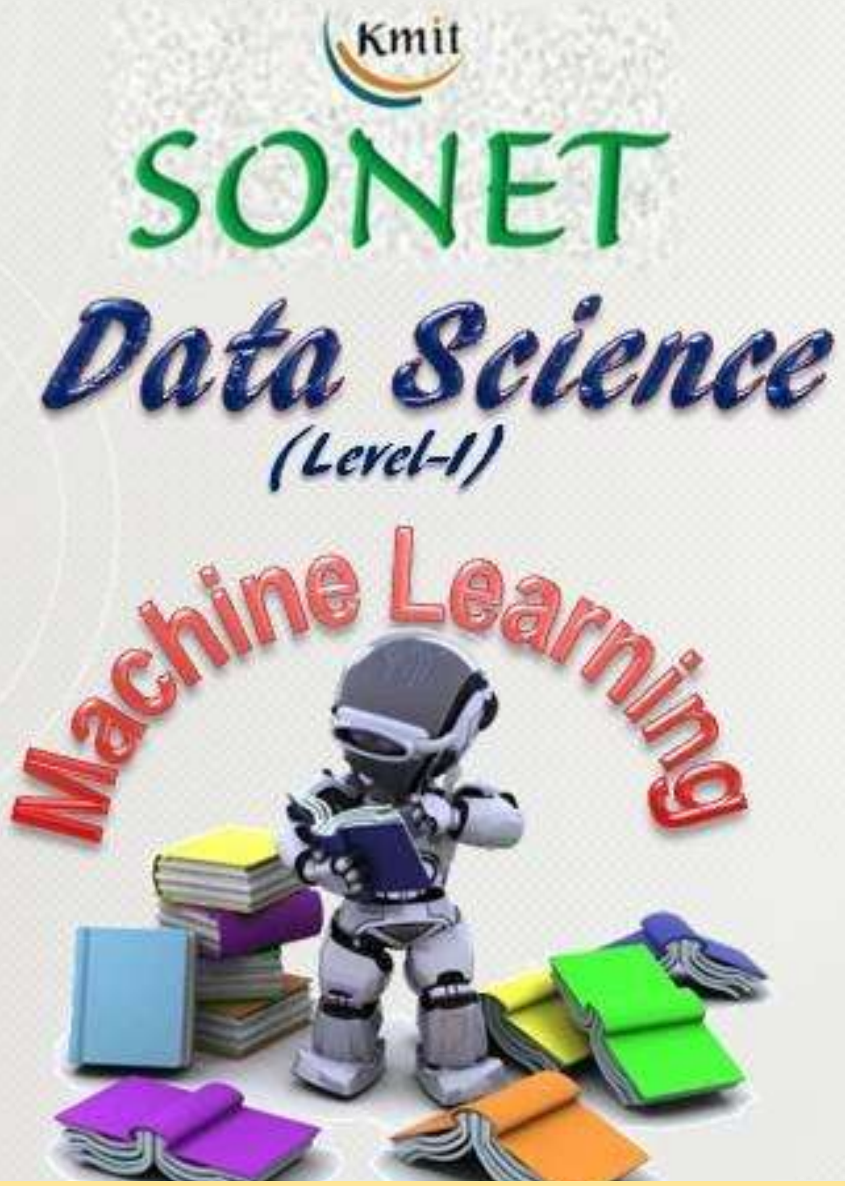


Ajeet K. Jain, M. Narsimlu

(ML TEAM)- SONET, KMIT, Hyderabad

Session - 3

This session deals with

Python Tokens(Building blocks)

Single Line and Multi Line comments

Examples on each Building block

Exercises

Python Building blocks

Identifiers

A Python identifier is a name used to identify a Variable, functions. Class, modules and objects

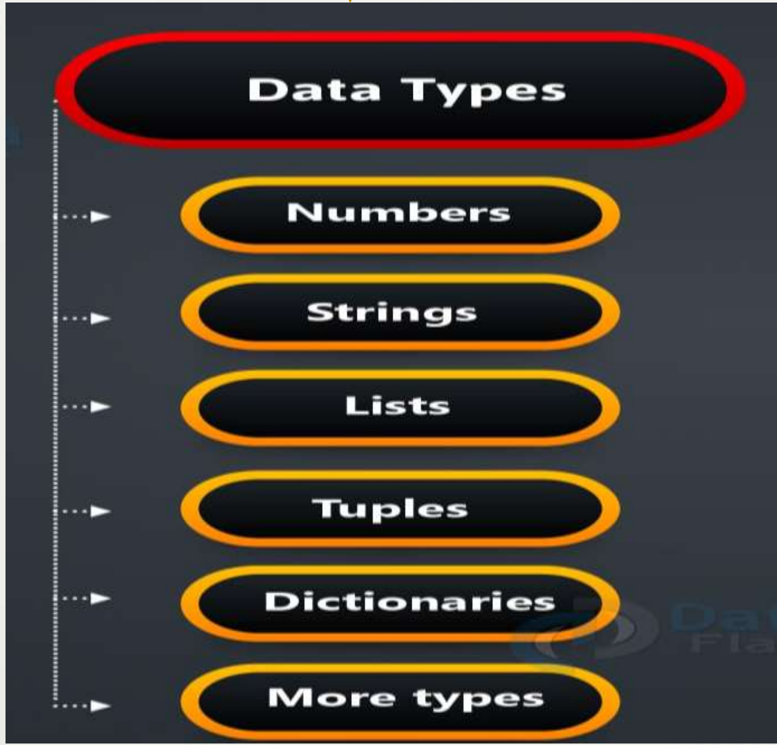
Keywords

Reserved words(and ,or, if, if else)

