

ARTIFICIAL INTELLIGENCE WITH



By, Mohammad Tahir Mirji

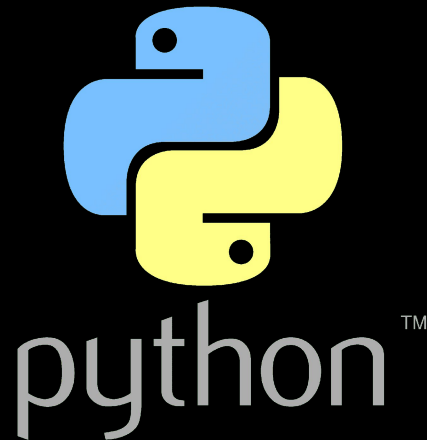
MTech CS

tahirmirji@github

Module 1

PYTHON FUNDAMENTALS

Part - 1



INTRODUCTION

INTRODUCTION

Python is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability.

The syntax in Python helps the programmers to do coding in fewer steps as compared to Java or C++.

Why we Program?

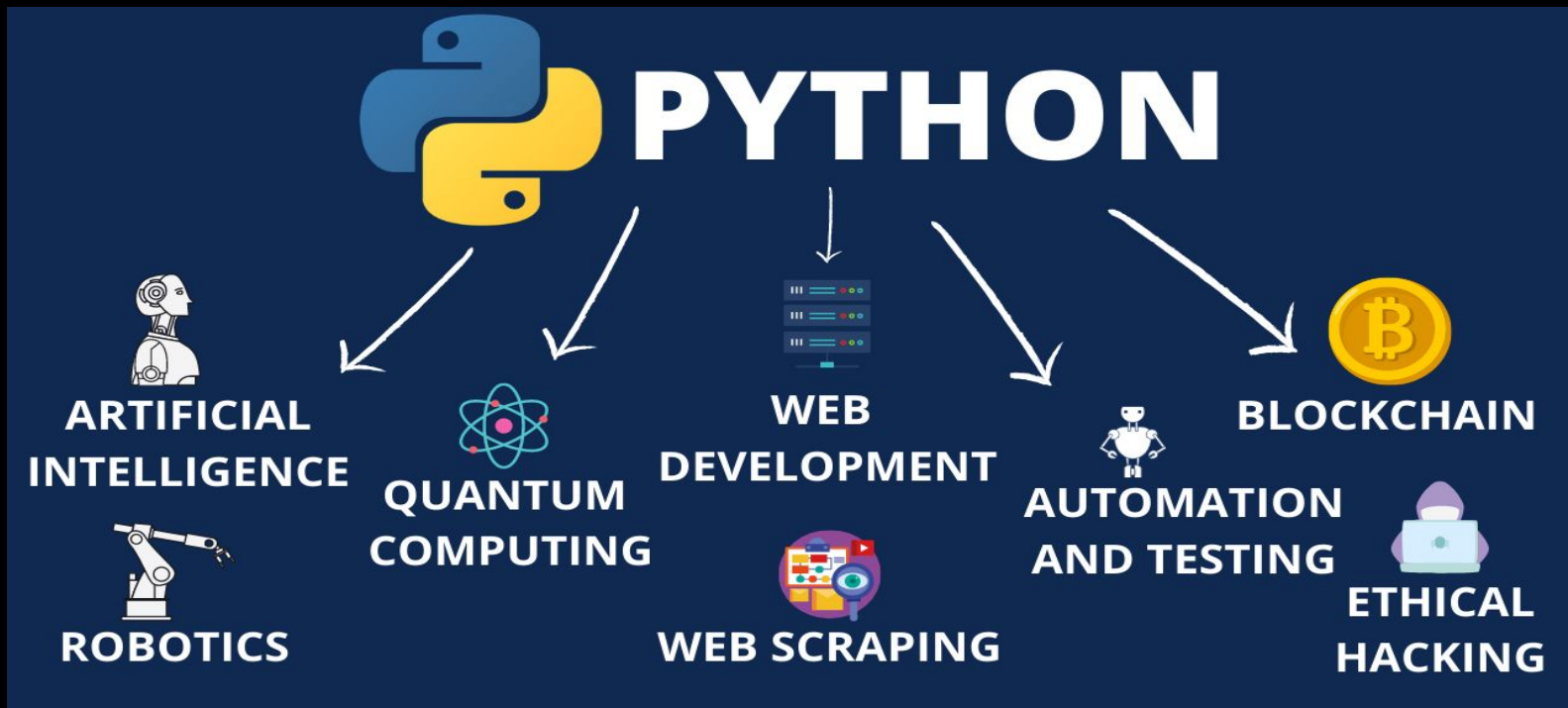
The reason that programming is so important is that it directs a computer to complete these commands over and over again, so people do not have to do the task repeatedly.

Instead, the software can do it automatically and accurately.

Why we Program in python?

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization.

Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.



Why python is so popular?



Advantages of Python

1. Presence of third-party modules
2. Extensive support libraries(NumPy for numerical calculations, Pandas for data analytics, etc.)
3. Open source and large active community base
4. Versatile, Easy to read, learn and write
5. User-friendly data structures
6. High-level language
7. Dynamically typed language(No need to mention data type based on the value assigned, it takes data type)
8. Object-Oriented and Procedural Programming language
9. Portable and Interactive
10. Ideal for prototypes – provide more functionality with less coding
11. Highly Efficient(Python's clean object-oriented design provides enhanced process control, and the language is
12. equipped with excellent text processing and integration capabilities, as well as its own unit testing framework,
13. which makes it more efficient.)
14. Internet of Things(IoT) Opportunities
15. Interpreted Language
16. Portable across Operating systems

Applications of Python

1. GUI-based desktop applications
2. Graphic design, image processing applications, Games, and Scientific/computational Applications
3. Web frameworks and applications
4. Enterprise and Business applications
5. Operating Systems
6. Education
7. Database Access
8. Language Development
9. Prototyping
10. Software Development
11. Data Science and Machine Learning
12. Scripting

How about Python Syntax?

The syntax of the Python programming language is the set of rules which defines how a Python program will be written.

Python Line Structure:

A Python program is divided into a number of logical lines and every logical line is terminated by the token NEWLINE.

Roadmap to Become AI Engineer?

AI is a field of software engineering for creating programming or machines that show human insight and is developing at a quick pace.

How to become an
**Artificial Intelligence
Engineer?**



Necessary Business Skills to Become Artificial Intelligence Engineer?

Innovative reasoning

It is very important for an AI engineer to possess innovative reasoning skills. Developing AI is all about thinking out the box, being creative for which innovative reasoning is a must.

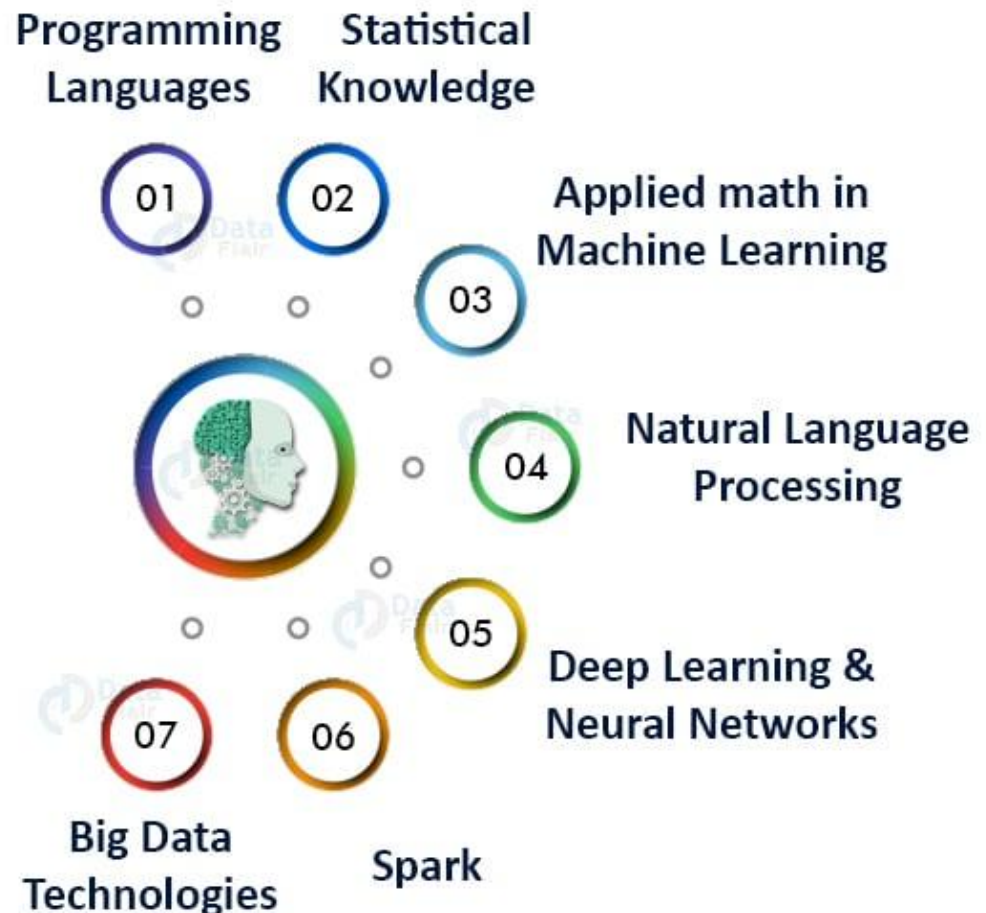
Problem Solving Skills

AI is a method to solve complex problems faced by humans. In order to develop such technology, problem-solving skills are highly important

Ability to work in a team

Teamwork is a virtue that can't be understated. In the technology industry as well, team workers are always preferred.

