PYTHON PROGRAMMING LANGUAGE

VISIT OUR WEBSITE

www.topitcourses.com

Python is one of the most important and widely used programming languages in the world today. Its popularity comes mainly from its simplicity and readability. Python uses easy-to-understand, English-like syntax, which makes it an ideal choice for beginners as well as professionals. Unlike many other programming languages, Python allows developers to focus more on problem-solving rather than complex syntax, making learning and development faster and more enjoyable.

One of the major reasons for Python’s importance is its wide range of applications across different industries. Python is used in web development, software development, data analysis, artificial intelligence, machine learning, automation, scientific computing, and cybersecurity. Popular companies like Google, Netflix, Instagram, and Amazon rely heavily on Python for building scalable and efficient systems. This versatility makes Python a valuable skill for anyone looking to build a strong career in technology.

Python plays a crucial role in data science and artificial intelligence, which are among the fastest-growing fields today. Powerful libraries such as NumPy, Pandas, Matplotlib, Scikit-learn, TensorFlow, and PyTorch enable developers and data scientists to analyze large amounts of data, create visualizations, and build intelligent models. Because of this, Python has become the preferred language for data-driven decision-making and AI-based solutions.

Another important advantage of Python is its ability to automate repetitive tasks. Python is widely used for scripting and automation in IT operations, testing, report generation, file handling, and system administration. By automating routine work, Python helps organizations save time, reduce errors, and improve productivity. This makes Python especially useful in DevOps and enterprise environments.

Python is also known for its strong community support and extensive library ecosystem. With thousands of open-source libraries and frameworks available, developers can quickly build applications without starting from scratch. Additionally, Python is a cross-platform language, meaning the same code can run on Windows, Linux, and macOS with minimal changes. Overall, Python’s simplicity, flexibility, wide usage, and career opportunities make it one of the most important programming languages to learn in the modern digital world.

Introduction

Python is an easy programming language and popular programming language too. Python is open-source and can get those libraries from python website python.org. In python, function and datatypes were implemented in C, C++. It can be used for many applications like data cleaning, databases and high-performance computing etc. It holds Data libraries like SciPy, NumPyetc.

Python is being used in Data Science technology development like Neural Networks, Artificial Intelligence, Statistics etc.

Python – Installation

You can understand the Installation of python on windows and Ubuntu in this tutorial. Before that let me give provide you the download link for downloading the Python.

For Linux and Unix Systems, below is the link to download Python:

https://docs.python.org/3/using/unix.html

For Windows system, below is the link to download Python:

https://docs.python.org/3/using/windows.html

Installation on Ubuntu or Linux Systems:

Python will come pre-installed on most of the Linux distributions. Just you need to give the
python3 to start programming in the terminal.

If you don`t have the Python, then get the source from

https://www.python.org/downloads/source/

follow the below commands.