Operators

1.Arithematic,2.Logical,3.Relational,
4.Assignment,
5.Bitwise and 6.Special

Data Types



Comments

Comments are an integral part of any program and comments are ignore by interpreter

It will help us to increase the readability of your code

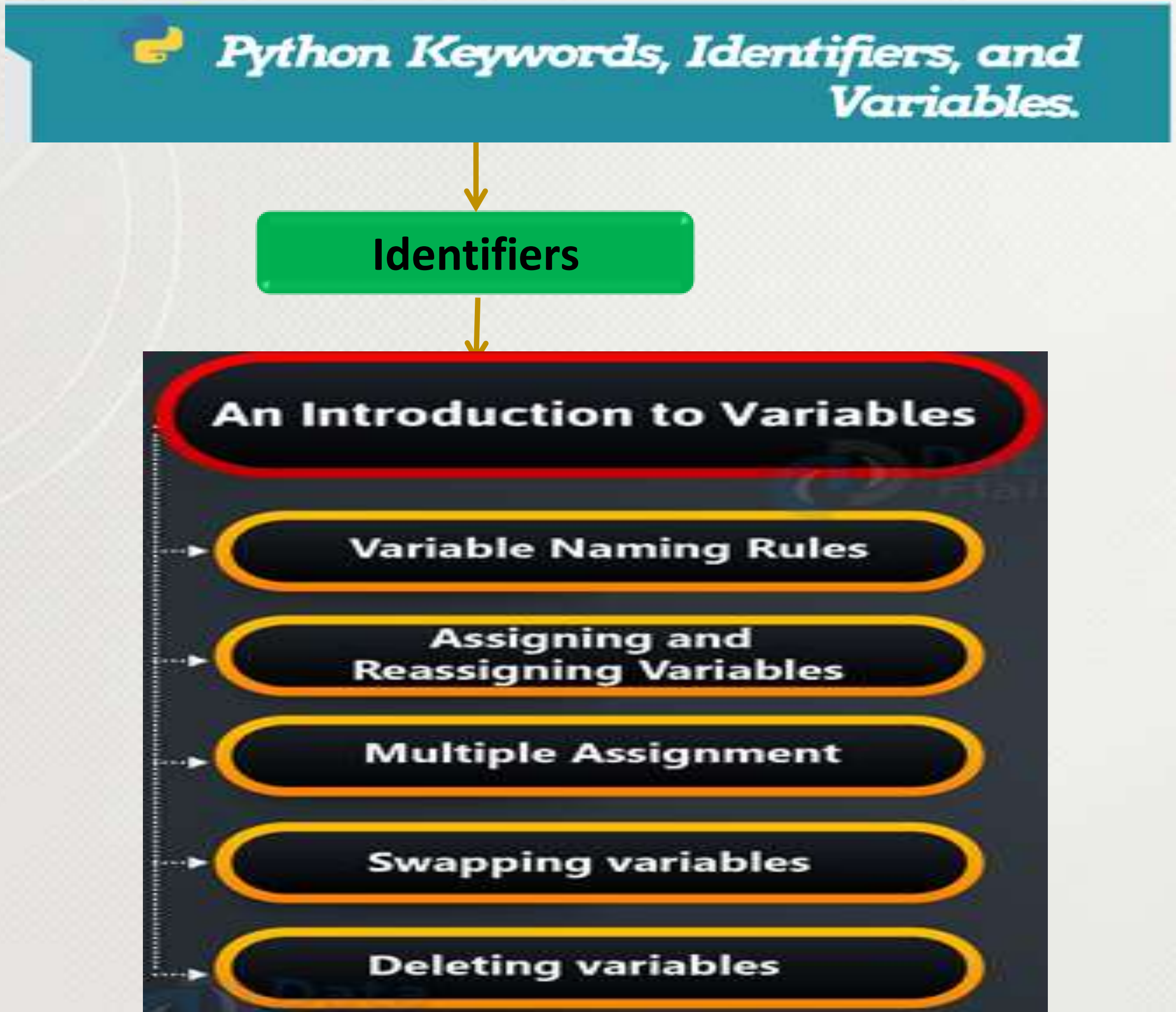
it’s important to make sure that **your code can be easily understood by others**

Describe parts of the code where necessary to facilitate the understanding of programmers

Single line comments are denoted by hash mark #
Ex: #python building blocks

Multi line comments are enclosed with “””   “”” or ‘’’   ‘’’
“””
1.Identifiers
2.Keywords
3.Data Types
“”””””

Identifiers


Python Keywords, Identifiers, and Variables.
Identifiers
An Introduction to Variables
Variable Naming Rules
Assigning and Reassigning Variables
Multiple Assignment
Swapping variables
Deleting variables

Identifiers in Python

Identifiers

A Python identifier is a name used to identify a
variable,
function,
class,
module or
other object.

Kmit SONET

Identifiers in Python

DATA SCIENCE

Python does not allow punctuation characters such as @, $, and % within identifiers.
Python is a case sensitive programming language.

we know that Python is a dynamically-typed language, we don’t specify the type of a variable when declaring one.

A variable is a container for a value.
Based on the value assigned, the interpreter decides its data type.
You can always store a different type in a variable.

SONET

Identifiers in Python

DATA SCIENCE

Identifier Naming Rules

Rule 1: An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

```
>>> A = 7
>>> print(A)
7
>>> A = Sree
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name ‘Sree' is not defined
>>> A = “Sree"
>>> print(A)
Sree
```

Identifier Naming Rules

- <u>Rule 2</u>: The rest of the identifier may contain letters(A-Z/a-z), underscores(_), and numbers(0-9).
- >>> year2 = 2018
- >>> year2
- 2018

Identifiers

- Rule 3: Python is case-sensitive, and so are Python identifiers.
- Name and
- name
- are two different identifiers.
- name=‘Netaji'
- >>> name
- ' Netaji'
- >>> Name
- Traceback (most recent call last):
- File "<pyshell#21>", line 1, in <module>
- Name

Exercises-1

Create a script for the following data:

19os_marks=86

Phy=67

Maths=89

I_Year_c=78

First_Name=“Sree”

Middle_Name=“Ram”

Last_Name=“Mohan”

$python_cost=560

Save the script name as “variables.py”

Check valid statements if it is valid print it using print() function.

