit type casting is performed as per the requirement and avoids type mismatches in Python.

Implicit Type Casting: In implicit type casting, Python handles the conversion of data types automatically. This process usually occurs when different types need to be used together in an expression. Python converts the lower precision type to a higher precision type to avoid data loss.

For instance, if you add an integer to a float, Python will automatically convert the integer to a float before performing the addition:

```
a = 3.57
b = 3
sum = a + b
print(sum)
```

In this example, Python automatically converts d from an integer to a float to match c. This process is called implicit type casting and helps ensure that operations run smoothly without requiring manual intervention.

Input from User: The input () function allows user input.

Syntax input(*prompt*)

prompt: A String, representing a default message before the input.

https://colab.research.google.com/drive/1AgnWL5QXtiLUtiPv9Uxc89seYhWrZ06t#scrollTo=OPmxzOBg1dQo

Write a program Take input two numbers from user and print the average of two number.

```
N1 = int (input("Enter first No: "))
N2 = int (input("Enter second No: "))
Avg = (N1 + N2)/2
print("Average of two number is: ", Avg)
```

write a program No 1 is >= No 2 without using else if if N1 is >= to N2 then return True or otherwise false.

```
N1 = int (input("Enter first No: "))
N2 = int (input("Enter second No: "))
print(N1 >= N2)
```

Strings: Strings in python are surrounded by either single quotation marks, or double quotation marks.

https://github.com/ibraheem-02/Python_Programs/blob/main/String_Functions.ipynb

'hello' is the same as "hello".

we can display a string literal with the print() function:

Assigning a string to a variable is done with the variable name followed by an equal sign and the string. we can assign a multiline string to a variable by using three quotes:

```
str = "I am Muhammad Ibrahim.\nStudent of IMCS at UOS."
print(str)

# Using upper trophy
str = '''I'm Muhammad Ibrahim.\nStudent of "IMCS" at UOS.'''
print(str)
```

String Functions

```
# concatination of two string
str1 = "Muhammad Ibrahim "
str2 = "I am student of IMCS"
str = str1 + str2
print(str)

# lenth of string
print("Length of string is: ",len(str))

# Indexing of string
# accesing chracter
ch = str [9]
print("Character 9 is ",ch)

# to captalize first letter
print(str1.capitalize()) # Store in new string creat itself
print("i Capitalized",str1)

# replace string letters
print("IMCS is Replaced with IMCS at UOS")
print(str.replace("IMCS","IMCS at UOS"))

# find word it returns index of first accurence
print("Letter I is at Index ",str.find("I"))

# if we search for letter that is not present it returns -1 coz its not
valid index
print("Letter not found ",str.find("z"))

# to count the letter how many time letter accur
print("t accurs ",str.count("t"))
```

Control Structure: Flow of execution of program.

https://github.com/ibraheem-02/Python_Programs/blob/main/Control_Structures.ipynb

Sequential Flow: Normal flow (linear flow) every statement execute one by one (line by line)
Example:
print ("This is the first line.")
print ("This is the second line.")
print ("This is the third line.")

Selection Flow: Selection flow is a programming concept where different sections of code are executed based on conditions. It allows programs to make decisions, which is fundamental in creating dynamic and interactive applications. In Python, selection flow is implemented using if, elif, and else statements. If (condition) => Boolean Result.

Single AlterNet

Syntax:
if(condition):
statement

```
# single alter net
n1 = int(input("Enter a no: "))
if n1 >= 0:
    print("Entered no is positive")
```

Double AlterNet

if(condition):
statement
else:
statement

```
#Double alternet
n1 = int(input("Enter a no: "))
if n1 >= 0:
    print("Entered no is positive")
else:
    print("Entered no is negative")
```

Multiple AlterNet

if(condition):
statement
elif:
statement
else:
Statement

```
print("Character should be (A,a,B,b,C,c)")
char = input("Enter Single Chracter: ")
if char == "A" or char == "a":
    print("Apple")
elif char == "B" or char == "b":
    print("Banana")
elif char == "C" or char == "c":
    print("Cherry")
else:
        print("Invalid Character")
```