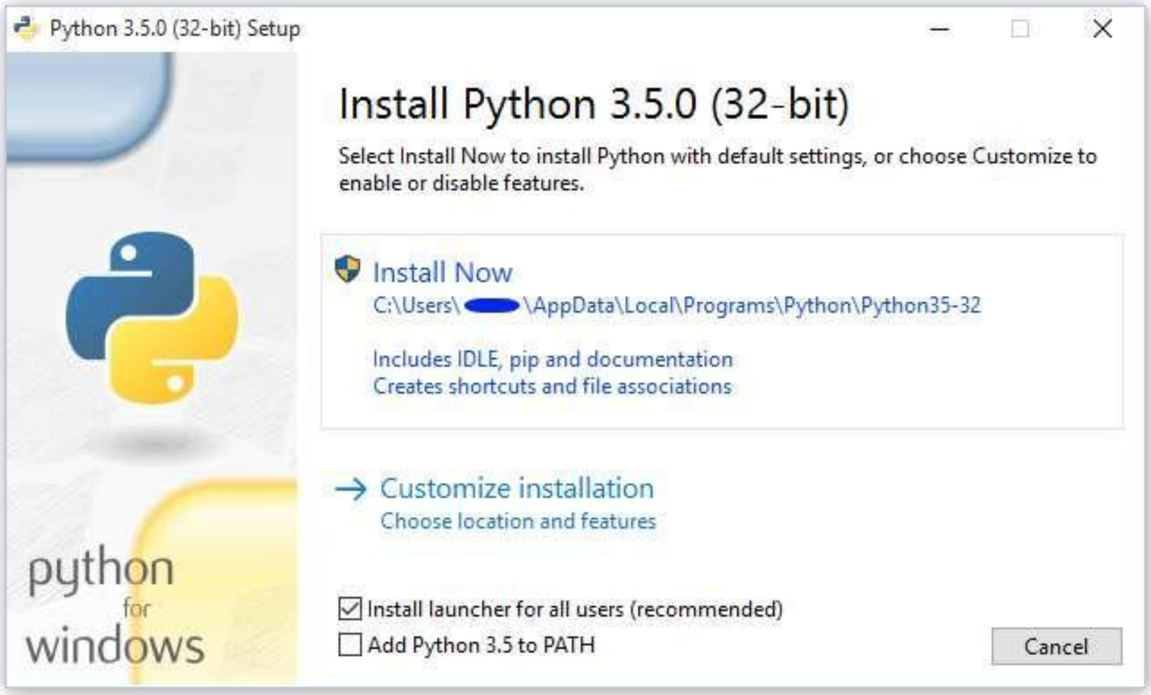
```
./configure

make

make install
```

Installation on windows systems:

Python installers are available to install python on 32-bit and 64-bit versions. Just download the installer and install the software. Once the installation opens the command prompt and give the python command to test.



Python 3.5.0 (32-bit) Setup

Install Python 3.5.0 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\[redacted]\AppData\Local\Programs\Python\Python35-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☐ Add Python 3.5 to PATH

Pic source: www.python.org

Python – Syntax

In this tutorial, you will learn the python syntax and an example about how to write a basic but popular Hello World print program.

Note: In this tutorial, we are using python version 3.5. Hence all the examples reflect the same results as like we execute in Python 3.5

Example:

```
>>> x="Hello World"

>>> print(x)

Hello World

>>>
```

In above Python terminal, x is a variable where we have assigned a value as hello world in double quotation as it is the string value. Then, we used print function to print the variable x which is in parenthesis. Do remember that we have given parenthesis for variable in print function.

Example:

```
>>> x=2

>>> print(x)

2

>>>
```

Python – Variables and Data types

Variable:

A variable is the location in the memory to store the values. A variable may hold different types of values like numbers, strings etc. In Python, no need to declare a datatype for a variable.It will understand by the value that is assigned to the variable.

The name of variable or a function that we define can be called as Identifier. An Identifier must obey the below rules.

1. Identifiers can have letters, digits, underscores.

2. there is no definite in length.

3. All identifiers must start with letter or underscore. You cannot use digits.

4. Identifier should not be a keyword. (which is a reserved word for python)

Examples of value assignment to a variable:

```
>>>x = 10                    # x is Integer

>>>y = 10.1                  # y is Float

>>>z = "Hello"                          # z is String

>>>x,y = y,x    # assign x value to y and y value to x

>>>a,b,c = 10,20,30 # assign a,b,c values sperated by comma at a time

>>>print(x,y,a,b,c)   # printing all the values
```

Data Types:

Python has different types of data types as below.

1. Numbers
2. String
3. List
4. Tuple
5. Dictionary
6. Boolean

Python – Numbers

In Python, Numbers can be defined in three ways.

1. Integer
2. Float
3. Complex

Let us understand how to work with Integer numbers

```
>>>x = 10    # assigning the integer value to variable x

>>>x

10
```

we can identify which type of value the variable x is holding

```
>>>x=10

>>>type(x)

<class 'int'>
```

Let us understand how to work with Float numbers

```
>>>y=10.1 # assigning the float value to variable y

>>>y

10.1
```

Let us identify the type of value the variable y is holding

```
>>>y=10.1

>>>type(y)
```

<class 'float'>

Let us understand how to work with complex type

>>>x=4+2y # 4 is the real part and 2y is the imaginary part

We can perform the various calculations with these numbers. Let us see few examples below.

```
>>>5+5

10

>>>5*5

25

>>>5-4

1
```

Python – Strings

In this tutorial, we will work on the Python Strings where we can learn about the manipulation of Strings, using String Operators and string methods and Functions. First, let us understand that how do we declare the strings in python programming language. We can declare and print the strings by placing them in single Quotes ('..'), Double Quotes (".."), and using the print function too. Python Strings are Immutable (An Object with a fixed value).

