SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Artificial Intelligence and Data Science

Course Name – Computational Thinking and Python Programming

I Year / I Semester

Unit 2-DATA, EXPRESSIONS, STATEMENTS

VARIABLES:

•A variable allows us to store a value by assigning it to a name, which can be used later.
•Named memory locations to store values.
•Programmers generally choose names for their variables that are meaningful.
•It can be of any length. No space is allowed.
•We don't need to declare a variable before using it. In Python, we simply assign a value to a variable and it will exist.

Assigning value to variable:

Value should be given on the right side of assignment operator(=) and variable on left side.

```
>>>counter =45
print(counter)
```

Assigning a single value to several variables simultaneously:

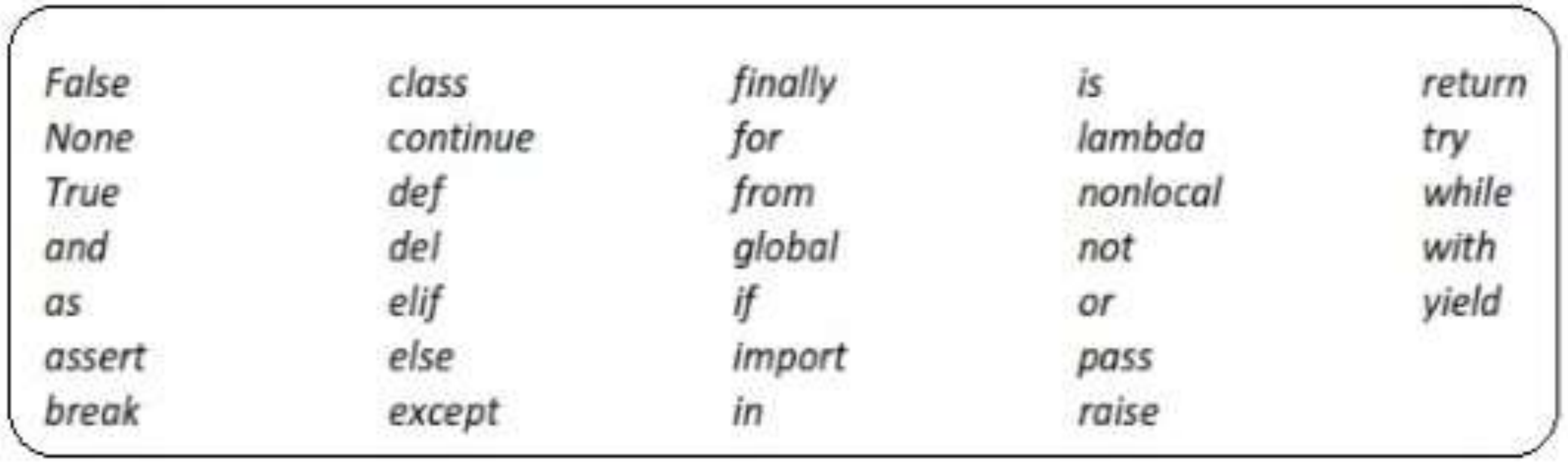
```
>>> a=b=c=100
```

Assigning multiple values to multiple variables:

```
>>> a,b,c=2,4,"ram"
```

KEYWORDS:

1. Keywords are the reserved words in Python.
2. We cannot use a keyword as variable name, function name or any other identifier.
3. They are used to define the syntax and structure of the Python language.
4. Keywords are case sensitive.



False	*class*	*finally*	*is*	*return*
None	*continue*	*for*	*lambda*	*try*
True	*def*	*from*	*nonlocal*	*while*
and	*del*	*global*	*not*	*with*
as	*elif*	*if*	*or*	*yield*
assert	*else*	*import*	*pass*	
break	*except*	*in*	*raise*	

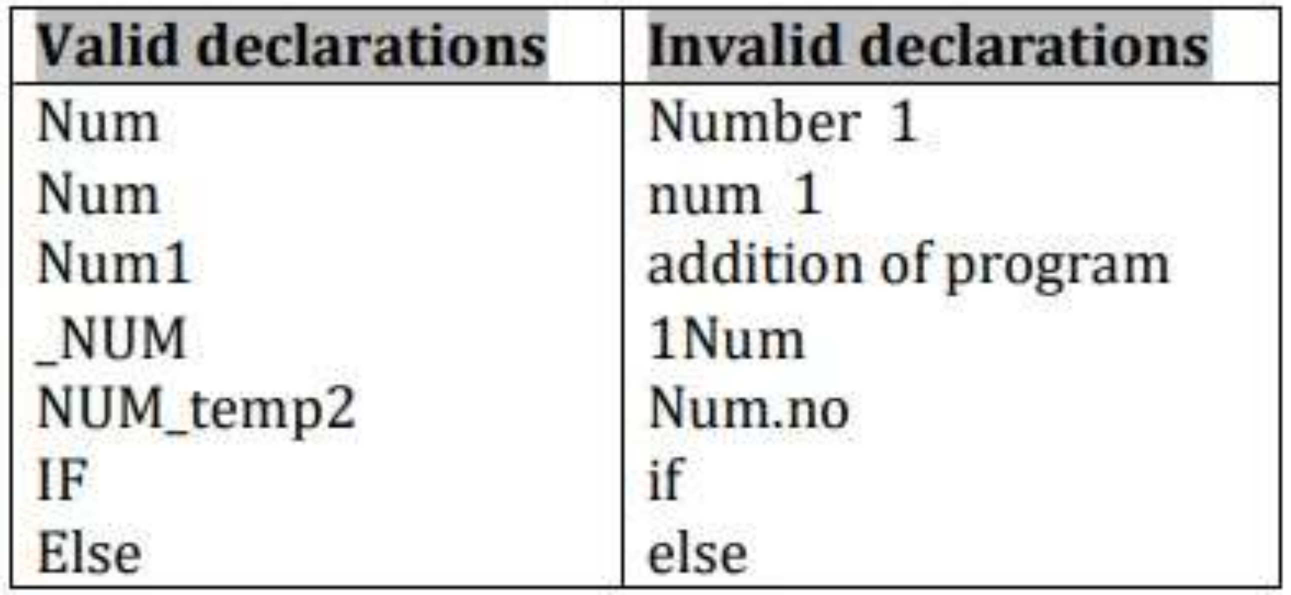
IDENTIFIERS:

