**AL-2001**

**Programming for AI**

**Lab # 2**

**Basics of Python, Variables, Data Types, Conditional Structure**

Note: Basics of python Variables has been covered in last lab.

**Data Types in Python**

* **Built-in Data Types:**

In programming, data type is an important concept. Variables can store data of different types, and different types can do different things.

**Python data types** are actually classes, and the defined variables are their instances or objects. Since Python is dynamically typed, the data type of a variable is determined at runtime based on the assigned value. In general, the data types are used to define the type of a variable. It represents the type of data we are going to store in a variable and determines what operations can be done on it.

Each programming language has its own classification of data items. With these data types, we can store different types of data values.

Python has the following data types built-in by default, in these categories:

|  |  |
| --- | --- |
| Text Type: | Str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | Dict |
| Set Types: | set, frozenset |
| Boolean Type: | Bool |
| Binary Types: | bytes, bytearray, memoryview |
| None Type: | NoneType |

**Example:**

# integer variable.

a=100

print("The type of variable having value", a, " is ", type(a))

# float variable.

c=20.345

print("The type of variable having value", c, " is ", type(c))

# complex variable.

d=10+3j

print("The type of variable having value", d, " is ", type(d))

## Python String Data Type:

A string in Python is an object of str class. It can be verified with type() function.

A string is a non-numeric data type. Obviously, we cannot perform arithmetic operations on it. However, operations such as **slicing** and **concatenation** can be done. Python's str class defines a number of useful methods for string processing. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python string is a sequence of one or more Unicode characters, enclosed in single, double or triple quotation marks (also called inverted commas). Python strings are immutable which means when you perform an operation on strings, you always produce a new string object of the same type, rather than modifying an existing string.

As long as the same sequence of characters is enclosed, single or double or triple quotes don't matter. The plus (+) sign is the string concatenation operator and the asterisk (\*) is the repetition operator in Python.

**Example:**

str = 'Hello World!'

print (str) # Prints complete string

print (str[0]) # Prints first character of the string

print (str[2:5]) # Prints characters starting from 3rd to 5th

print (str[2:]) # Prints string starting from 3rd character

print (str \* 2) # Prints string two times

print (str + "TEST") # Prints concatenated string

## Python Sequence Data Types:

## Sequence is a collection data type. It is an ordered collection of items. Items in the sequence have a positional index starting with 0. It is conceptually similar to an array in C or C++. There are following three sequence data types defined in Python.

* List Data Type
* Tuple Data Type
* Range Data Type

### **Python List Data Type:**

Python Lists are the most versatile compound data types. A Python list contains items separated by commas and enclosed within square brackets ([]). To some extent, Python **lists are similar to arrays** in C. One difference between them is that all the items belonging to a Python list can be of different data type whereas C array can store elements related to a particular data type.

**Example:**

list = [ 'abcd', 786 , 2.23, 'Ali', 76.2 ]

tinylist = [123, 'umar']

print (list) # Prints complete list

print (list[0]) # Prints first element of the list

print (list[1:3]) # Prints elements starting from 2nd till 3rd

print (list[2:]) # Prints elements starting from 3rd element

print (tinylist \* 2) # Prints list two times

print (list + tinylist) # Prints concatenated lists

**Python Tuple Data Type:**

Python tuple is another sequence data type that is similar to a list. A Python tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses (...).

A tuple is also a sequence, hence each item in the tuple has an index referring to its position in the collection. The index starts from 0.

**Example:**

tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )

tinytuple = (123, 'john')

print (tuple) # Prints the complete tuple

print (tuple[0]) # Prints first element of the tuple

print (tuple[1:3]) # Prints elements of the tuple starting from 2nd till 3rd

print (tuple[2:]) # Prints elements of the tuple starting from 3rd element

print (tinytuple \* 2) # Prints the contents of the tuple twice

print (tuple + tinytuple) # Prints concatenated tuples

**Python Range Data Type:**

A Python range is an immutable sequence of numbers which is typically used to iterate through a specific number of items. It is represented by the Range class. The constructor of this class accepts a sequence of numbers starting from 0 and increments to 1 until it reaches a specified number. Following is the syntax of the function −

**range(start, stop, step)**

Here is the description of the parameters used −

**start:** Integer number to specify starting position, (Its optional, Default: 0)

**stop:** Integer number to specify ending position (It's mandatory)

**step:** Integer number to specify increment, (Its optional, Default: 1)

**Examples:**

1. for i in range(5):

print(i)

1. for i in range(2, 5):

print(i)

1. for i in range(1, 5, 2):

print(i)

**Python Dictionary Data Type:**

Python dictionaries are kind of hash table type. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Python dictionary is like associative arrays or hashes found in Perl and consist of key: value pairs. The pairs are separated by comma and put inside curly brackets {}. To establish mapping between key and value, the semicolon':' symbol is put between the two.