Using the Strings in single quotes ('...')

```
>>> 'hello world'

'hello world'
```

Below command will give an error.

```
>>> 'let's start'  # this will give us an error

 File ",stdin>, line 1

  'let's do it'

         ^

SyntaxError: invalid syntax
```

**To overcome that we must use escape character **

```
>>> 'let\'start'

"let's start"
```

Using the Strings in Double quotes ("...")

```
>>> "let's start"        # using double quotes to avoid escape character
```

Using the Strings in Print() function

```
>>>print("let's start")  # we have enclose the strings in double quotation inside print funtion

let's start
```

Using the 3 double quotes start and end of the string allows us to print the data including spaces and newlines.

```
>>>print("""let's
```

String Concatenation:

Multiple Strings can be concatenated using (+) symbol. Let us see the example of

```
>>> x="hello"

>>> y="world"

>>>x+y

'helloworld'
```

Starting from first character, 4th position excluded

```
>>>x[:-                  # Starting from fourth character from right, 4th position excluded
'PYTH'

      4]
>>>X[0:]                 # Starting from first character till end
'PY'

>>>x[-             # Starting from -6th position until start ie., -1 postion
'PYTHON'

      6:]
>>>x[:4]
'Y'  'PYTHON'
```

concatenating the strings.

Example

String Repetition:

String repetition can be performed by using the (*) symbol. Let us see the example of repetition of strings.

Example:

Strings are indexed with each character in a memory location when assigned to a variable. The indexed number starts from zero '0' from first character till the end of the string. Whereas, reverse indexing starts with '-1' from right to left until the starting character. Let us try few examples of retrieving the characters from a word PYTHON in either ways.

Example:

String Methods in Python:

Python String Methods	Description
capitalize()	Returns the String with first Character as Capital Letter
casefold()	Returns a casefolded copy
center(width[, fillchar])	This will pads the string with a character specified
count(sub[, start[, end]])	Returns the number of occurances of substring in string
encode(encoding="utf-8", errors="strict")	returns an encoded string
endswith(suffix[, start[, end]])	Check the string if it ends with the specified
expandtabs(tabsize=8)	Replace the tab with space
find(sub[, start[, end]])	returns the highest index
format(*args, **kwargs)	formats the string
format_map(mapping)	formats the string except the mapping is directly used
index(sub[, start[, end]])	returns the index of substring
isalnum()	checks for alphanumeric Char
isalpha()	Checks if all characters are Alphabets
isdecimal()	Checks for decimal characters

isdigit()	Checks for digit char
isidentifier()	checks for valid Identifier
islower()	checks for lowercase of all alphabets in string
isnumeric()	Checks for Numeric Char
isprintable()	Checks for Printable Char
isspace()	Checks for Whitespace Characters
istitle()	Returns true if the string is titlecased
isupper()	Checks if all characters are Uppercase
join(iterable)	returns concatenated string
ljust(width[, fillchar])	returns left-justified string

lower()	returns lowercased string
lstrip([chars])	Removes Leading Characters
partition(sep)	returns a tuple
replace(old, new[, count])	replaces the substring
rfind(sub[, start[, end]])	Returns the Highest Index
rindex(sub[, start[, end]])	Returns Highest Index but raises when substring is not found
rjust(width[, fillchar])	Returns the string right justified
rpartition(sep)	Returns a tuple
rsplit(sep=None, maxsplit=-1)	Splits String From Right
rstrip([chars])	Removes Trailing Characters
split(sep=None, maxsplit=-1)	Splits String from Left
splitlines([keepends])	Splits String at Lines
startswith(prefix[, start[, end]])	Checks if String Starts with the Specified String
strip([chars])	Removes Both Leading and Trailing Characters
swapcase()	swap uppercase characters to lowercase and vice versa
title()	Returns a Title Cased String
translate(table)	returns mapped charactered string
upper()	returns uppercased string
zfill(width)	Returns a Copy of The String Padded With Zeros

Python – Sequences

Sequence in Python can be defined with a generic term as an ordered set which can be classified as two sequence types. They are mutable and immutable. There are different types of sequences in python. They are Lists, Tuples, Ranges.

Lists: Lists will come under mutable type in which data elements can be changed.

Tuples: Tuples are also like Lists which comes under immutable type which cannot be changed.

Ranges: Ranges is mostly used for looping operations and this will come under immutable type.

The common Sequence Operations are listed below:

x in s: Returns true if an item in s is equal to x

x not in s : Returns false if an item in s is equal to x

s+t : concatenation

s*n : adding s to itself n number of times

s[i] : ith item of s, index starts from zero

s[i:j]: slice of s from i to j

len(s) : length of s

max(s) : largest item in s

min(s) : smallest item of s

s.count(x): total number of occurrences of x in s

Python – Lists

Python Lists holds the data of any datatype like an array of elements and these are mutable means the possibility of changing the content or data in it. List can be created by giving the values that are separated by commas and enclosed in square brackets. Let us see different types of value assignments to a list.

Example:

```
List1=[10,20,30,40,50];

List2=['A','B','C','D'];

List3=[10.1,11.2,12.3];
```

```
List4=['html','java','oracle'];

List5=['html',10.1,10,'A'];
```

As we know the way strings can be accessed, same way Lists can be accessed. Below is example of indexing in python for your understanding again.