Keywords

- Reserved words (keywords) cannot be used as identifier names.
- and def import not as del with
- finally in or try while yield
- assert elif for pass nonlocal from
- break else is raise exec lambda
- class except if global
- continue return
- False True None
- Different python versions have different keywords.

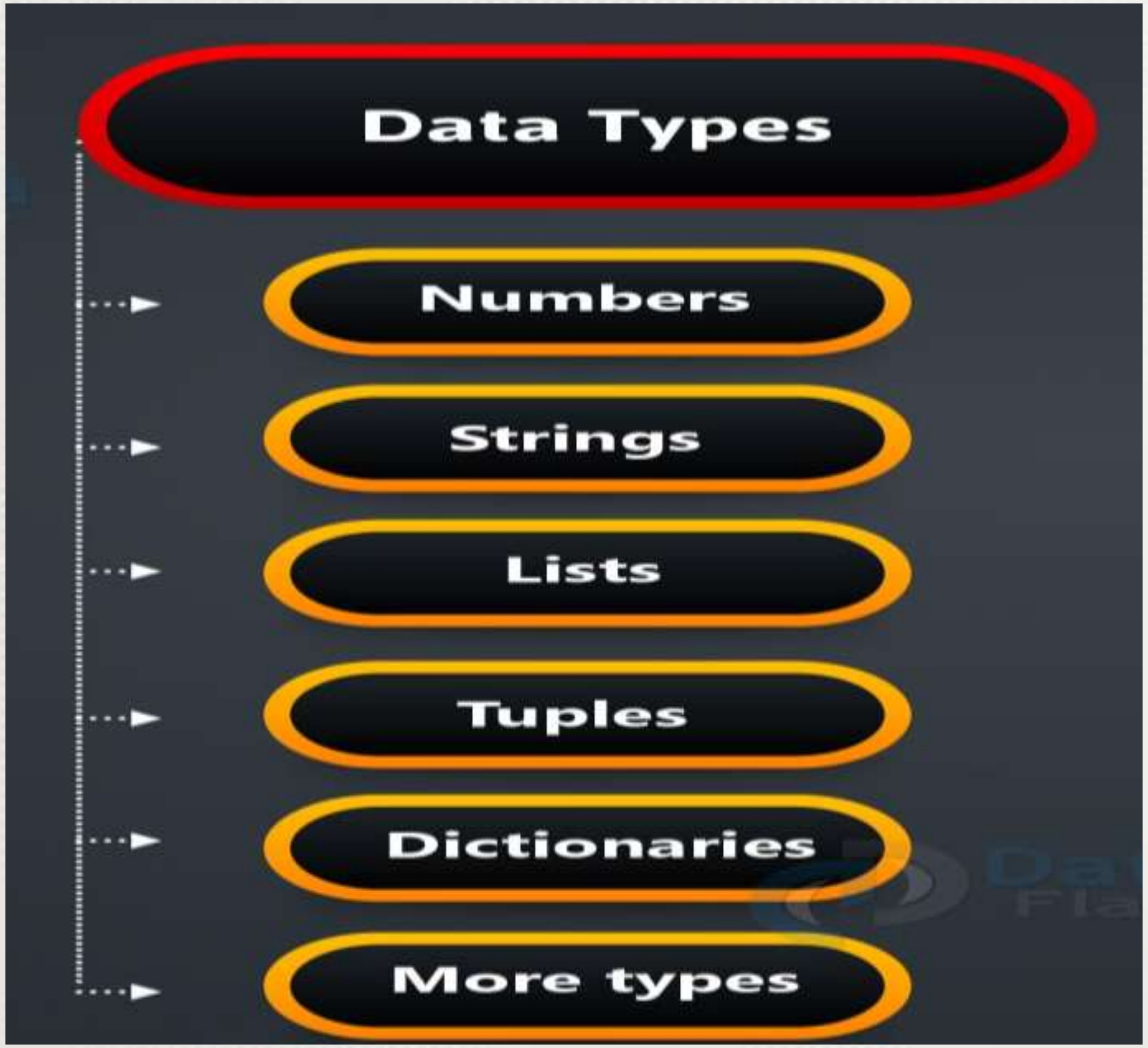
Keywords

- How to obtain Keywords in your Python version?
- >>> import keyword
- Here import is a statement using which we are importing module keyword in our program.
- Module is a .py file
- Every .py file is also called module
- >>> print(keyword.kwlist)
- ['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

Exercises-2

Find out which of the statements are valid statement
1.product_name=“python”
2.global=48
3.None=“Java”
4.local=“Bhagyanagar”
5.nonlocal=“Pune”
Save the script name as “py_keyword.py”
Display the valid statements and remove invalid statement

Data Types



Data Types
Numbers
Strings
Lists
Tuples
Dictionaries
More types

Number Data Types

- Numeric Types -- int, float, long, complex
- There are four distinct numeric types:

1. *plain integers*,
2. *long integers*,
3. *floating point numbers*, and
4. *complex numbers*.

- In addition, Booleans are a subtype of plain integers.
- Plain integers (also just called *integers*) are implemented using long in C, which gives them at least 32 bits of precision.
- Long integers have unlimited precision.