Technical Competency in becoming an AI Engineer



Steps to Be Machine Learning Engineer?

1. Learn Programming Language

To build a machine learning model, you should be familiar with programming languages. If you are a beginner in programming, then start with Python.

2. Brush-Up Your Math skills

Knowledge of mathematics is essential to understand, how machine learning and its algorithms work. You should know the basics of these math topics for machine learning-

- Probability and Statistics

- Linear Algebra

- Calculus

- Matrix

Knowledge of statistics will give you the ability to decide which algorithm is good for a certain problem.

3. Learn Machine Learning Concepts

You need to learn the basics of Machine Learning like- Types of Machine Learning algorithms(Supervised, Unsupervised, Semi-Supervised, Reinforcement Learning), then the detail of each Machine Learning algorithm, and other concepts.

Steps to Be Machine Learning Engineer?

4. Learn Data Science Tools

After gaining Python and Machine Learning knowledge, it's time to practice. And for that, you need to use Data Science tools like Jupyter and Anaconda. Spend your few hours and play with these tools. Understand what they're for and why you should use them.

5. Familiar with pandas, NumPy, and Matplotlib.

Now, it's time to know how to deal with data. Why? because in order to build a machine learning model, the first requirement is data. And for that, you need to have knowledge of data manipulation, analysis, and visualization.

6. Start Practicing with Real- World Projects

Now, you have enough skills to build your first Machine Learning Model. It's time to work on some machine learning projects. Projects are essential to get a job as a Machine Learning Engineer.

7. Gain Deep Learning Skills

Because Machine Learning works perfectly fine with small datasets. But, when you have large datasets, then Machine learning Algorithms fail. So for that Deep Learning is used. Deep Learning gives perfect results for large datasets.

What is Character Set?

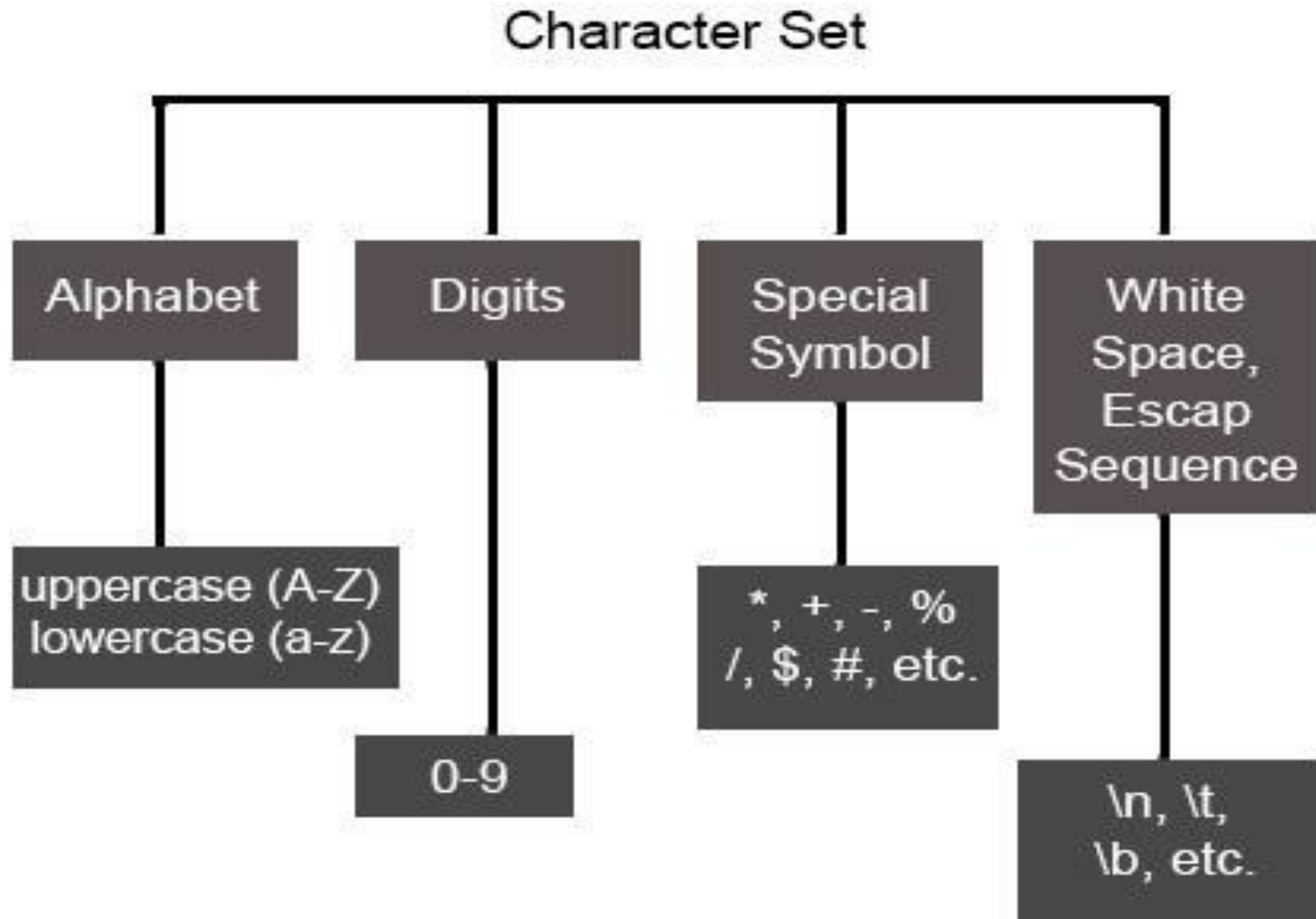
PYTHON CHARACTERSET

What is Character Set?

Character set is a bunch of identifying elements in the programming language.

PYTHON CHARACTERSET

PYTHON CHARACTERSET



What is Token or lexical unit?

TOKENS OR LEXICAL UNIT

What is Token?

Individual elements that are identified by programming language are called tokens or lexical unit.

TYPES OF LEXICAL UNITS

TOKENS / LEXICAL UNITS



What is Keyword or reserved word?

1. Keyword/Reserved Word

What is Keyword?

Keywords are also called as reserved words these are having special meaning in python language. The words are defined in the python interpreter hence these cant be used as programming identifiers.

Some Keywords of Python Language

Some Keywords of Python Language

and	assert
break	class
continue	def
del	elif
else	except
exec	finally
for	from

Some Keywords of Python Language

global	if
import	in
is	lambda
not	or
pass	print
raise	return
try	while
with	yield

What is an identifier?

2. IDENTIFIERS

What is an identifier?

A Python Identifier is a name given to a function, class, variable, module, or other objects that you'll be using in your Python program.

In short, its a name appeared in the program.

For example: a, b, c

a b and c are the identifiers and

a b & c and , are the tokens

PYTHON NAMING CONVENTIONS

OR

IDENTIFIER FORMATION RULES

PYTHON NAMING CONVENTIONS(1/4)

What are the python naming conventions?

- 1. An identifier can be a combination of uppercase letters, lowercase letters, underscores, and digits (0-9).**

Hence, the following are valid identifiers:

**myClass, my_variable, var_1, and
print_hello_world.**

PYTHON NAMING CONVENTIONS(2/4)

What are the python naming conventions?

2. The first character must be letter.
3. Special characters such as %, @, and \$ are not allowed within identifiers.
4. An identifier should not begin with a number.
Hence, 2variable is not valid, but variable2 is acceptable.

PYTHON NAMING CONVENTIONS(3/4)

What are the python naming conventions?

5. Python is a case-sensitive language and this behaviour extends to identifiers. Thus, Labour and labour are two distinct identifiers in Python.
6. You cannot use Python keywords as identifiers.
7. You cannot use Python keywords as identifiers.
8. You can use underscores to separate multiple words in your identifier.

PYTHON NAMING CONVENTIONS

SOME VALID IDENTIFIERS:

Myfile1 DATE9_7_8 y3m9d3
_xs
MYFILE _FXd

SOME INVALID IDENTIFIERS:

MY-REC 28dre break
elif false del

What are literals?

3. LITERALS / CONSTANT VALUES

What are literals?

Literals are also called as constants or constant values these are the values which never change during the execution of program.

What are the types of literals?

TYPES OF LITERALS / CONSTANT VALUES

What are the types of literals?

- 1) String Literals or Constants.**
- 2) Numeric Literals or Constants.**
- 3) Boolean Literals or Constants.**
- 4) Special Literal None.**
- 5) Literal Collections.**

What is string?

1. STRING LITERALS OR CONSTANTS

What is string?

Sequence of letters enclosed in quotes is called string or string literal or constant.

Python supports both form of quotes i.e.

`'Hello'`

`"Hello"`

Representation of String

REPRESENTATION OF STRING

```
>>> s = "Hello Python"
```

This is how Python would index the string:

Backward Indexing

-12	-11	-10	-9	-8	-6	-6	-5	-4	-3	-2	-1
H	e	l	l	o		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11

Forward Indexing

REPRESENTATION OF STRING

-12	-11	-10	-9	-8	-6	-6	-5	-4	-3	-2	-1
H	e	l	l	o		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11

To access the first character on the string you just created, type and enter the variable name `s` and the index `0` within square brackets like this:

```
>>>s[0]
```

You'll get this output:

`'H'`

REPRESENTATION OF STRING

-12	-11	-10	-9	-8	-6	-6	-5	-4	-3	-2	-1
H	e	l	l	o		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11

To access the last character, you can use this expression:

```
>>>s[len(s)-1]
```

You'll get the output:
'n'

Len() function
is used to find
the length of
the string.

