WEEK-2

Basics I/O operations Input- input (), raw_input() ; output – print (), formatting output. Datatypes Scalar

type: Numeric (int, long, float, complex), Boolean, bytes, None; Type casting Operators Arithmetic,

Comparison/Relational, Logical/Boolean, Bitwise; string operators; Expressions and operator precedence.

Basics I/O operations:

Python provides us with two inbuilt functions to read the input from the keyboard.

- input( prompt )
- raw_input( prompt )

☞ **input():**

- In Python, input() function is used to gather data from theuser.
- Syntax `variable_name = input([prompt])`

where, prompt is a string written inside the parenthesis thatis printed on the screen.

- The prompt statement gives an indication to the user of the value that needs to be entered through the keyboard.
- When the user presses Enter key, the program resumes and input returns what the user typed as a string.
- Even when the user inputs a number, it is treated as a stringwhich should be casted or converted to number explicitly using appropriate type casting function.
- Example:

1) person = input("What is your name?") # What is your name? Laxmi
2) age=int(input(“How old you are??)) # How old you are??36

☞ **raw_ input():** This function works in older versions of python and is similar to input(). (like Python 2.x).

Ex: name = raw_input("Enter your name : ")
print(name)

**Output: To display output on the screen using print()**

- The print() function allows a program to display text onto the console(screen).
- The print function will print everything as strings and anything thatis not already a string is automatically converted to its stringrepresentation.

**Syntax: print(object(s),sep=separator,end=end,file=file,flush=False)**

| Parameter | Description |
|---|---|
| object(s) | Any object, and as many as you like. Will be converted to string before printed |
| sep='separator' | Optional. Specify how to separate the objects, if there is more than one. Default is ' ' |
| end='end' | Optional. Specify what to print at the end. Default is '\n' (line feed) |
| File | Optional. An object with a write method. Default is sys.stdout |
| Flush | Optional. A Boolean, specifying if the output is flushed (True) or buffered (False). Default is False |

Examples:

```
1)print("Hello", "how are you?")
2)print("Hello", "how are you?", sep="---")
```

- Two major string formats which are used inside the print() function to display the contents onto the console as they are less error proneand results in cleaner code. They are

1) str.format()
2) f-strings

1.str.format() Method

- str.format() method is used when you need to insert thevalue of a variable, expression or an object into anotherstring and display it to the user as a single string.
- The format() method returns a new string with insertedvalues.
- The format() method works for all releases of Python3.x
- The format() method uses its arguments to substitute an appropriate value for each format code in the template.

- Example:

```
country = input("Which country do you live in?")
print("I live in {0}".format(country))
```

**1) Program to Demonstrate the Positional Change of Indexes of Arguments**

```
a = 10
b = 20
print("The values of a is {0} and b is {1}".format(a, b))
print("The values of b is {1} and a is {0}".format(a, b))
```

**2. f-strings ( Formatted Strings):**

- Formatted strings or f-strings were introduced in Python 3.6
- A f-string is a string literal that is prefixed with “f”.
- These strings may contain replacement fields, which are expressions enclosed within curly braces {}.
- The expressions are replaced with their values.
- Example:

```
country = input("Which country do you live in?")
print(f"I live in {country}")
```

**PROGRAM 1: Given the Radius, Write Python Program to Find the Area and Circumference of a Circle**

```
radius = int(input("Enter the radius of a circle"))
area_of_a_circle = 3.1415 * radius * radius
circumference_of_a_circle = 2 * 3.1415 * radius
print(f"Area={area_of_a_circle}andCircumference={circumference_of_a_circle}")
```

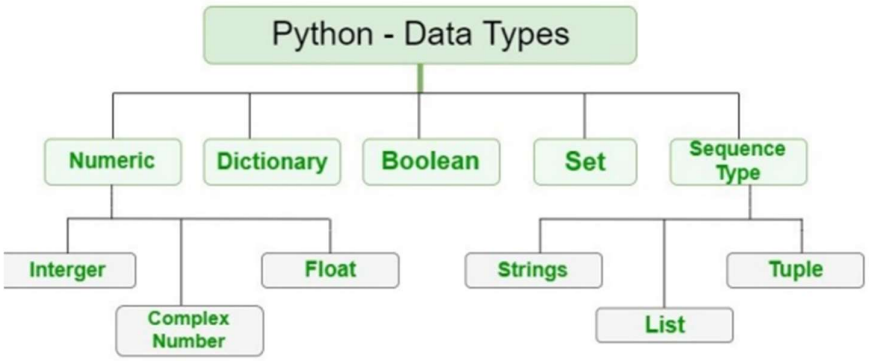
**PROGRAM 2: Write Python Program to Convert the Given Number of Days to a Measure of Time Given in Years, Weeks and Days. For Example,375 Days Is Equal to 1 Year, 1 Week and 3 Days (Ignore Leap Year).**

```
number_of_days = int(input("Enter number of days"))

number_of_years = int(number_of_days/365)

number_of_weeks = int(number_of_days % 365 / 7)
remaining_number_of_days = int(number_of_days % 365 % 7)

print(f"Years = {number_of_years}, Weeks = {number_of_weeks}, Days
={remaining_number_of_days}")
```