Number Data Types

- Numeric Types -- int, float, long, complex
- Floating point numbers are implemented using double in C.

Python type()

- **Python** have a built-in method called as **type** which generally come in handy while figuring out the data **type** of variable used in the program in the runtime.
- If a single argument (object) is passed to **type()**built-in, it returns data **type** of the given object.

Python id()

- **Python** have a built-in method called as **id** which returns identity of an object.
- Identity is an integer or long integer which is unique or constant for this object during its life time.
- Syntax id(object name)
- A = 10
- print(id(A))
- **1404123696**

Python id and type()

id() Function : Returns Unique number
Assigning one object to other

```
a = 20
a1 = a
print(a,a1)
print("data type of a",type(a),"value of a",a)
print("data type of a1",type(a1),"value of a1",a1)
print("Address of a",id(a),"Address of a1",id(a1))
```

Result id and type()

- 20 20
- data type of a <class 'int'> value of a 20
- data type of a1 <class 'int'> value of a1 20
- Address of a 1404123696 Address of a1 1404123696

id() Function : Returns Unique number
Assigning same values to different objects

```
b=10
b1=10
print("value of b",b,“a Addr  ",id(b),"value of b1",b1,“b Addr ",id(b1))
b=100
print("value of b",b,“b Addr ",id(b),"value of b1",b1,“b1 Addr ",id(b1))
```

Result

- value of b 10 a Addr   1401895312 value of b1 10 b Addr 1401895312
- value of b 100 b Addr 1401896752 value of b1 10 b1 Addr 1401895312

Type conversion

- Note in C that 6 / 4 gives 1 and not 1.5.
- This so happens because 6 and 4 both are integers and therefore would evaluate to only an integer constant.
- Similarly 5 / 8 evaluates to zero, since 5 and 8 are integer constants and hence must return an integer value.

Type Conversion

- >>> 6/4
- 1.5
- >>> 10/3
- 3.3333333333333335
- >>> 5/8
- 0.625

Type conversion

- Type Conversion:
- 1.Implicit conversion
- 2.Explicit Conversion

- 1.Implicit conversion
- Converting from lower level data type to higher level data. by default data type is implicit
- Ex: int to float
- i1=4
- i2=2
- res=i1/i2
- print(res)

Explicit conversion

- **Converting from higher level data type to lower level data type**
- **Ex: float to int , string to int , string to float**
- **Example1:**
- f1=68.5
- res=int(f1)
- print(res)
- **Ex2:**
- f1="23.84"
- res=float(f1)
- print(res)

Reading Data from the user

- The input Function.
- There are hardly any programs without any input.
- Input can come in various ways, for example
- from a database,
- another computer,
- mouse clicks
- mouse clicks and movements
- or from the internet. ...
- For this purpose, Python provides the function input().
- Input() has an optional parameter, which is the prompt string.

Input()

```
• >>> X = input() # input() returns string
• 34
• >>> print(x)
• 34
• >>> x+6
• Traceback (most recent call last):
•   File "<pyshell#2>", line 1, in <module>
•     x+6
• TypeError: Can't convert 'int' object to str implicitly
```

Input()

- X= int(input()) # Typecasting string to integer
- X= int(input("Enter value to x: "))
- # we are giving some text in input function

Reading data from keyboard

- Reading data from keyboard using input().but input() function by default it will take string as input
- s1=input("enter value1 ")
- s2=input("enter value2 ")
- print(s1+s2)
- Output:
- enter value1 4
- enter value2 2
- 42

Reading data from keyboard

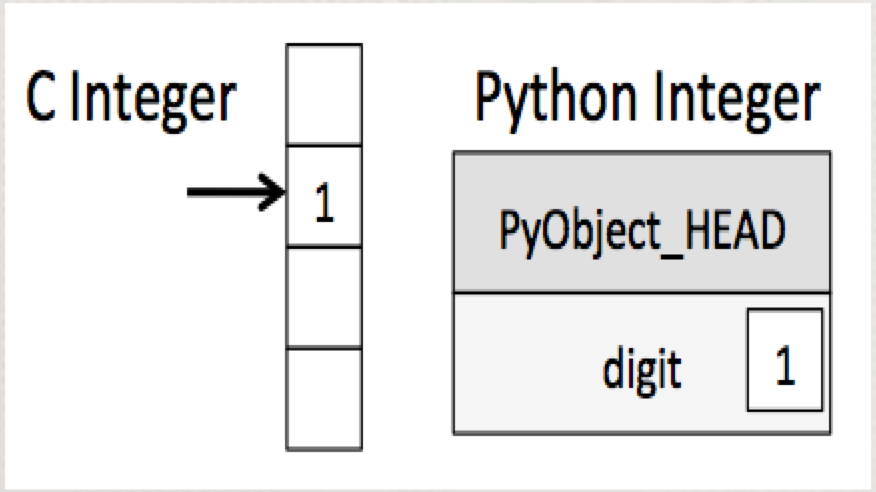
- s1=int(input("enter value1 "))
- s2=int(input("enter value2 "))
- print(s1+s2)
- Output:
- enter value1 4
- enter value2 2
- 6

Memory allocation for data types

PyObject_HEAD is the part of the structure containing the reference count, type code, and other pieces

A C integer is essentially a label for a position in memory whose bytes encode an integer value