>>> {1:'one', 2:'two', 3:'three'}

In Python, dictionary is an object of the built-in dict class. We can check it with the type() function.

>>> type({1:'one', 2:'two', 3:'three'})

<class 'dict'>

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

**Example** of Dictionary Data Type

dict = {}

dict['one'] = "This is one"

dict[2] = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}

print (dict['one']) # Prints value for 'one' key

print (dict[2]) # Prints value for 2 key

print (tinydict) # Prints complete dictionary

print (tinydict.keys()) # Prints all the keys

print (tinydict.values()) # Prints all the values

**Python Boolean Data Type:**

Python boolean type is one of built-in data types which represents one of the two values either True or False. Python bool() function allows you to evaluate the value of any expression and returns either True or False based on the expression.

A Boolean number has only two possible values, as represented by the keywords, True and False. They correspond to integer 1 and 0 respectively.

>>> type (True)

<class 'bool'>

>>> type(False)

<class 'bool'>

Example of Boolean Data Type

Following is a program which prints the value of boolean variables a and b −

Open Compiler

a = True

# display the value of a

print(a)

# display the data type of a

print(type(a))

This will produce the following result −

true

<class 'bool'>

**Example:**

# Returns false as a is not equal to b

a = 2

b = 4

print(bool(a==b))

# Following also prints the same

print(a==b)

# Returns False as a is None

a = None

print(bool(a))

# Returns false as a is an empty sequence

a = ()

print(bool(a))

# Returns false as a is 0

a = 0.0

print(bool(a))

# Returns false as a is 10

a = 10

print(bool(a))

**Python None Type:**

Python's none type is represented by the "nonetype."  It is an object of its own data type. The nonetype represents the null type of values or absence of a value.

**Example of None Type**

In the following example, we are assigning None to a variable x and printing its type, which will be nonetype –

# Declaring a variable

# And, assigning a Null value (None)

x = None

# Printing its value and type

print("x = ", x)

print("type of x = ", type(x))

**Python Conditional Structure**

**Python Conditions and If statements:**

Python supports the usual logical conditions from mathematics:

* Equals: a == b
* Not Equals: a != b
* Less than: a < b
* Less than or equal to: a <= b
* Greater than: a > b
* Greater than or equal to: a >= b

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

**Example of If statement:**

a = 33;  
b = 200;  
if b > a:  
  print("b is greater than a")

In this example we use two variables, a and b, which are used as part of the if statement to test whether b is greater than a. As a is 33, and b is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

## Indentation:

## Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

**Example:**

If statement, without indentation (will raise an error):

a = 33;  
b = 200;  
if b > a:  
print("b is greater than a") # you will get an error

## Elif:

The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".

**Example:**

a = 33;  
b = 33;  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")  
In this example a is equal to b, so the first condition is not true, but the elif condition is true, so we print to screen that "a and b are equal".

## Else:

The else keyword catches anything which isn't caught by the preceding conditions.

**Example:**

a = 200;  
b = 33;  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")  
else:  
  print("a is greater than b")

In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that "a is greater than b".

You can also have an else without the elif:

**Example:**

a = 200;  
b = 33;  
if b > a:  
  print("b is greater than a")

else:  
  print("b is not greater than a")

## Short Hand If:

If you have only one statement to execute, you can put it on the same line as the if statement.

**Example:**

One line if statement:

if a > b: print("a is greater than b")

**Short Hand If ... Else:**

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

**Example:**

One line if else statement:

a = 2;  
b = 330;  
print("A") if a > b else print("B")

This technique is known as **Ternary Operators**, or **Conditional Expressions**.