Data Types:

- Data types tell the type of data that the variable is holding.
- Data types are the classification or categorization of input values.
- Python variables can store data of different types.
- Python has the following built-in data types.
- Following are the standard or built-in data type of Python:

1) Numeric
2) Sequence Type
3) Boolean
4) Set
5) Dictionary




**Numeric:**

In Python, numeric data type represents the data which has numeric value. Numeric value can be integer, floating number or complex numbers.

**Integers:** This value is represented by int class. It contains positive or negative whole numbers (without fraction or decimal). In Python there is no limit to how long an integer value can be.

Ex:
x= 1
y=23534958345
z=-2356
print(x)
print(y)
print(z)
print(type(x))
print(type(y))
print(type(z))

**Float:** This value is represented by float class. It is a real number with floating point representation. It is
specified by a decimal point.
Ex:
x= 1.14560
y=1.1
z=-35.689
print(x)
print(y)
print(z)
print(type(x))
print(type(y))
print(type(z))

**Complex Numbers**: Complex number is represented by complex class. It is specified as (real part) + (imaginary part) j.
x= 3+5j
y=6j
z=-7j
print(x)
print(y)
print(z)
print(type(x))
print(type(y))

**Boolean**

- Boolean represent one of the 2 values: True or False
- Booleans values are used in conditional statements.
- The Boolean values, True and False are treated as reserved words.

**None**

None is another special data type in Python. None is frequently used to represent the absence of a value. For example,

```
money = None
print(money)   # output: None
```

## Type Casting

It is the method to convert the variable data type into a certain data type. There can be two types of Type Casting in Python

1.Implicit Type Casting.

2.Explicit Type Casting

**Implicit Type Conversion**

In this, methods, Python converts data type into another data typeautomatically. In this process, users don't have to involve.

**Explicit Type Casting:** In this method, Python need user involvement to convert the variabledata type into certain data type in order to the operation required

**1. Integer conversion function: int()**

Ex:

```
a= int(3.5)
print(a)
print(type(a))
b= int("5")
print(b)
print(type(b))
```

**2. String conversion function: str()**

Ex:

```
a= str(8)
print(a)
print(type(a))
b= str(3.5)
print(b)
print(type(b))
```

**3. Float conversion function: float()**

```
a= float(8)
print(a)
print(type(a))
b= float("3")
print(b)
print(type(b))
c= float(35.50)
print(c)
print(type(c))
```

**4. complex() casting function: complex()**

Ex:

```
a= complex(8)
print(a)
print(type(a))
b= complex(3,5)
print(b)
print(type(b))
```

## Operators:

Python language supports a wide range of operators.

They are

1) Arithmetic Operators
2) Assignment Operator
3) Comparison Operators
4) Logical Operators
5) Bitwise Operators
6) Membership operator
7) Identity Operator

**Arithmetic Operators**

Arithmetic operators are used to execute arithmetic operations such as addition, subtraction, division, multiplication etc

| Operator | Description | Example |
|---|---|---|
| + | Addition | 5+2 gives 7 |
| - | Subtraction | 5-2 gives 2 |
| * | Multiplication | 5*2 gives 10 |
| / | Division | 5/2 gives 2.5 |
| % | Modulus: Gives the remainder | 5%2 gives 1 |
| // | Floor Division: Gives the integral part of the quotient | 5//2 gives 2 |
| ** | Exponent | 5**2 gives 25 |

**Assignment Operators**

- Assignment operators are used for assigning the values generated after evaluating the right operand to the left operand.
- Assignment operation always works from right to left. Assignment operators are either simple assignment operator or compound assignment operators.
- Simple assignment is done with the equal sign (=) and simply assigns the value of its right operand to the variable on the left.
- For example,

1. x = 5
   x = x+1
   print(x)  # will print the value 6
2. y = 4
   y +=2  # Compound assignment
   print(y)  #will print the value 6 here
3. z=z+2  # In this case error is thrown because z is undefined.

**Comparison Operators**

- When the values of two operands are to be compared then comparison operators are used.
- The output of these comparison operators is always a Boolean value, eitherTrue or False.
- The operands can be Numbers or Strings or Boolean values.
- Strings are compared letter by letter using their Unicodes
- For example: “P” is less than “Q”, and “Gpta” is greater than “Gptb”.