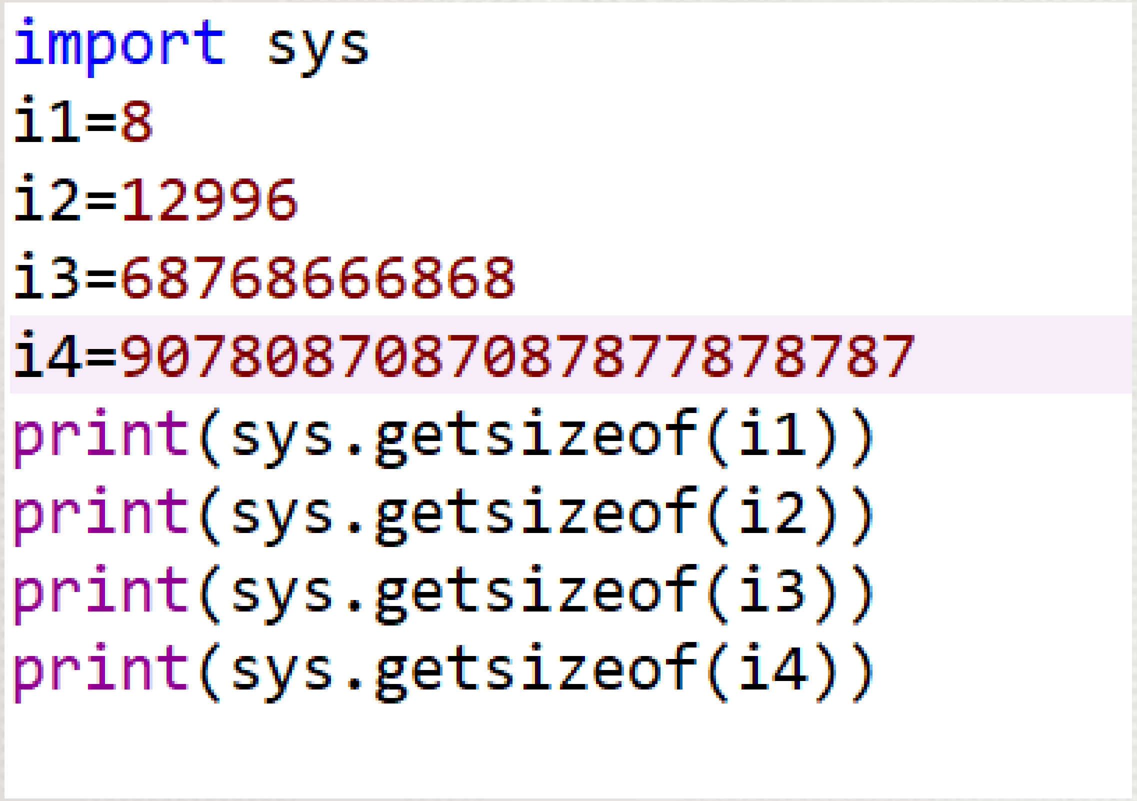
A Python integer is a pointer to a position in memory containing all the Python object information, including the bytes that contain the integer value.



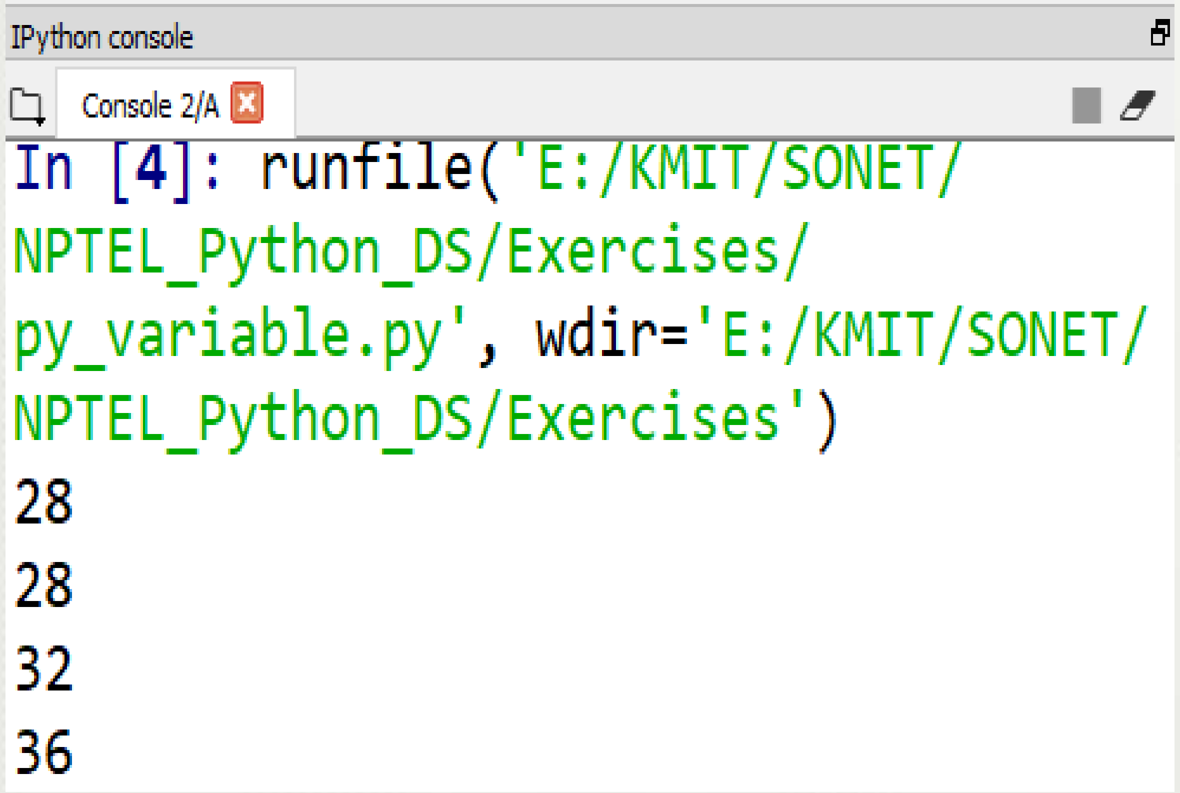
This extra information in the Python integer structure is what allows Python to be coded so freely and dynamically.