REPRESENTATION OF STRING

-12	-11	-10	-9	-8	-6	-6	-5	-4	-3	-2	-1
H	e	l	l	o		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11

The expression introduces you to the *len function*.
There is actually an easier way to
access the last item on the string:

```
>>>s[-1]
```

```
'n'
```

To access the penultimate character:

```
>>>s[-2]
```

```
'o'
```

TYPES OF STRINGS

What are the types of strings supported in python?

Python supports two ways of representation of strings:

- 1) Single Line Strings.**
- 2) Multi Line Strings.**

TYPES OF STRINGS

SINGLE LINE STRINGS

Strings created using single quote or double quote must end in one line are called single line strings.

For Example:

Item="Computer"

Or

Item= 'Computer'

MULTI LINE STRINGS

Strings created using single quote or double quote and spread across multiple lines are called **Multi Line Strings**.

by adding **backslash ** one can continue to type on next line.

For instance: **Item = 'Key**
 board'

SIZE OF STRINGS

SIZE OF STRINGS

'\\' Size is 1 (\ is an
escape sequence)

'abc' size is 3

"\ab" size is 2

"Raama\'s Laptop" size is 13

Strings with Triple Quotes

STRINGS WITH TRIPLE QUOTES

For multi line strings created by triple quotes, while calculating size, the EOL(End of Line) character at the end of line is also counted.

For instance:

Str2=""x
y
z""

y
z"' with red arrows pointing to the end of each line, indicating that the End of Line (EOL) character is counted in the string length." data-bbox="168 695 239 868"/>

Enter keys are considered as EOL so size of str2 is 5

2. NUMERICAL LITERALS

Numerical Literals have the following types:

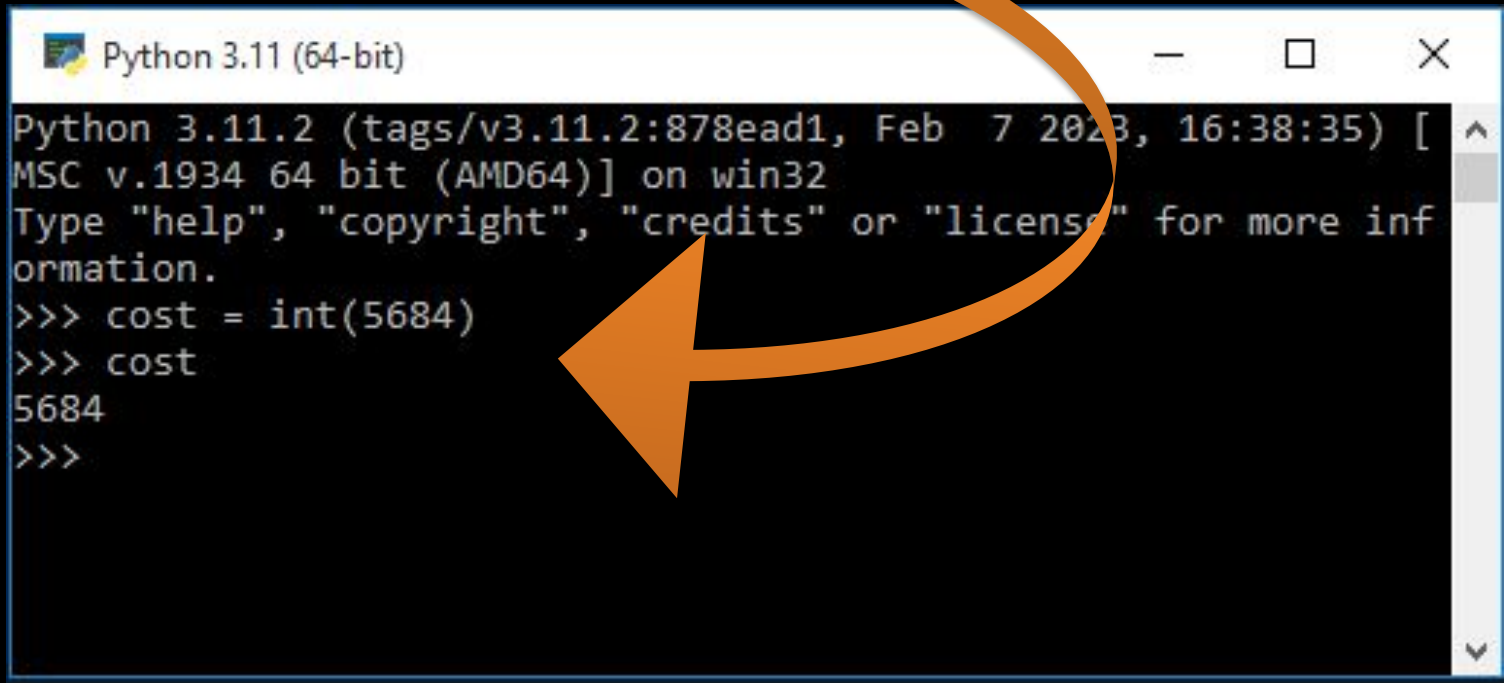
int or integers - Whole numbers

float - real values

Complex - Complex numbers

INTEGER LITERALS OR CONSTANTS

- ◆ **Decimal Integer Literals:** Any whole number (+ve) or (-ve).



```
Python 3.11 (64-bit)
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [
MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more inf
ormation.
>>> cost = int(5684)
>>> cost
5684
>>>
```

INTEGER LITERALS OR CONSTANTS

- ❖ **Octal Integer Literals(base 8):** A Sequence of digits starting with 0O (digit zero followed by letter o) is taken to be an Octal Integer **Literals**.



```
Python 3.11 (64-bit)
>>> a=007
>>> a
7
>>>
```

A screenshot of a Python 3.11 (64-bit) command prompt window. The window title bar shows the Python logo and text. The command prompt shows the assignment of the variable 'a' with the value '007', followed by the command to print 'a', which outputs '7'. A large orange curved arrow points from the text '00' in the code 'a=007' to the output '7', illustrating that the leading zeros are ignored in octal literals.

INTEGER LITERALS OR CONSTANTS

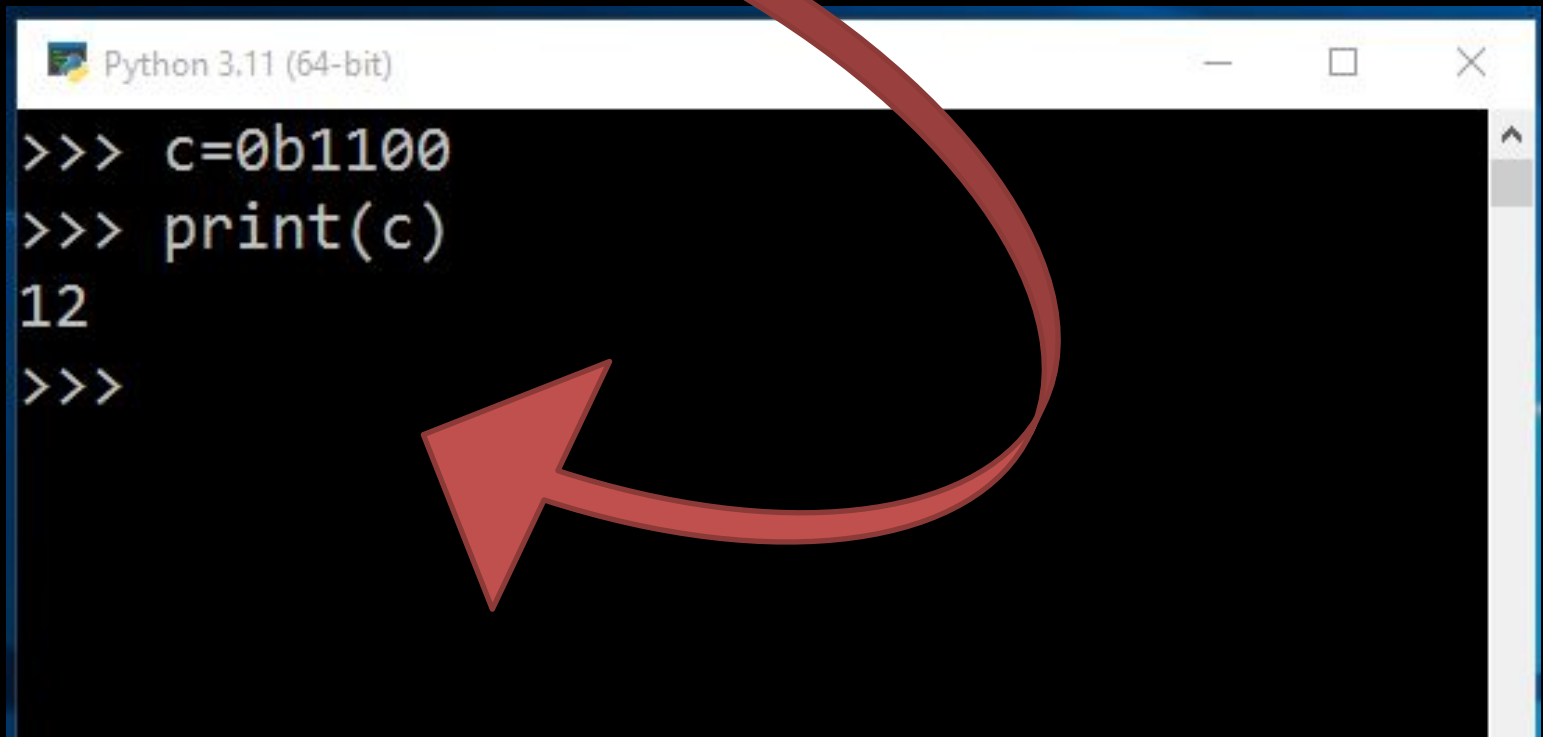
- ◆ **Hexadecimal Integer Literals (base 16):** Sequence of digits preceded by `0x` or `0X` is hexadecimal integer literals



```
Select Python 3.11 (64-bit)
>>> a=007
>>> a
7
>>> val = 0xA0C
>>> val
2572
>>>
```