Example:

```
List1=[10,20,30,40,50];

      0   1  2  3  4  ---> Forward Indexing

     -5  -4 -3 -2 -1  ---> Backward Indexing
```

Accessing and slicing the elements in List

Now let us take a list which holds different datatypes and will access the elements in that list.

Example:

```
>>> list5=['html',10.1,10,'A'];

>>> list5[0]

'html'

>>> list5[1:2];

[10.1]

>>>list5[-2:-1];

[10]

>>>list5[:-1];

['html', 10.1, 10]

>>>list5[:-2];

['html', 10.1]

>>>list5[1:-2];

[10.1]
```

```
>>>list5[1:-1];

[10.1, 10]

>>> list5[-1];

'A'

>>> list5[3:];

['A']
```

Using Functions with Lists:

```
>>> list5=['html',10.1,10,'A'];

>>>len(l

ist5) 4


>>> 10 in

list5 True

>>> 'html' in

list5 True


>>>num=[10,20,30,40];

>>>

sum(num)

100

>>>
```

Example:

Checking if the Lists are mutable:

Example:

```
>>>score=[10,20,30,80,50]

>>> score

[10, 20, 30, 80, 50]

>>>score[3]=40

>>> score

[10, 20, 30, 40, 50]
```

List Comprehension:

List comprehension works like iterate operations as mentioned below.

Syntax:

```
[x for x in iterable]
```

Example:

```
>>>var=[x for x in range(5)];

>>>var

[0, 1, 2, 3, 4]

>>>var=[x+1 for x in range(5)];

>>>var

[1, 2, 3, 4, 5]

>>>var=[x for x in range(5) if x%3==0];

>>>var

[0, 3]
```

Adding Elements to a list:

We can add two lists as shown in the below example.

Example:

```
>>> var1=[10,20]

>>> var2=[30,40]

>>> var3=var1+var2

>>> var3

[10, 20, 30, 40]
```

Replicating elements in Lists:

we can replicate elements in lists as shown in the below example.

Example:

```
>>> var1*2

[10, 20, 10, 20]



>>> var1*3

[10, 20, 10, 20, 10, 20]



>>> var1*4

[10, 20, 10, 20, 10, 20, 10, 20]
```

Appending elements in Lists:

We can append an element to an existing list as shown in the below example.

Example:

```
>>>  var1.append(30)

>>> var1
```

```
[10, 20, 30]



>>> var1.append(40)

>>> var1

[10, 20, 30, 40]

>>> var2

[30, 40]
```

Python – Tuples

Tuples are generally used to store the heterogeneous data which immutable. Even Tuple looks like Lists but Lists are mutable. To create a tuple, we need to use the comma which separates the values enclosed parentheses.

Example:

```
>>> tup1=()          # Creating an empty tuple

>>>tup1 ()



>>> tup1=(10)
>>>tup1

10


>>> tup1=(10,20,30);

>>> tup1 (10,

20, 30)

>>> tup1=tuple([1,1,2,2,3,3])

>>>
     tup1

(1, 1, 2, 2, 3, 3)
>>>
```

```
>>> tup1

'tuple'
```

```
>>> tup1=(10,20,30);

>>> max(tup1) 30


>>> min(tup1) 10


>>>len(tup1)

3
```

Operators with Tuples:

```
>>> 20 in tup1

True


>>> 30 not in tup1
False
```

Slicing in Tuples:

```
>>> tup1[0:4]

(10, 20, 30)


>>> tup1[0:1]

(10,)
>>>
        tup1[0:2]
```

Python – Dictionary

Dictionaries are created or indexed by key-value pairs. In which keys are immutable type.

Tuples can be used as keys, but lists cannot be used as keys. Just because lists are mutable.

Generally, the key-value pairs which are stored in the Dictionary can be accessed with the
key. we can delete a key value too. Let us see some examples.

Example:

```
>>>    score={'maths':80,'physics':70,'chemistry':85}

>>> score


{'physics': 70, 'maths': 80, 'chemistry': 85}

>>> score['maths'] 80

>>>
>>> del score['maths']

Traceback (most recent call last): File
    score['maths']

"<stdin>", line 1, in <module> KeyError:

'maths'

>>> score


>>>score.keys() dict_keys(['physics',

'chemistry'])



>>>
{'physics': 70, 'chemistry': 85}
dict_keys(['physics', 'chemistry'])
```

Python – Ranges

Range a kind of data type in python which is an immutable. Range will be used in for loops
for number of iterations. Range is a constructor which takes arguments and those must be
integers. Below is the syntax.