Memory allocation for data types

Output



```
import sys
i1=8
i2=12996
i3=68768666868
i4=9078087087087877878787
print(sys.getsizeof(i1))
print(sys.getsizeof(i2))
print(sys.getsizeof(i3))
print(sys.getsizeof(i4))
```



```
In [4]: runfile('E:/KMIT/SONET/
NPTEL_Python_DS/Exercises/
py_variable.py', wdir='E:/KMIT/SONET/
NPTEL_Python_DS/Exercises')
28
28
32
36
```

Memory allocation for Float data types

It does not return the size of any individual float, it returns the size of the float *class*.

Float class contains a lot more data than just any single float, so the returned size will also be much bigger

Every float object will contain a reference counter and a pointer to the type (a pointer to the float class)
Size of float is fixed in python i.e 24 bits

Example:

```
import sys
F2=8.8
F3=65.45
f1=808080807979999.99808080
print(sys.getsizeof(f1))
print(sys.getsizeof(f2))
print(sys.getsizeof(f3))
```

Output

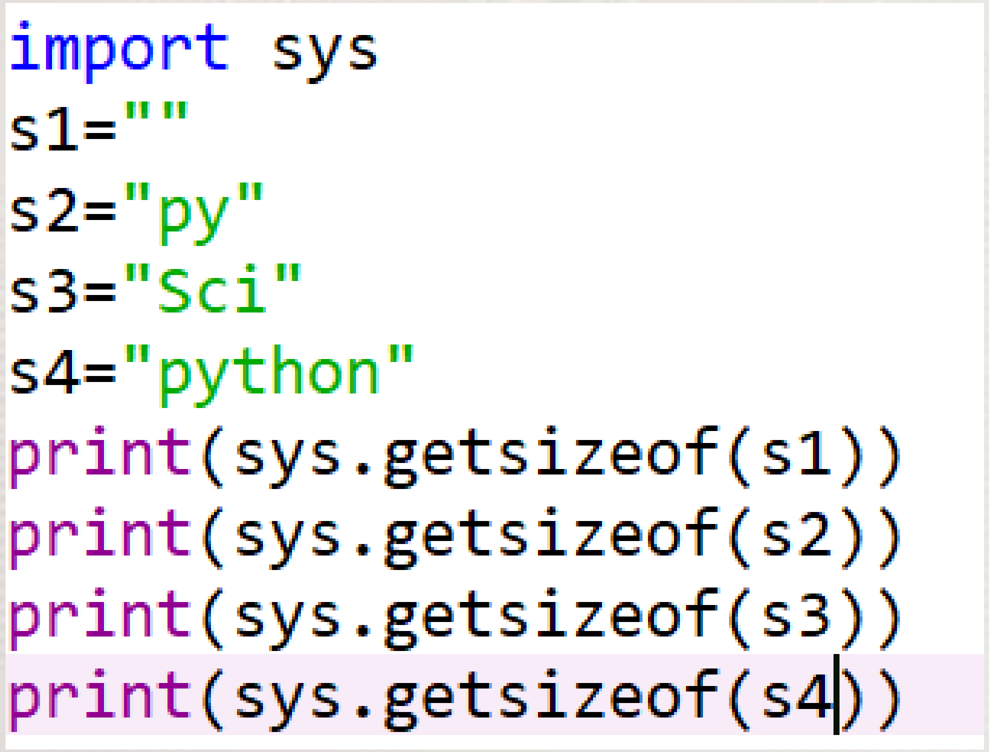
```
24
```

Memory allocation for data types

The size of empty string is 49 bits in spyder

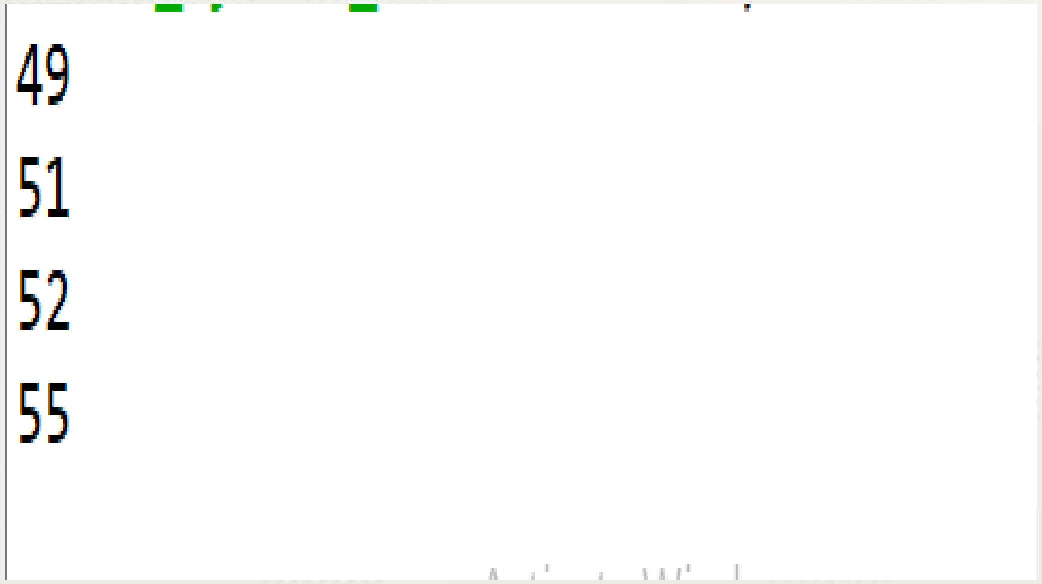
The size of string object is varying based on the data which string object has