INTEGER LITERALS OR CONSTANTS

◆ **Binary literals (base 2):** To signify binary literals, you'll use the prefix '0B' or '0b' (zero and uppercase or lowercase 'b').



```
Python 3.11 (64-bit)
>>> c=0b1100
>>> print(c)
12
>>>
```

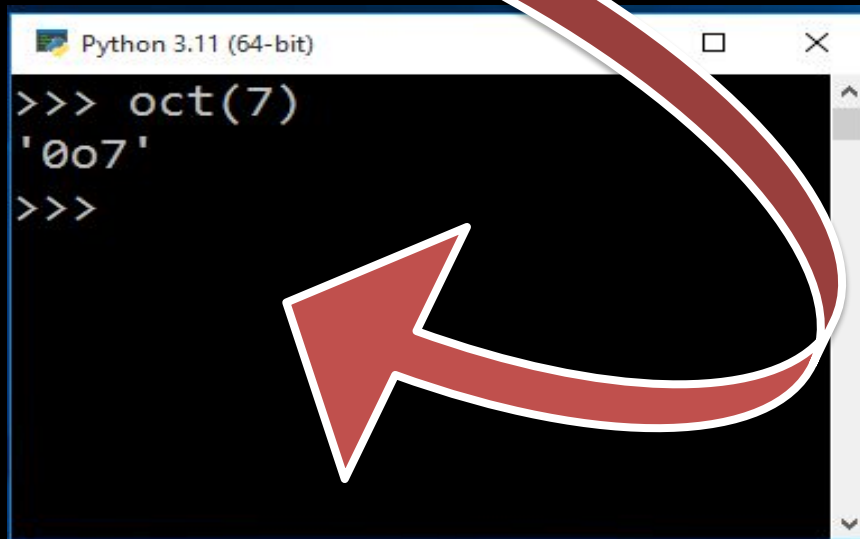
A screenshot of a Python 3.11 (64-bit) terminal window. The window has a title bar with the text "Python 3.11 (64-bit)" and standard window controls (minimize, maximize, close). The terminal content shows a series of commands and their output: the first command is `c=0b1100`, the second is `print(c)`, which outputs `12`, and the prompt `>>>` is shown again. A large, thick, red curved arrow originates from the text "Binary literals" in the paragraph above and points directly to the `0b1100` literal in the code.

Converting Integers to their String Representation

oct ()

To convert an integer into its string representation, you can use the functions *hex()*, *bin()*, and *oct()*.

To convert the integer 7 to its octal literal, type and enter `oct(7)` on the command prompt. You'll get the output `'0o7'`:

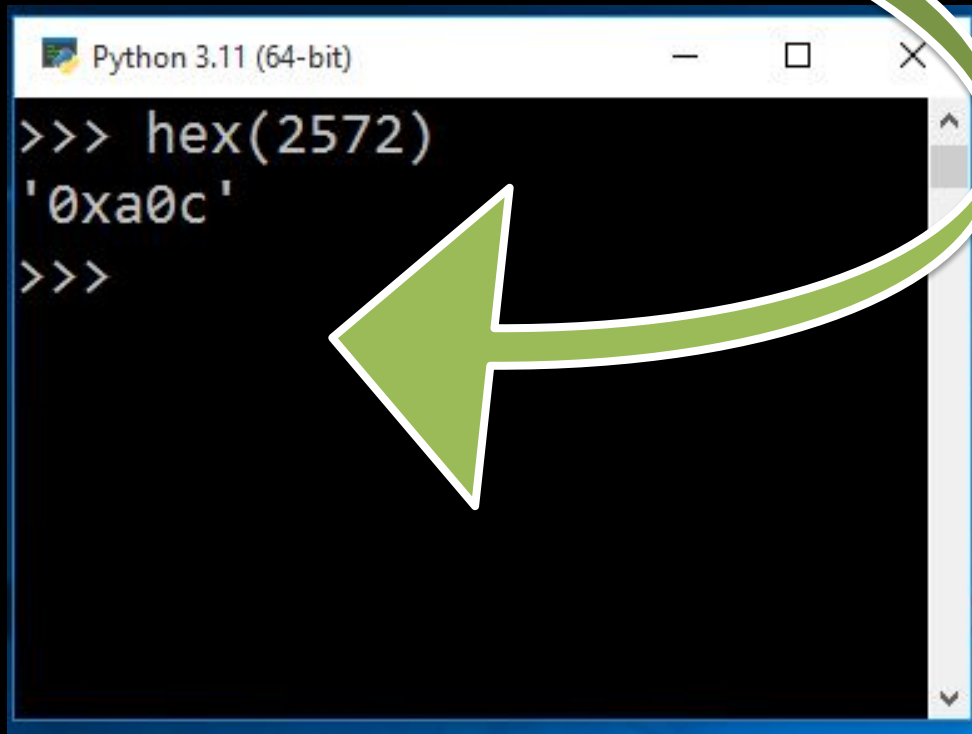


```
Python 3.11 (64-bit)
>>> oct(7)
'0o7'
>>>
```

A screenshot of a Python 3.11 (64-bit) command prompt window. The window title bar is white with a blue border. The background is black. The text is white. The prompt shows the command `oct(7)` being entered and the output `'0o7'` being displayed. A large red arrow with a white outline points from the text '0o7' in the paragraph above to the output in the screenshot.

hex ()

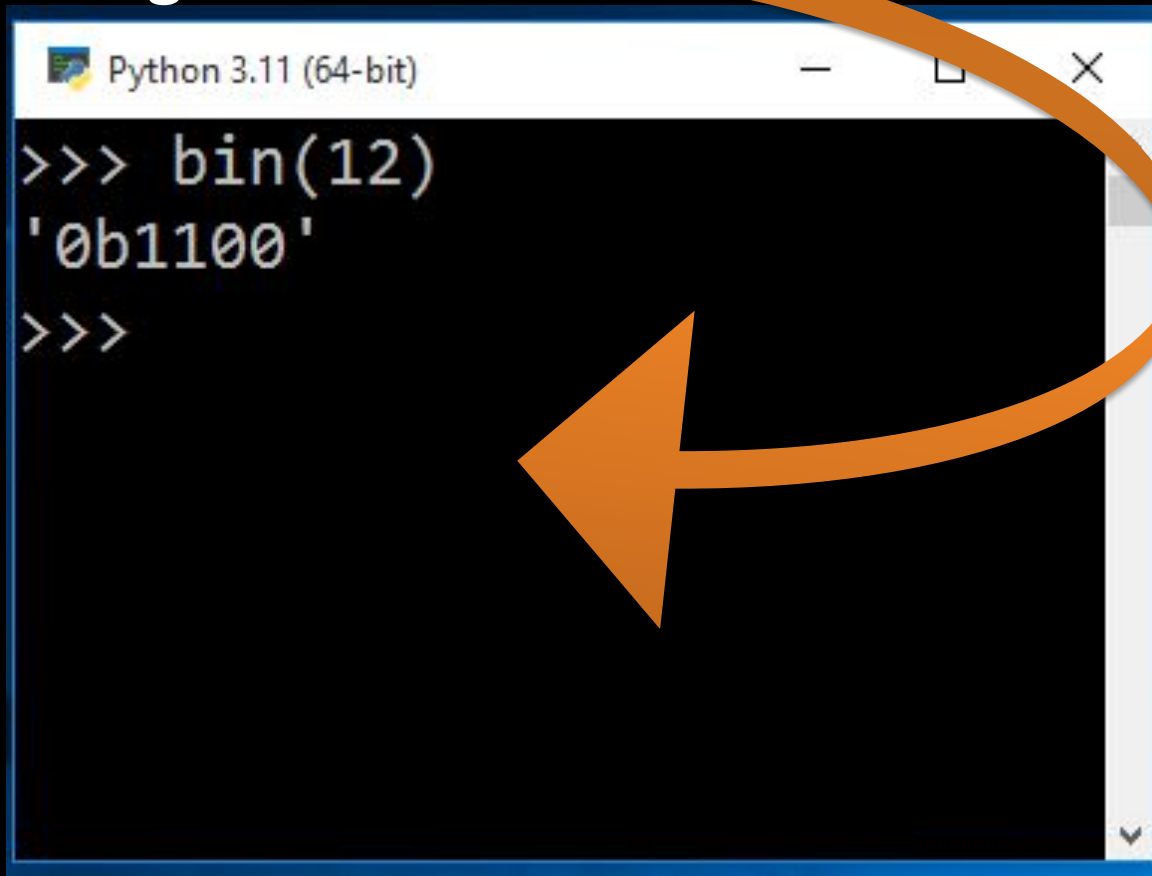
Here is what happens when you convert the integer 2572 to a hexadecimal literal:



```
Python 3.11 (64-bit)
>>> hex(2572)
'0xa0c'
>>>
```

bin ()

see what happens when you use the bin() function to convert the integer 12 to its binary string:



```
Python 3.11 (64-bit)
>>> bin(12)
'0b1100'
>>>
```

FLOATING POINT LITERALS OR CONSTANTS

FLOATING POINT LITERALS OR CONSTANTS

Floating point literals are also called as real literals having fractional part.

These may be written in one of the two forms:

1. Fractional Form: for example 15.75

Exponent Form: It consists of two parts **Mantissa** and **Exponent**.

