Python Programming - II

Data Types and Conversion

CONTENTS

Practicing with Data Types

NUMBERS:

- Integer
- Float
- Complex

```
>>> Days = 7

>>> Pi = 3.14

>>> Long = 0122L (Prior to Python 2.x)

>>> num = 3.14j
```

WAP - Addition of two numbers one complex and Long using 2 variables.

Tip - Using Print statement you can print the values.

Example:

```
>>> print(a)
12
>>> print(c)
5.0
```

Practicing with Data Types

STRING:

```
>>> Name = 'Instagram'

>>> Rating = "4.7 stars"

>>> Type = "Social Networking"

>>> Desc = "Made for photo sharing across the Globe with variety of 'filters' enhancing your photos with advanced tools to improve the quality and completely made using Python Programming Language"
```

WAP - To print statements in a meaning full paragraph about any MOVIE

Tip - Use + operator to append two sentences in a single print statement.

Example:

```
>>> print(Name,"is a",Rating,"rated app")
```

Practicing with Data Types

STRING:

```
>>> to_do = "LEARN PYTHON"
```

0 1 2 3 ... 11

```
>>> print to_do

>>> print to_do[0]

>>> print to_do[6:11]
```

String is simply an array of characters, and hence you can achieve a fine level of control over it.

WAP - To Assign a string QWERTY to a variable and print "T Y ER" using same variable in single line.

Practicing with Data Types

LISTS:

List is a compund data type, consisting a editable sequence of items enclosed in square brackets ([ ])

```
>>> grocery_list = ['Tomatoes', 'Broccoli', 'Mushrooms', 'Beetroot']
>>> user_details = ['Chitvan', 25, 'A-ve']

>>> print(grocery_list[1])

>>> print(user_details *2)

>>> print(grocery_list[1:3])

>>> print(grocery_list[2:])
```

WAP - To add the above two lists into new list named "**combined_list**", w/o typing items individually. Print only last 2 items from each sublists using combined_list.

Practicing with Data Types

TUPLES:

Tuples are similar to list, it is a sequence data type with read-only access once it is created and are enclosed in round brackets (( ))

```
>>> pytuple = ('Angela', 'Hilary', 'Kiran', 2017)

>>> print(pytuple)

>>> print(pytuple[0])

>>> print(pytuple[0:3])

>>> print(pytuple *2)

>>> print(pytuple + pytuple)
```

WAP - To assign a tuple with all types of integers and string and print any one string and integer on same line

Ex:

Hilary 2017

Practicing with Data Types

DICTIONARY:

Dictionary in python is similar to Hash table, they can be stored in Key : Value pairs. It can be in any Python type, object

```
>>> dict = {}

>>> dict['Apple'] = "A fruit, red in color"

>>> dict['Appoint'] = "To call upon"

>>> dict2 = {'Name':'Edward Snowden', 'code-name' : 'Citizenfour',
'emp-id' : 12203, 'OS' : 'TAILS'}

>>> print(dict)
>>> print(dict2)
>>> print(dict.keys())
>>> print(dict.values())
```

WAP - Design a Dictionary for Number in Numerical to Word as Key value pair, print it in two columns

tip: use '\n'

Data Type Conversion

Function	Description
int(x)	Converts x to an integer.
float(x)	Converts x to a floating-point number.
complex(real [,imag])	Creates a complex number.
str(x)	Converts object x to a string representation.
repr(x)	Converts object x to an expression string.
eval(str)	Evaluates a string and returns an object.
tuple(s)	Converts s to a tuple.
list(s)	Converts s to a list.
set(s)	Converts s to a set.
dict(d)	Creates a dictionary. d must be a sequence of (key,value) tuples.
frozenset(s)	Converts s to a frozen set.
chr(x)	Converts an integer to a character.

Data Type Conversion

Function	Description
unichr(x)	Converts an integer to a Unicode character.
ord(x)	Converts a single character to its integer value.
hex(x)	Converts an integer to a hexadecimal string.
oct(x)	Converts an integer to an octal string.

Operators - 1. Comparison

As the name goes, in programming paradigm it means Relational Operators. Helps deriving a relation between two entities.