There are six comparison operators available in Python

| Opeartor | Operator name | Description |
|---|---|---|
| == | Is equal to | If the values of two operands are equal, then the condition becomes True. |
| != | Is not equal to | If the values of two operands are not equal, then the condition becomes True. |
| < | Is less than | If the value of left operand is lesser than the value of right operand, then condition becomes True. |
| <= | Is less than or equal | If the value of left operand is lesser than or equal to the value of right operand, then condition becomes True. |
| > | Is greater than | If the value of left operand is greater than the value of right operand, then condition becomes True. |
| >= | Is greater than or equal | If the value of left operand is greater than or equal to the value of right operand, then condition becomes True. |

**Logical Operators**

- The logical operators are used for comparing the logical values of their operands and to return the resulting logical value.
- The values of the operands on which the logical operators operate evaluate to either True or False.
- The result of the logical operator is always a Boolean value, True or False
- There are 3 logical operators available in python as shown in the table

| Operator | Description | Example |
|---|---|---|
| **and** | Returns True if both statements are True | 5 > 2 and 5 < 7 gives True |
| **or** | Returns True if at least one of the statement is True | 5 > 2 or 5 < 3 gives True |
| **not** | Reverses the result. | not(5 > 2 or 5 < 3) gives False |

**Logical Operators on Non boolean types:**

- Logical operators on Non Boolean types will always result in Non Boolean value

**Logical and** on Non Boolean data works as follows:
Assume x and y are two non boolean operands

- If x evaluates to False then the result is x
- If x evaluates to True then the result is y
- Ex :
  print(0 and 10) results in 0
  print(10 and 20) results in 20

- **Logical or** on Non Boolean data works as follows:
  Assume x and y are two non boolean operands
  - If x evaluates to False then the result is y
  - If x evaluates to True then the result is x
  - Ex :
    print(0 or 10) results in 10
    print(30 or 20) results in 30

## Bitwise Operators:

- In Python, bitwise operators are used to performing bitwise calculations on integers.
- The integers are first converted into binary and then operations are performed on bit by bit, hence the name bitwise operators.
- The following table shows the different bitwise opearators

| Operator | Name | Description |
|---|---|---|
| & | AND | Sets output bit to 1 if both input bits are 1 |
| \| | OR | Sets output bit to 1 if at least one of the two input bits is 1 |
| ^ | XOR | Sets output bit to 1 if exactly one of the two input bits is 1 |
| ~ | NOT | Inverts all bits |
| << | Left Shift | Shift left by pushing zeros in from the right and let the leftmost bits fall off |
| >> | Right Shift | Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off |

Example:

```
In [5]:
1 a=3 # binary 0011
2 b=5 # binary 0101
3 print(a&b)
4 print(a|b)
5 print(a^b)
6 print(10<<2)
7 print(10>>2)
```

```
1
7
6
40
2
```

Python language offers some special types of operators like the identity operator or the membership operator.

**Membership Operator:**

- in and not in are the membership operators in Python.
- They are used to test whether a value or variable is found in a sequence such as string, list, tuple, set and dictionary.

**in : Returns True if value/variable is found in the sequence**

**not in :Returns True if value/variable is not found in the sequence**

Example:

```
name="laxmi"
print("l" in name)          # prints True as the letter “l” is present in the string “laxmi”
print("y" not in name)    #prints True as the letter “y” is not present in the string
# “laxmi”
```

**Identity operators**

- is and is not are the identity operators in Python.
- They are used to check if two values (or variables) are located on the same part of the memory. Two variables that are equal  does not imply that they are identical.

**is :Returns True if the operands are identical (refer to the same object)**

**is not :Returns True if the operands are not identical (do not refer to the same object)**

## Operator Precedence and Associativity

Operator precedence determines the way in which operators are parsed with respect to each other. Almost all the operators have left-to-right associativity except = and ** operators

| Precedence | Operator | Name |
|---|---|---|
| 1 | () | Paranthesis |
| 2 | ** | Exponentiation |
| 3 | +a, -a, ~a | Unary plus, minus, complement |
| 4 | / * // % | Divide, Multiply, Floor Division, Modulo |
| 5 | + - | Addition, Subtraction |
| 6 | >> << | Shift Operators |
| 7 | & | Bitwise AND |
| 8 | ^ | Bitwise XOR |
| 9 | \| | Bitwise OR |
| 10 | >= <= > < | Comparison Operators |
| 11 | != == | Equality Operators |
| 12 | += -= /= *= = | Assignment Operators |
| 13 | is, is not, in, not in | Identity and Membership Operators |
| 14 | and, or, not | Logical Operators |