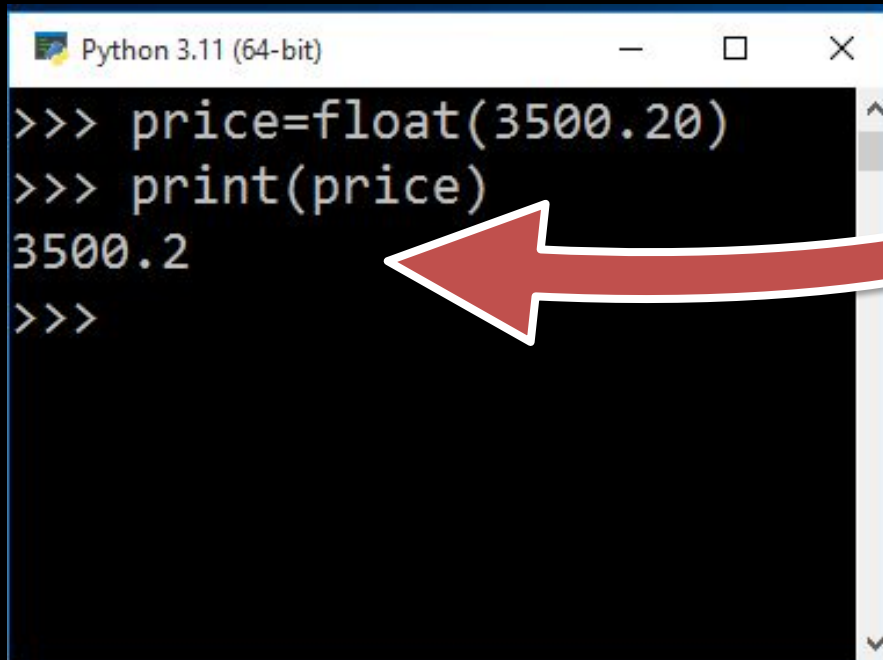
for example 0.000000123 can be represented as

$$1.23 \times 10^{-7} = 1.23\text{E-}7.$$

where **mantissa part is 1.23** and **E-7 is the exponent**.

FLOATING POINT LITERALS OR CONSTANTS

Float type



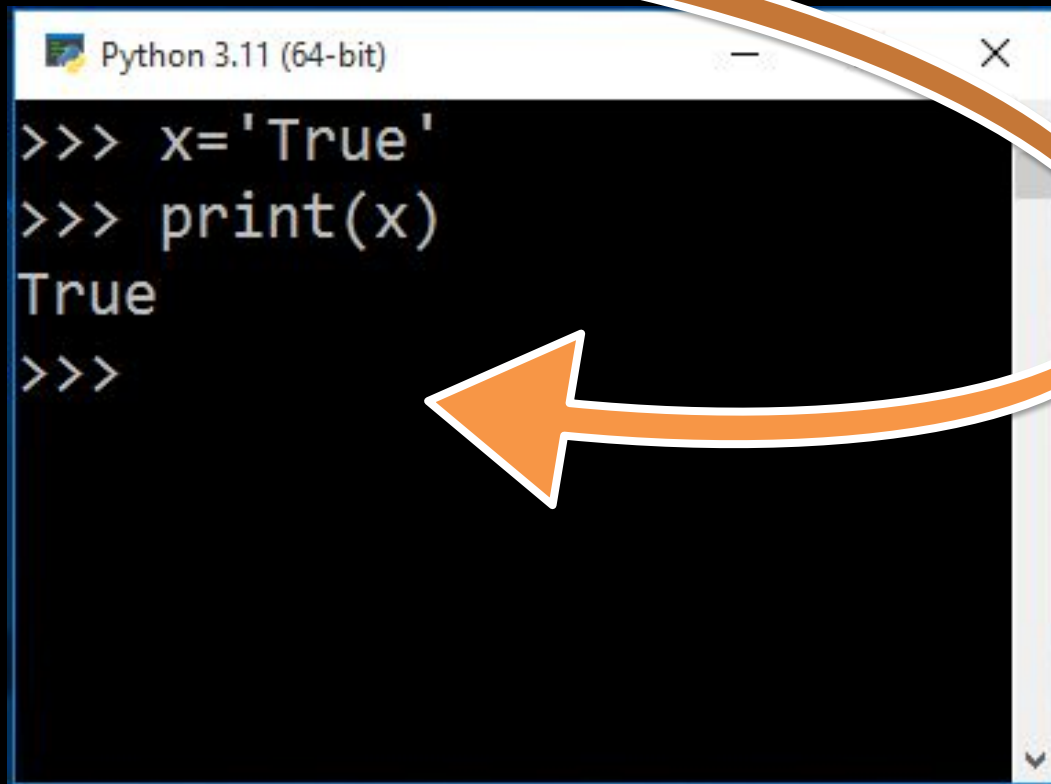
```
>>> price=float(3500.20)
>>> print(price)
3500.2
>>>
```

A screenshot of a Python 3.11 (64-bit) terminal window. The window title bar shows the Python logo and text. The terminal content shows a sequence of commands: `price=float(3500.20)`, `print(price)`, and the output `3500.2`. A large red curved arrow originates from the text 'Float type' and points to the output `3500.2`.

BOOLEAN LITERALS OR CONSTANTS.

3) BOOLEAN LITERALS OR CONSTANTS.

A Boolean literal in python is used to represent the Boolean values (true or false).



```
>>> x='True'
>>> print(x)
True
>>>
```

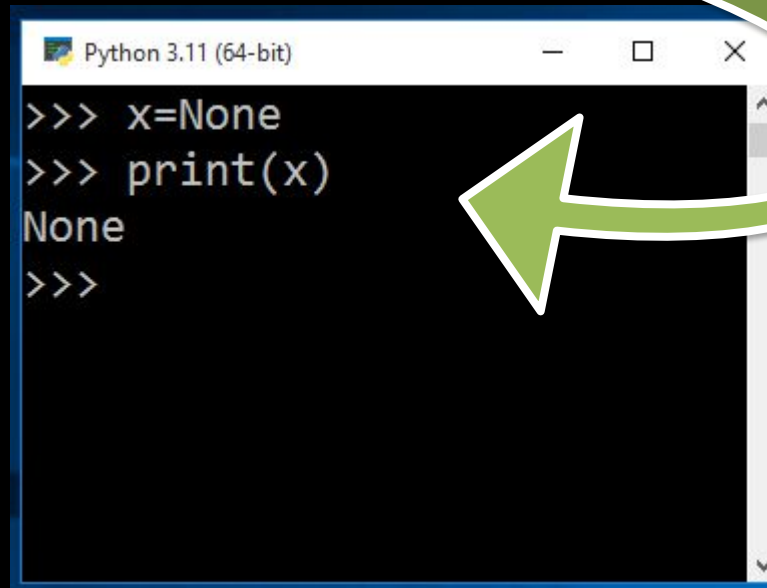
A screenshot of a Python 3.11 (64-bit) terminal window. The window title bar shows the Python logo and the text 'Python 3.11 (64-bit)'. The terminal content shows a series of commands and their output: the first command is `>>> x='True'`, the second is `>>> print(x)`, and the output is `True`. The prompt `>>>` appears again on the next line. A large, thick, orange curved arrow with a white outline starts from the top right of the terminal window and points towards the text 'Boolean values (true or false)' in the paragraph above.

Special Literal - None

4) SPECIAL LITERAL NONE

The **None** literal is used to indicate absence of value.

For example: `val = None`



```
Python 3.11 (64-bit)
>>> x=None
>>> print(x)
None
>>>
```

A screenshot of a Python 3.11 (64-bit) terminal window. The window title bar shows the Python logo and text 'Python 3.11 (64-bit)'. The terminal content shows a prompt '>>>' followed by 'x=None', another prompt '>>>' followed by 'print(x)', the output 'None', and a final prompt '>>>'. A large green arrow with a white outline points from the text 'val = None' in the preceding text block to the 'x=None' line in the terminal.

LITERAL COLLECTIONS

5) LITERAL COLLECTIONS

Python supports literal collections also such as tuple and lists ..etc

It will be too complex to discuss as we are in the beginning, subsequent chapters we will cover literal collections.

4. OPERATORS

OPERATORS

What is an operator?

In Python, an operator is a symbol or a keyword that performs some kind of operation on one or more values, producing a result. Python supports a wide range of operators, including arithmetic, comparison, assignment, logical, bitwise, membership, and identity operators.

Types of Operators

Arithmetic operators: +, -, *, /, %, ** (exponentiation), // (floor division)

Comparison operators: == (equal to), != (not equal to), < (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to)

Assignment operators: =, +=, -=, *=, /=, %=, **=, //= (assigns the result of an arithmetic operation to a variable)

Logical operators: and, or, not (used for combining and negating Boolean values)

Bitwise operators: &, |, ^, ~ (bitwise negation), << (left shift), >> (right shift)

Membership operators: in, not in (used to test if a value is a member of a sequence, such as a list or a string)

Identity operators: is, is not (used to test if two objects are the same object in memory)

5. PUNCTUATORS

PUNCTUATORS

Punctuators are also called as separators

The Followings are used as punctuators:

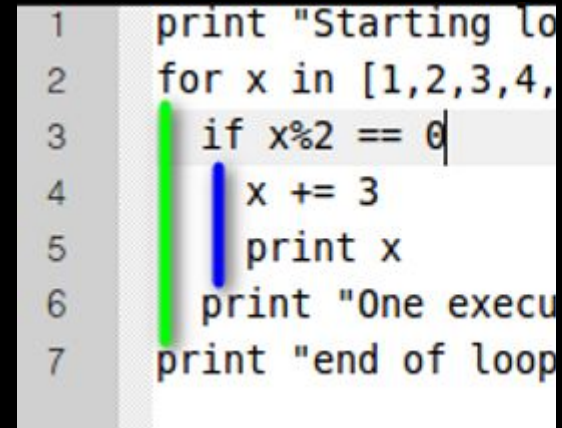
Brackets	[]
Parentheses	()
Braces	{ }
Comma	,
Semicolon	;
Colon	:
Asterisk	*
Ellipsis	...
Equal Sign	=
Pound Sign/Hash	#

WHITE SPACE

WHITE SPACE

Python uses four spaces as default indentation spaces.