You can also have multiple else statements on the same line:

**Example:**

One line if else statement, with 3 conditions:

a = 330;  
b = 330;  
print("A") if a > b else print("=") if a == b else print("B")

**And:**

The and keyword is a logical operator, and is used to combine conditional statements:

**Example:**

Test if a is greater than b, AND if c is greater than a:

a = 200;  
b = 33;  
c = 500  
if a > b and c > a:  
  print("Both conditions are True")

**OR:**

The or keyword is a logical operator, and is used to combine conditional statements:

**Example:**

Test if a is greater than b, OR if a is greater than c:

a = 200;  
b = 33;  
c = 500;  
if a > b or a > c:  
  print("At least one of the conditions is True")

**Not:**

The not keyword is a logical operator, and is used to reverse the result of the conditional statement:

**Example:**

Test if a is NOT greater than b:

a = 33;  
b = 200;  
if not a > b:  
  print("a is NOT greater than b")

**Nested If:**

You can have if statements inside if statements, this is called nested if statements.

**Example:**

x = 41;  
if x > 10:  
  print("Above ten,")  
  if x > 20:  
    print("and also above 20!")  
  else:  
    print("but not above 20.")

**The pass Statement:**

if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

**Example:**

a = 33;  
b = 200;  
  
if b > a:  
  pass

**Code Example (Taking input, variable, print):**

# Take input from the user

user\_name = input("Enter your name: ")

# Use a variable to store the input

showname = "Hello, " + user\_name + "!"

# Print the output

print(showname)

**Code to print a table via input:**

# Take input from the user

number = int(input("Enter a number to print its table: "))

# Print the multiplication table

print(f"Multiplication table for {number} is:")

for i in range(1, 11):

print(f"{number} x {i} = {number \* i}")

**LAB 2**

**Objectives**

• **Practice on Python Basics, Variables, Conditional Structure**

**Note: Carefully read the following instructions (***Each instruction contains a weightage***)**

* First think about statement problems and then write your logic on Copy / Notebook.
* Write **Your Name** and **Roll No** on your Paper/Sheet’s first page.
* Do not copy from any source otherwise you will be penalized with negative marks.
* Complete your lab **within given Time Slot**.
* Paste all your codes along with screenshots in a word file and renamed with your roll number.
* Keep all your source files in your computer for verification. Do not overwrite a single source file for all programs.

**[Problems/Task]**

* Find the area of a Rectangle. **(Area = Length \* Width)**
* Write a program to calculate the Sum, average & percentage of student marks based on Math, Science, English, and Urdu. **(Percentage = (Obtained / Total) \* 100)**
* Calculate the Interest of a Bank Deposit **(Formula "Interest=Amount\*Years\*Rate/100)**
* Convert Temperature from Fahrenheit **(℉) to Celsius (℃) (Formula C=5/9\*(F-32))**
* Compute the perimeter of a rectangle. **(Perimeter = 2 \* (Length \* Width))**
* A brand offered 10% discount on each shirt purchased. The original price of the shirt is 550, find the discounted price of one shirt. Write the code for this problem statement.
* Write a program logic that calculates the total of a retail sale. The program should ask the user for the following: the retail price of the item being purchased and the sales tax rate. Once the information has been entered the program should calculate and display the following: the sales tax for the purchase and the total sale.
* Write a program that calculates the current balance in a savings account. The program should obtain from the user the following information: the starting balance, the total amount of deposits made, and the total amount of withdrawals made. After the program has calculated the

current balance, it should be displayed on the screen. Assume one input for deposits and one input for withdrawals.

* Write a code to read the age of a candidate and determine whether it is eligible for casting his/her own vote or not. **Eligibility (age >= 18)**
* [Write a program to input electricity unit charges and calculate total electricity bill according to the given condition:](https://codeforwin.org/2015/05/c-program-to-calculate-electricity-bill.html)  
  For first 50 units Rs. 0.50/unit  
  For next 100 units Rs. 0.75/unit  
  For next 100 units Rs. 1.20/unit  
  For unit above 250 Rs. 1.50/unit  
  An additional surcharge of 20% is added to the bill
* Draw a flow diagram for a program that calculates the area of a shape. It asks you enter number of sides of the shape then asks you to enter the lengths. It is restricted to three major shapes. If user enters ‘0’ as number of sides then you enter ‘radius’ length and program print area of the circle. If user enters ‘3’ then program prints the area of the triangle. If user enters ‘4’ then program prints the area of the rectangle.
* Write a program to check whether a given number is a perfect number or not.

Test Data:

Input the number: 56

**Expected Output:**

The positive divisor: 1 2 4 7 8 14 28

The sum of the divisor is: 64

So, the number is not perfect.

* Write a program to find the perfect numbers within a given number of ranges.

**Test Data:**

Input the starting range or number: 1

Input the ending range of number: 50

**Expected Output:**

The Perfect numbers within the given range: 6 28

**You need to done with your exercise within given time.**