Ex:



```
import sys
s1=""
s2="py"
s3="Sci"
s4="python"
print(sys.getsizeof(s1))
print(sys.getsizeof(s2))
print(sys.getsizeof(s3))
print(sys.getsizeof(s4))
```

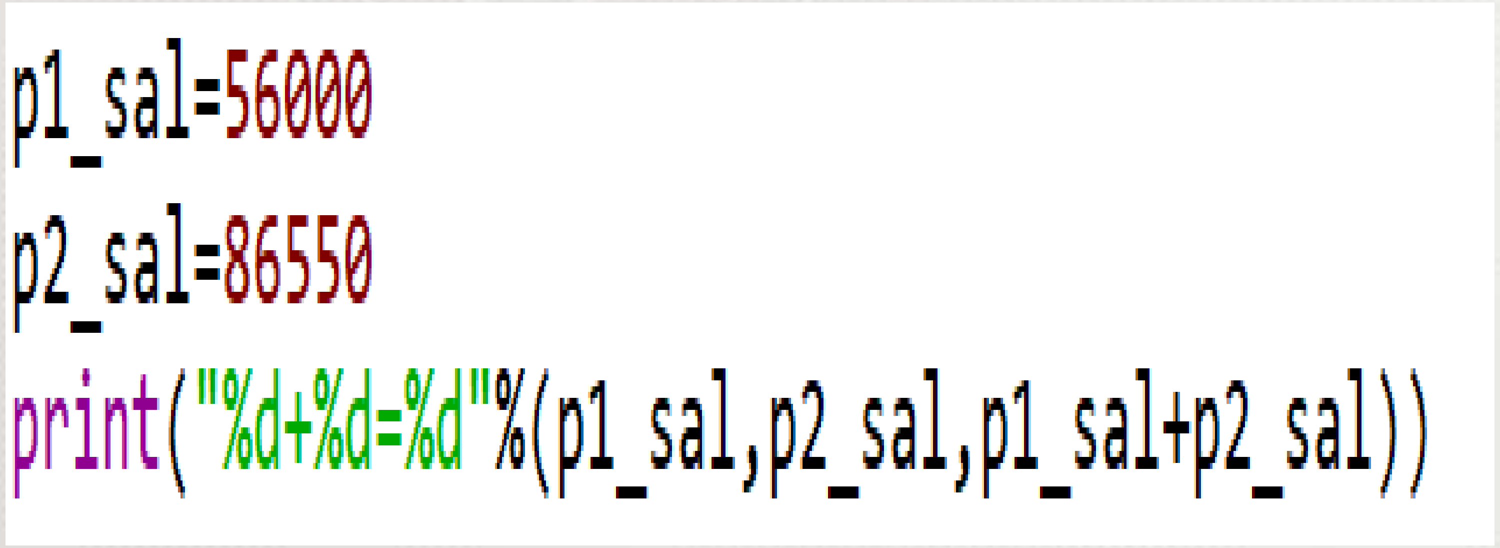
Output



```
49
51
52
55
```

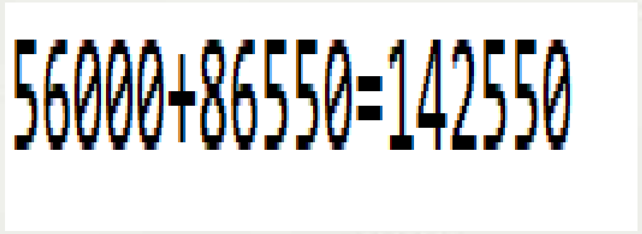
Formating the data

Format specifier for int is %d
Format specifier for float is %f
Format specifier for string is %s



```
p1_sal=56000
p2_sal=86550
print("%d+%d=%d"%(p1_sal,p2_sal,p1_sal+p2_sal))
```