- Use consistent indentation instead.
- The first line with less indentation is outside of the block.
- The first line with more indentation starts a nested block.
- Often a colon appears at the start of a new block.
(E.g. for function and class definitions.).

A screenshot of a code editor showing Python code with indentation. The code is as follows:

```
1 print "Starting loop"
2 for x in [1,2,3,4,5]:
3     if x%2 == 0:
4         x += 3
5         print x
6     print "One execution"
7 print "end of loop"
```

A green vertical bar highlights the indentation of the loop body (lines 3-6). A blue vertical bar highlights the indentation of the if statement body (lines 4-5).

Note:

However, the number of spaces can be anything; it is up to the user.

But a minimum of one space is needed to indent a statement. The first line of python code cannot have an indentation.

COMMENTS

COMMENTS

Comments are non executable statements in a program.

Single line comment always starts with #

Multiline comment will be in triple quotes.

For example: “” write a program to find the simple interest “”.

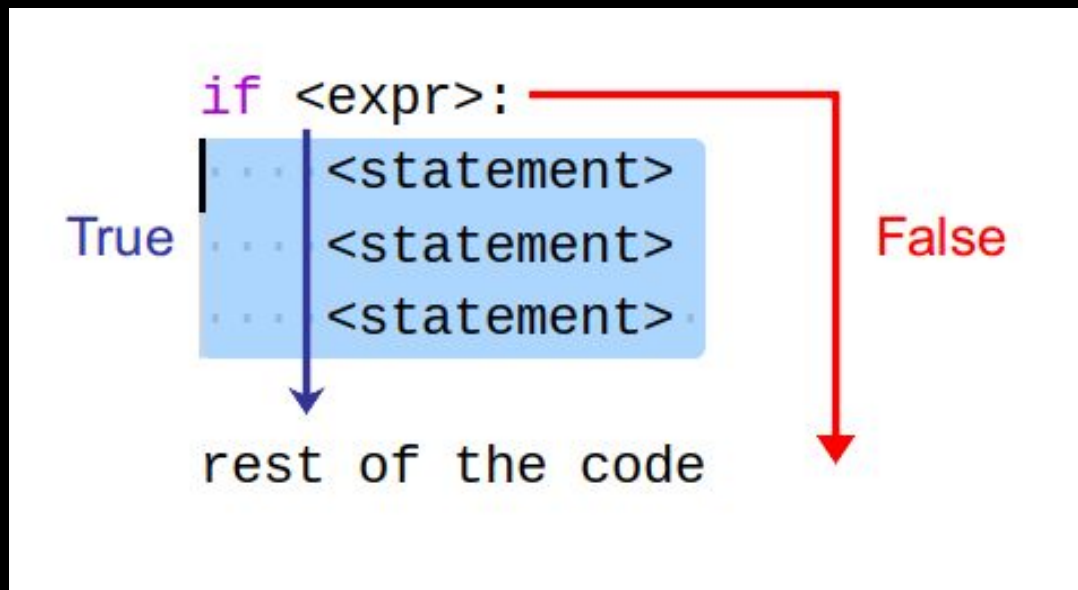
Note: Triple apostrophe is called docstrings.

STATEMENTS

STATEMENTS

In computer terminology statement refers to an instruction.

Program contains several statements. A collection of statements makes program . Another name for a program is code.



FUNCTIONS

FUNCTIONS

What is function?

Function is a self contained program segment which carries out some specific well defined task.

For Example:

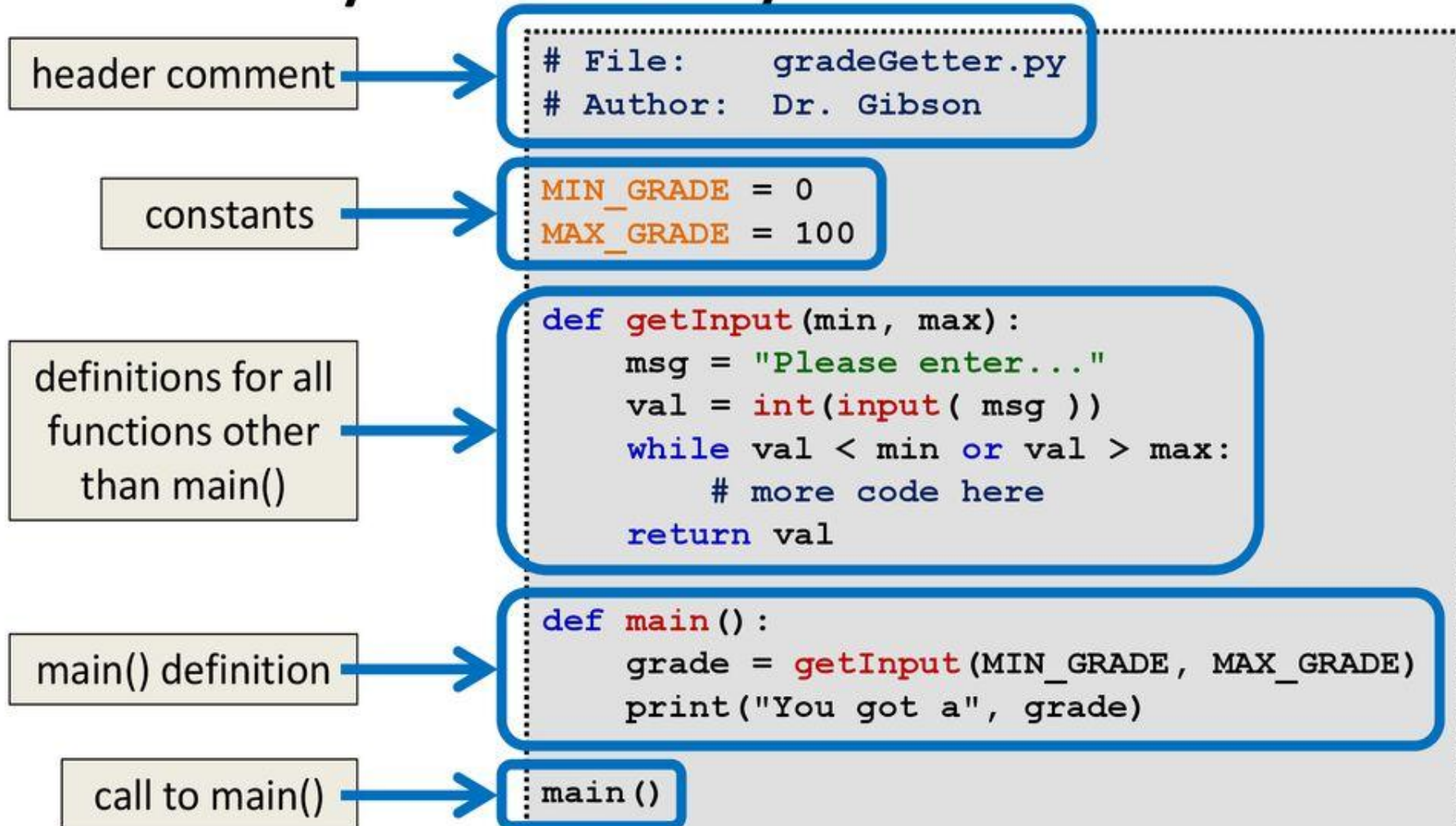
```
def sqr( num ):  
    result= num *num  
    print ("Square = " , result)  
sqr(2)
```

Output:

Square = 4

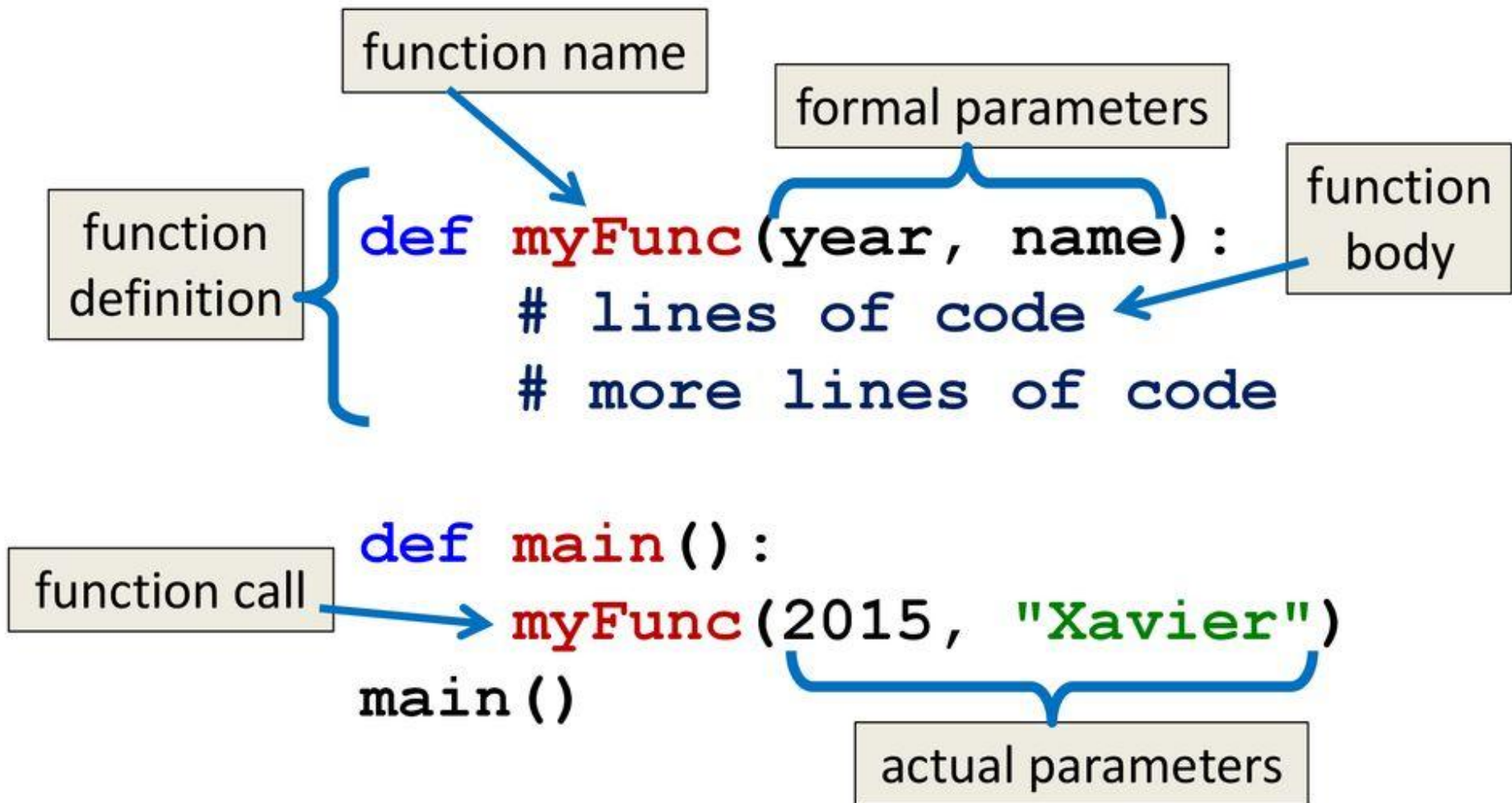
FUNCTIONS

Layout of a Python File



FUNCTIONS

Function Vocabulary



PYTHON PROGRAMMING CONVENTIONS

PYTHON PROGRAMMING CONVENTIONS

Statement Termination: python does not use any symbol to terminate the statement.

Maximum Line Length: Line Length be maximum 79 characters.

Whitespaces: you should always have whitespace around operators but not with parenthesis.

PYTHON PROGRAMMING CONVENTIONS

Block or Code Block:

A group of statements which are part of another statement or function is called Block or Code Block.

Case Sensitive:

Python is case sensitive.

Python Blocks

Block 1

Block 2

Block 3

Block 2 - Continuation

Block 1 - Continuation

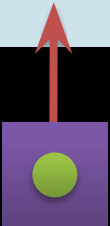
VARIABLES AND ASSIGNMENTS

VARIABLES AND ASSIGNMENTS

Named labels are called **variables**.

For example: **marks = 86**

78	79	80	81	82	83	84	85	86	87
2000	2016	2018	2026	2032	2044	2048	2050	2054	2068



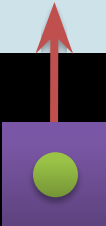
A diagram illustrating memory allocation. A table shows memory addresses (2000 to 2068) and their corresponding values (78 to 87). A red arrow points from a green circle (representing the variable 'marks') to the value 86 at memory location 2054.

marks refers to location **2054**

VARIABLES AND ASSIGNMENTS

Now marks = 81

78	79	80	81	82	83	84	85	86	87
2000	2016	2018	2026	2032	2044	2048	2050	2054	2068



marks refers to
location 2026

Note: Variables in python do not have fixed locations unlike other programming languages

VARIABLES AND ASSIGNMENTS

Lvalues & Rvalues:

Lvalue: Expressions that is on LHS (Left Hand Side) is called Lvalue.

Rvalue: Expressions that is on RHS (Right Hand Side) is called Rvalue.

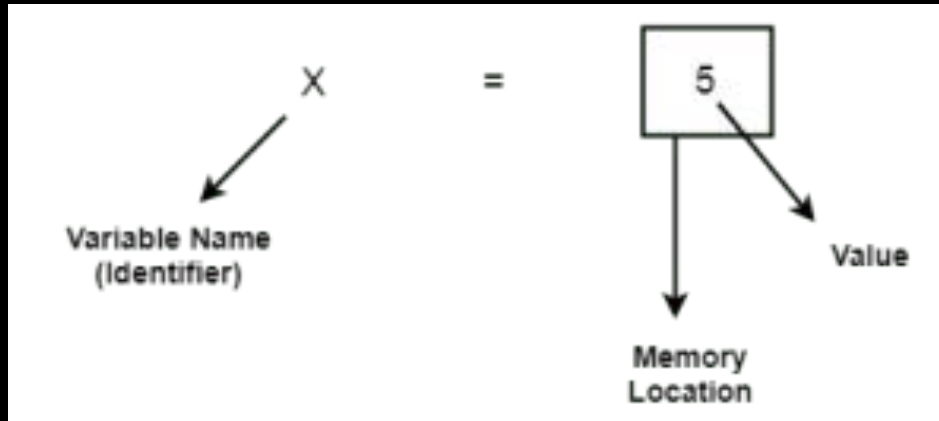
VARIABLES AND ASSIGNMENTS

Multiple Assignments

Python is very versatile with assignment statements.

1. Assigning same value to multiple variables:

a=b=c=d=e=5



VARIABLES AND ASSIGNMENTS

Multiple Assignments

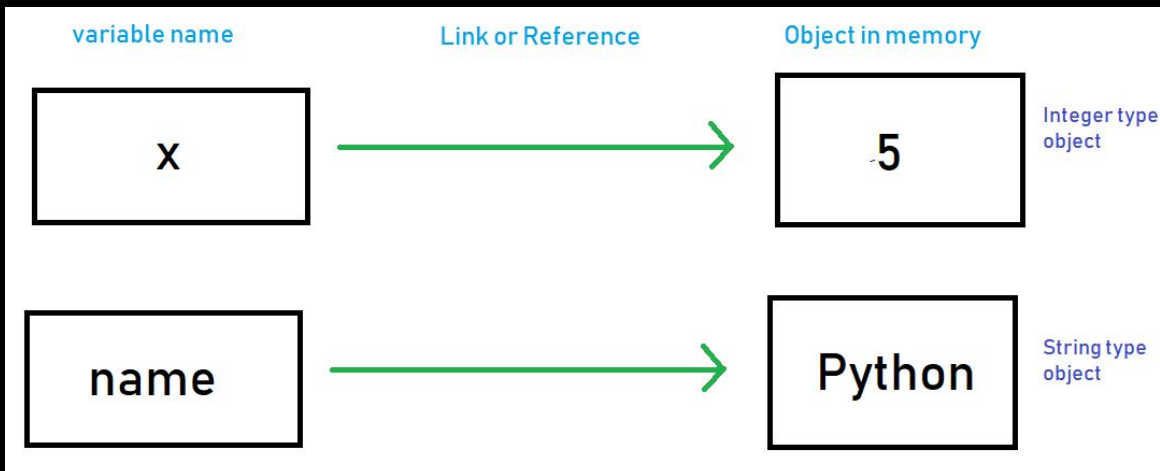
2. Assigning Multiple values to multiple variables:

`p,q,r =5,10,15`

`print(q, r)` will print 10 15

`p,q=q,p`

`print (p,q)` will print 10 5



VARIABLES AND ASSIGNMENTS

Multiple Assignments

2. Assigning Multiple values to multiple variables:

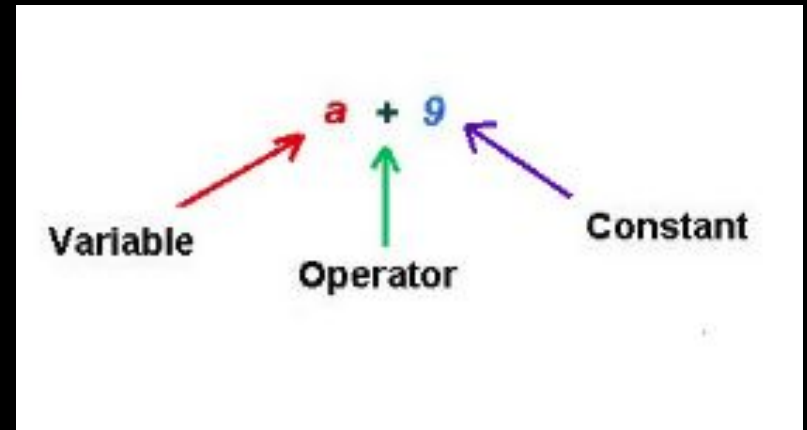
`a,b,c = 5,10,7`

`b,c,a = a+1, b+2, c-1`

`print(a,b,c)` will print 6 6 12

Now,

`X=10`



VARIABLES AND ASSIGNMENTS

Multiple Assignments

Expressions separated by commas are evaluated from left to right.