Identifier is the name given to entities like class, functions, variables etc. in Python.

1. Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore (_).
2. all are valid example.
3. An identifier cannot start with a digit.
4. Keywords cannot be used as identifiers.
5. Cannot use special symbols like !, @, #, $, % etc. in our identifier.
6. Identifier can be of any length.

Example:

Names like myClass, var_1, and **this_is_a_long_variabl**e



Valid declarations	Invalid declarations
Num	Number 1
Num	num 1
Num1	addition of program
_NUM	1Num
NUM_temp2	Num.no
IF	if
Else	else

STATEMENTS AND EXPRESSIONS:

-

Statements:

-Instructions that a Python interpreter can executes are called statements.

-A statement is a unit of code like creating a variable or displaying a value.

```
>>> n = 17
>>> print(n)
```

Here, The first line is an assignment statement that gives a value to n.

The second line is a print statement that displays the value of n.

-

Expressions:

-An expression is a **combination of values, variables, and operators.**

-A value all by itself is considered an expression, and also a variable.

So the following are all legal expressions:

```
>>>  42
42
>>>  a=2
>>>  a+3+2
7
>>>  z=("hi"+"friend")
>>>  print(z)
Hifriend
```

.

INPUT AND OUTPUT

INPUT: Input is data entered by user (end user) in the program.
In python, **input () function** is available for input.

Syntax for input() is:

variable = input (“data”)

Example:

```
>>>  x=input("enter the name:") enter the name: george
>>>y=int(input("enter the number"))
enter the number 3
#python accepts string as default data type. conversion is required for type.
```

OUTPUT: Output can be displayed to the user using Print statement .

Syntax:

print (expression/constant/variable)

Example:

```
print ("Hello") Hello
```

COMMENTS:

1. A **hash sign (#)** is the beginning of a comment.
2. Anything written after # in a line is ignored by interpreter.

Eg: percentage = (minute * 100) / 60   **# calculating percentage of an hour**

3. Python **does not have multiple-line commenting feature.** You have to comment each line individually as follows :

Example:

```
#              This is a comment.
#              This is a comment, too.
#              I said that already.
```

LINES AND INDENTATION:

1. Most of the programming languages like C, C++, Java use braces { } to define a block of code. But, python uses indentation.
2. Blocks of code are denoted by line indentation.
3. It is a space given to the block of codes for class and function definitions or flow control.

Example:

```
a=3
b=1
if a>b:
print("a is greater")
else:
print("b is greater")
```

QUOTATION IN PYTHON:

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals.

Anything that is represented using quotations are considered as string.
1. single quotes (' ') Eg, 'This a string in single quotes'
2. double quotes (" ") Eg, "'This a string in double quotes'"
3. triple quotes(""" """) Eg, This is a paragraph. It is made up of multiple lines and sentences."""

TUPLE ASSIGNMENT

1. An assignment to all of the elements in a tuple using a single assignment statement.
2. Python has a very powerful **tuple assignment** feature that allows a tuple of variables on the left of an assignment to be assigned values from a tuple on the right of the assignment.
3. The left side is a tuple of variables; the right side is a tuple of values.
4. Each value is assigned to its respective variable.
5. All the expressions on the right side are evaluated before any of the assignments. This feature makes tuple assignment quite versatile.
5. Naturally, the number of variables on the left and the number of values on the right have to be the same.

```
>>>   (a, b, c, d) = (1, 2, 3)
ValueError: need more than 3 values to unpack
```

Example:

-It is useful to swap the values of two variables. With **conventional assignment statements**, we have to use a temporary variable. For example, to swap a and b:

Swap two numbers

```
a=2;b=3
print(a,b)
temp = a
a = b
b = temp
print(a,b)
```

Output:

```
(2, 3)
(3, 2)
>>>
```

-Tuple assignment solves this problem neatly:

(a, b) = (b, a)

One way to think of tuple assignment is as tuple packing/unpacking.

In tuple packing, the values on the left are 'packed' together in a tuple:

```
>>> b = ("George", 25, "20000")    # tuple packing
```

-In tuple unpacking, **the values in a tuple on the right are 'unpacked' into the variables/names on the right:**

```
>>> b = ("George", 25, "20000") # tuple packing
>>> (name, age, salary) = b # tuple unpacking
>>>   name
'George'
>>>   age
25
>>>   salary
'20000'
```

-The right side can be any kind of sequence (string,list,tuple)

Example:

-To split an email address in to user name and a domain

```
>>>     mailid='god@abc.org'
>>>     name,domain=mailid.split('@')
>>>     print name
god
print (domain)
abc.org
```