Output

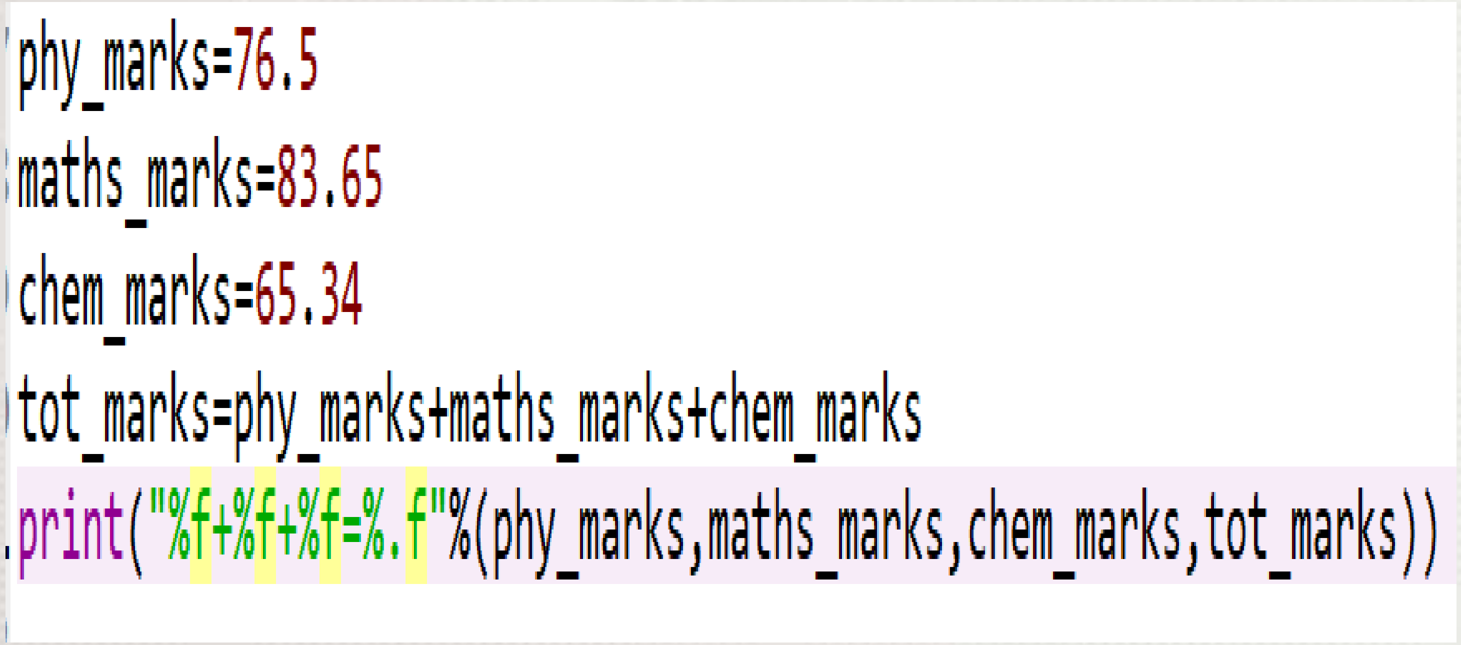


```
56000+86550=142550
```

Kmit SONET

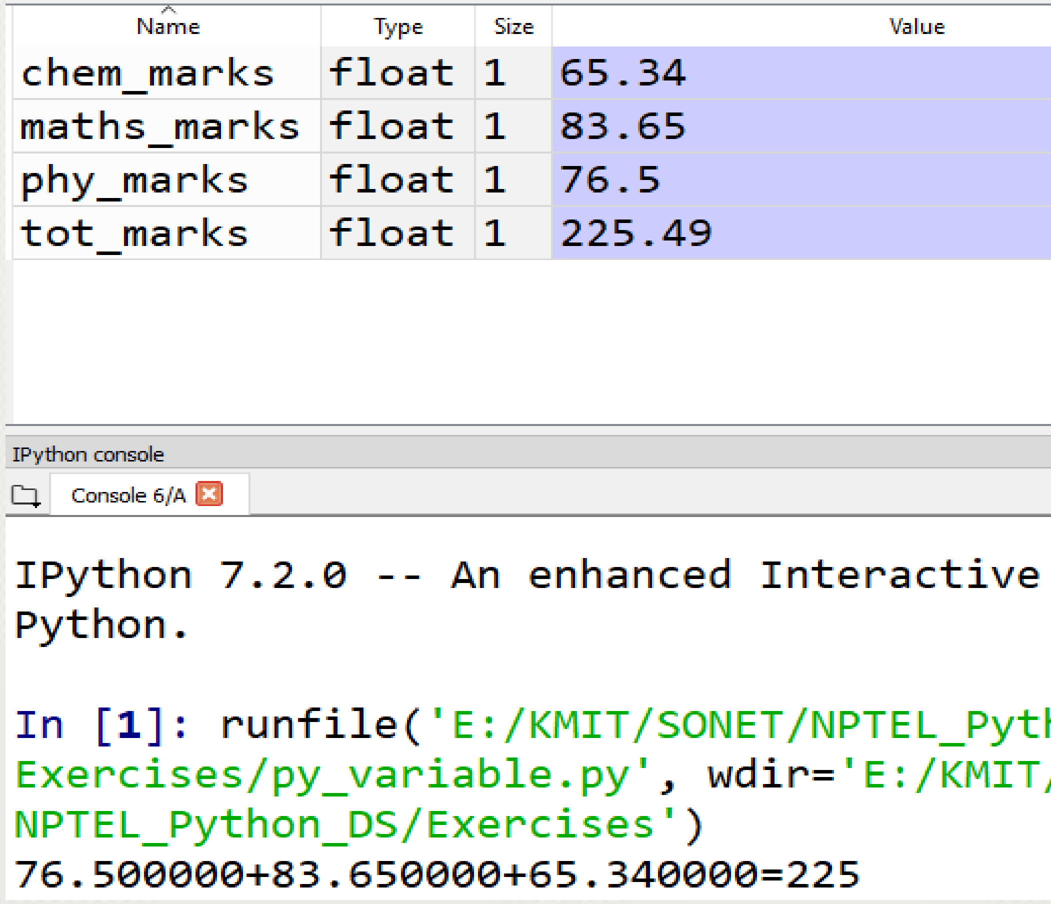
Formating the data

DATA SCIENCE



```
phy_marks=76.5
maths_marks=83.65
chem_marks=65.34
tot_marks=phy_marks+maths_marks+chem_marks
print("%f+%f+%f=%.f"%(phy_marks,maths_marks,chem_marks,tot_marks))
```

Output



Name	Type	Size	Value
chem_marks	float	1	65.34
maths_marks	float	1	83.65
phy_marks	float	1	76.5
tot_marks	float	1	225.49

IPython console

Console 6/A

```
IPython 7.2.0 -- An enhanced Interactive
Python.

In [1]: runfile('E:/KMIT/SONET/NPTEL_Pyt
Exercises/py_variable.py', wdir='E:/KMIT/
NPTEL_Python_DS/Exercises')
76.500000+83.650000+65.340000=225
```

Exercise-3

- 1.A=4
- B=6
- Perform addition operation and display Format the output as “3+4=7” using format specifier
- 2. f1=7.8
- f2=6.5
- Perform addition operation and display the result as “7.80+6.50=14.30”

Conclusion

You are aware of

Python Building blocks

Why Python and Python Trend

We will proceed with

Python Operators

Kmit SONET

DATA SCIENCE

THANK YOU