Now,

```
x = 10
```

```
y,y = x+2,x+5
```

```
y,y = 12,15
```

First It will assign $y = 12$ then $y = 15$

So `print(y)` will print 15

VARIABLES AND ASSIGNMENTS

Dynamic Typing:

A variable pointing to a value of certain type can be made to point to a value/object of different type this is called **Dynamic Typing**.

x=10

print(x)

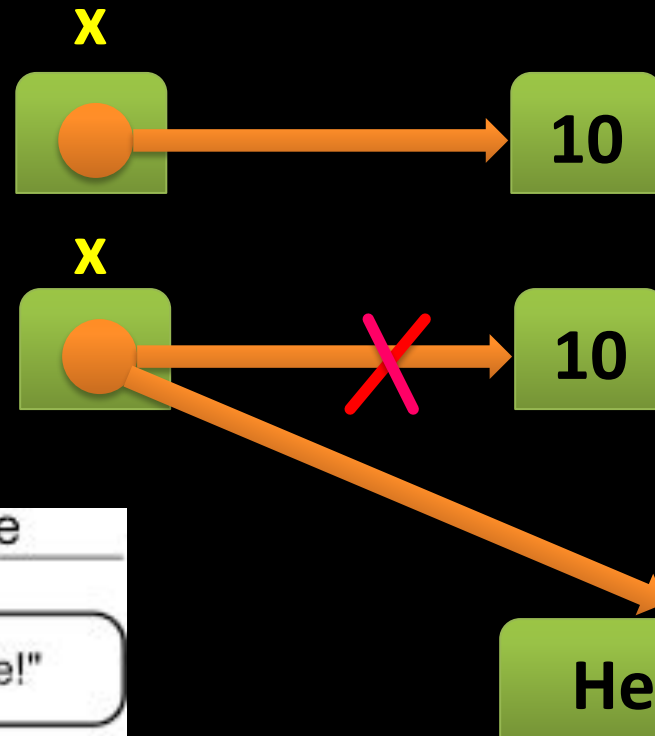
x=" Hello World"

print(x)

		Strong	Weak
Dynamic		Python Clojure Ruby Erlang	PHP Perl JavaScript VB
		C# Java Scala Haskel	C C++

VARIABLES AND ASSIGNMENTS

Output will be
10
Hello World



Variable	Value
message	"Hi there!"
n	17
pi	3.141592653

VARIABLES AND ASSIGNMENTS

Caution with Dynamic Typing:

`x = 'day'`

`y = x/2`

Error! String can not be divided.



VARIABLES AND ASSIGNMENTS

type() function:

To know the data type of a value which is pointing
use type ()

```
>>>a=10
```

```
>>>type(a)
```

```
<class 'int'>
```

Type returned as integer



```
>>>a=20.4
```

```
>>>type(a)
```

```
<class 'float'>
```

Type returned as float



VARIABLES AND ASSIGNMENTS

type() function:

To know the data type of a value which is pointing
use type ()

```
>>>a="Hello"
```

```
>>>type(a)
```

```
<class 'str'>
```

Type returned as string



INPUT () FUNCTION

INPUT () FUNCTION

Input() Function is a built in function of python used to read values from the user

The general format or syntax of the input() is:

Val = input(message)

Where , message is a string

For Example:

Where,

Val is a variable which is the label for a memory location where the value is stored.

INPUT () FUNCTION

For Example:

```
p = input("Enter the value")
```

By default input receives string formatted input

```
x = int(input("Enter x value"))
```

Reads the value and converts it in to integer type data or value.

```
y=float(input("Enter y value"))
```

Reads the value and converts it in to float type data or value.

INPUT () FUNCTION

int () and float () Functions:

Python offers two functions to be used with input() to convert the received values:

Example 1: >>age = int(input("Enter age"))

Example 2: >>sal=float(input("Enter salary"))

PRINT () FUNCTION

PRINT () FUNCTION

print() Function is a built in function of python used to display the values on the screen.

The general format or syntax of the input() is:

```
print(*objects, sep=' ', end='\n', file=sys.stdout,  
flush=False)
```

The print function can print an arbitrary number of values ("value1, value2, ..."), which are separated by commas.

These values are separated by blanks. In the following example we can see two print calls. We are printing two values in both cases, i.e. a string and a float number.

PRINT () FUNCTION

Let's discuss print() Parameters:

objects - object to be printed. * indicates that there may be more than one object

sep - objects are separated by sep. Default value: ' '

end - end is printed at last

file - must be an object with write(string) method. If omitted it, sys.stdout will be used which prints objects on the screen.

flush - If True, the stream is forcibly flushed. Default value: False

PRINT () FUNCTION

Example 1: How print() works in Python?

```
print("Hello World.")
```

```
a = 5
```

#Two objects are passed:

```
print("a =", a)
```

```
b = a
```

Three objects passed:

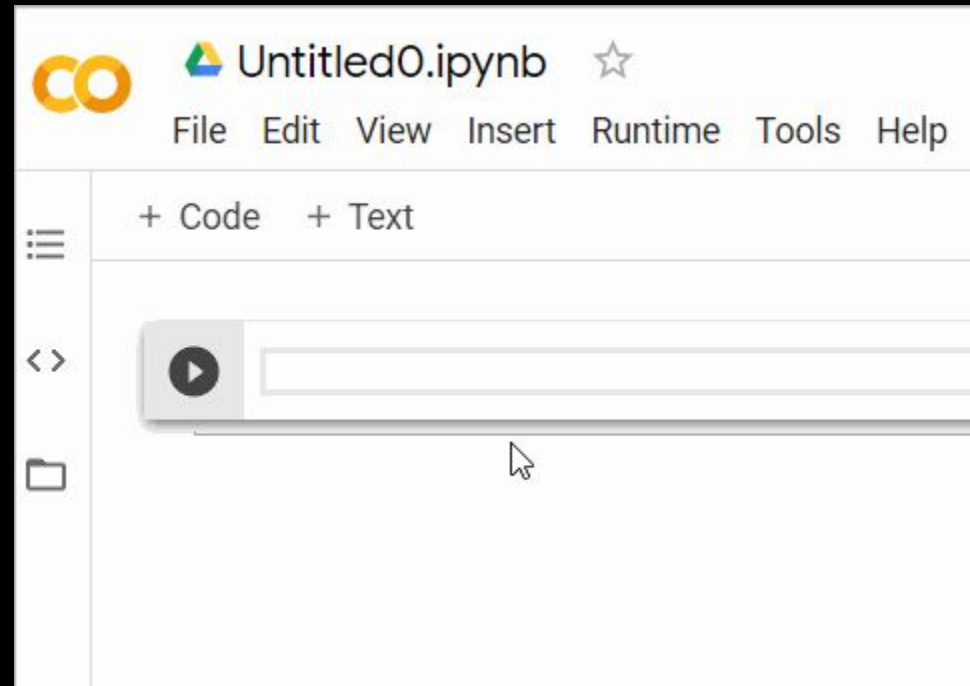
```
print('a =', a, '= b')
```

Output

Python is fun.

```
a = 5
```

```
a = 5 = b
```



PRINT () FUNCTION

Example 2: How print() works in Python?

```
>>> print("a = ", a)
```

```
a = 3.564
```

```
>>> print("a = \n", a)
```

```
a =
```

```
3.564
```

```
>>>
```

Any Questions Please



Sample Test Questions On PYTHON FUNDAMENTALS

PYTHON FUNDAMENTALS

Each carries 2 Marks Questions

(10 x 2 = 20)

- 1. What is EOL?**
- 2. What is an escape sequence?**
- 3. What is the maximum line length in a python program?**
- 4. Write any four keywords of python language**
- 5. What are the types of Assignment statements?
Explain**
- 6. Explain with a diagram how a variable refers to a memory location?**

PYTHON FUNDAMENTALS

Each carries 2 Marks Questions

(10 x 2 = 20)

- 7. What is Dynamic typing?**
- 8. Write any four python naming conventions**
- 9. What is input () function? Write down the general format of input () function and explain with proper example.**
- 10. What is print () function? Write down the general format of print () function and explain with proper example.**

QUESTION BANK

PYTHON FUNDAMENTALS

PYTHON FUNDAMENTALS

One Mark Questions

1. What is character set?
2. What is token?
3. List the types of tokens
4. What is keyword?
5. What is an identifier? Give suitable example.
6. What is a literal?
7. What is string?
8. What is single line string?
9. What is multi line string?
10. What is EOL?
11. What is an escape sequence?
12. What is Boolean literal?

PYTHON FUNDAMENTALS

One Mark Questions

13. What is none?
14. What is an operator?
15. What is Unary Operator?
16. What is Binary Operator?
17. List the shift operators
18. List the Bitwise operators
19. What is an assignment statement?
20. What is Punctuators?
21. What is comment?
22. What is whitespace?

PYTHON FUNDAMENTALS

One Mark Questions

- 23. What is statement?**
- 24. Weather python uses statement termination? Justify your answer**
- 25. What is the maximum line length in a python program?**
- 26. What is Block?**
- 27. What is Code Block?**
- 28. What is Code?**
- 29. What do you mean by case sensitive language?**

PYTHON FUNDAMENTALS

One Mark Questions

- 30. What is variable?**
- 31. What is Lvalue?**
- 32. What is Rvalue?**
- 33. What is an Assignment statement?**
- 34. What is Dynamic typing?**

PYTHON FUNDAMENTALS

Two Marks Questions

- 1. Explain the character set of python**
- 2. What are the types of tokens supported in python language?**
- 3. Write any four keywords of python language**
- 4. What are the types of literals?**
- 5. Explain Boolean literals**
- 6. What are relational operators?**
- 7. What are the types of Assignment statements? Explain**
- 8. What is General Structure or General format or Syntax?**

PYTHON FUNDAMENTALS

Two Marks Questions

- 9. What are the types of comments? Explain with suitable examples**
- 10. Explain with a diagram how a variable refers to a memory location?**
- 11. While dealing with dynamic typing what caution must be taken care of? Explain with suitable example.**

PYTHON FUNDAMENTALS

Three Marks Questions

- 1. What are the python naming conventions?**
- 2. Explain the representation of string in python language.**
- 3. Explain the types of strings supported by python language.**
- 4. Explain escape sequences.**
- 5. Explain numerical literals supported in python language.**
- 6. Explain the Floating point literals supported in python language.**

PYTHON FUNDAMENTALS

Three Marks Questions

- 7. Explain the General structure of python program and give example.**
- 8. What is whitespace how its useful in python programming?**
- 9. What is input () function? Write down the general format of input () function and explain with proper example.**
- 10. What is print () function? Write down the general format of print () function and explain with proper example.**

Thank You

do follow me on github @tahirmirji for code snippets

@tahirmirji