Ex. Value of a=**20** & b=**30**

Operator	Description	Expression
==	Values of two operand if are equal, then condition returns true	(a == b) FALSE
!=	Values of two operand if are not equal, then condition returns true	(a != b) TRUE
>	Left value if greater, then condition is true	(a > b) FALSE
<	Right value if greater, then condition is true	(a < b) TRUE
>=	Left value if greater than or equal to Right value, condition answer true	(a >= b) FALSE
<=	Right value if greater than or equal to Left value, condition answer true	(a <= b) FALSE

Operators - 1. Comparison

Example:

```
>>> a=21
>>> b=20
>>> if(a == b):
...     print("A is qual to B")
... else:
...     print("They are not equal")
...
They are not equal
```

Try the remaining Relational Operators

Operators - 2. Assignment

Operator	Description	Example
=	Assign values from right side operands to left side operand	sum = a + b
+=	It adds right operand to the left operand and assign the result to left operand	b += a b = b + a
-=	It subtracts right operand from the left operand and assign the result to left operand	b -= a b = b - a
*=	It multiplies right operand with the left operand and assign the result to left operand	b *= a b = b * a
/=	It divides left operand with the right operand and assign the result to left operand	b /= a
%=	It takes modulus using two operands and assign the result to left operand	b %= a
**=	Performs exponential (power) calculation on operators and assign value to the left operand	b **= a
//=	It performs floor division on operators and assign value to the left operand	b //= a

Operators - 2. Assignment

Multiple Assignment:

One or more variables can be assigned one or multiple values. It shares similar terminology as of Mapping Cardinality

Example:

```
>>> a = b = c = 1
>>> a , b, c = 1, 2, 'sum ='

>>> print(c,a+b)
```

Operators - 3. Bitwise

Operator	Description
& AND	Operator copies a bit to the result, if it exists in both operands
| OR	It copies a bit, if it exists in either operand.
^ XOR	It copies the bit, if it is set in one operand but not both.
~ 1's Complement	It is unary and has the effect of 'flipping' bits.
<< Left Shift	The left operand's value is moved left by the number of bits specified by the right operand.
>> Right Shift	The left operand's value is moved right by the number of bits specified by the right operand.

Operators - 3. Bitwise

Example:

```
>>> a=10
>>> b=13

>>> print("ANDing a:", a, " & b:", b, "is ", a&b, ":",bin(a&b))

ANDing a: 10  & b: 13 is  8 : 0b1000
```

WAP - to obtain ANDing of 2 complemented values a=2 & b=3 and print it in both Integer and Binary formats

Operators - 4. Logical

Assume that variables a and b holds the value of 'True' and 'False' respectively

Operator	Description	Example
and (Logical AND)	If both the operand are true then condition becomes true.	a and b
or (Logical OR)	If any of the two operands are non-zero then condition becomes true.	a or b
not (Logical NOT)	Used to reverse the logical state of its operand.	not(a and b)

Operators - 5. Membership

Python's membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below-

Operator	Description
in	Evaluates to true, if it finds a variable in the specified sequence and false otherwise.
not in	Evaluates to true, if it does not find a variable in the specified sequence and false otherwise.

Consider the data: a = [1, 2, 3, 4, 5], x=2, y=7

Find the output of

```
>>> x-y in a
>>> y-x in a
>>> y not in a
```

Operators - 6. Identity

Identity operators compare the memory locations of two objects. There are two Identity operators as explained below:

Operator	Description
is	Evaluates to true, if it finds a variable in the specified sequence and false otherwise.
is not	Evaluates to true, if it does not find a variable in the specified sequence and false otherwise.

Consider the data: A = 1, B = 2

Find the output of

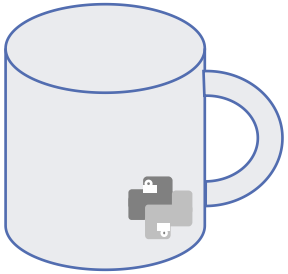
```
>>> A is B
>>> A is not id(A)
>>> A is not B
```

Precedence of Operators

Operator	Description
**	Exponention
~ + -	Complement, Unary plus and Minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive 'OR' and regular 'OR'
<= < > >=	Comparison operators
<> == !=	Equality operators

Precedence of Operators

Operator	Description
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
no or and	Logical operators



To Be Resumed after a Break