Syntax:

```
class  range(stop)

class range(start, stop[, step])
```

stop: the value of stop parameter.

step: the value of step parameter. If the value is omitted, it defaults to 1.

start: the value of start parameter. If the value is omitted, it defaults to zero.

Examples:

```
>>>list(range(5))

[0, 1, 2, 3, 4]

>>>list(range(10,20))

[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

>>>list(range(10,20,5))

[10, 15]

>>>list(range(10,20,2))

[10, 12, 14, 16, 18]

# It will not take the float numbers

>>>list(range(0,0.1))

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

TypeError: 'float' object cannot be interpreted as an integer

>>>list(range(0,2))

[0, 1]
```

```
>>>list(range(0,1))

[0]

>>>list(range(0,10,5))

[0, 5]
```

Python – Sets

In this tutorial, we will learn about sets in python. A set is a datatype which holds an unordered collection with immutable and no duplicate elements. By the name, Set can be used for various mathematical operations. Mathematical operation may be union, intersection or difference, etc. Let us see the example of using the Set below.

Example:

```
>>> set1={'html','c','java','python','sql'} print(set1)

>>>
{'c', 'python', 'sql', 'html', 'java'}


# Below we have given duplicates

>>>
       set1={'html','c','java','python','sql','java'}
# we can observe that duplicates are ignored

>>>

{'c', 'python', 'java', 'html', 'sql'}
       print(set1)
>>> set1

{'c', 'python', 'java', 'html', 'sql'}
```

Membership testing in Sets:

```
>>>   set1={'html','java','python','sql','java'}

>>> set1

{'python', 'java', 'html', 'sql'}

>>> print(set1)

{'python', 'java', 'html', 'sql'}
```

```
>>> 'c' in set1

False

>>> 'java' in set1

True
```

Sets Operations in Python:

```
>>>    set1={'html','java','python','sql','java'}

>>>  set2={'html','oracle','ruby'}


# Unique words in set1

>>> set1

{'python', 'java', 'html', 'sql'}

>>> set2




# words in set1 but not in set2

>>> set1-set2

{'python', 'java', 'sql'}


{'ruby', 'html', 'oracle'}
# Words in set1 or set2 or both

>>>

{'ruby', 'html', 'oracle', 'python', 'java', 'sql'}




>>>
```

```
{'html'}


# Words in set1 or set2 but not both

>>> set1^set2

{'oracle', 'python', 'sql', 'ruby', 'java'}
```

Python - Operators

Operators in Python helps us to perform the mathematical operations with numbers. There are different operators in Python as below.

Operators	Description
//	Integerdivision
+	addition
-	subtraction
*	multiplication
/	Float division
%	Provide remainder after division(Modulus)
**	Perform exponent (raise to power)

Let us try implementing every operator now.

1. Addition: symbol used (+)

```
>>> 10+10

20

>>>20+30

50

>>>50+50

100
```

Substration: symbol used (-)

```
>>>20-10
10
>>>50-40
10
>>>100-30
70
```

1. **multiplication: Symbol used (*)**

```
>>>5*2
10
>>>10*2
20
>>>20*2
40
```

2. **Float Division: This will divide and provide the result in floating value and the symbol used (/)**

```
>>>5/2
2.5
>>>10/2
5.0
```

3. **Integer Division: This will divide and truncate the decimal and provide the Integer value and the symbol used (//)**

```
>>>5//2
2
>>>7//2
```

3

4. Exponentiation Operator: This will help us to calculate a power b and return the result

```
>>>10**3 # This mean 10*10*10
1000
```

5. Modulus Operator: This will provide the remainder after the calucation and symbol used (%)

```
>>>10%3
1
```

What if we want to work with multiple operators at a time. Here comes the Operator precedence in Python.

Operator	Description
lambda	Lambda expression
if – else	Conditional expression
or	Boolean OR
and	Boolean AND
not x	Boolean NOT
in, not in, is, is not, <, <=, >, >=, !=, ==	Comparisons, including membership tests and identity tests
|	Bitwise OR
^	Bitwise XOR
&	Bitwise AND
<<, >>	Shifts

+, -	Addition and subtraction
*, @, /, //, %	Multiplication, matrix multiplication division, remainder [5]
+x, -x, ~x	Positive, negative, bitwise NOT
**	Exponentiation [6]
await x	Await expression
x[index], x[index:index], x(arguments...), x.attribute	Subscription, slicing, call, attribute reference
(expressions...), [expressions...], {key: value...}, {expressions...}	Binding or tuple display, list display, dictionary display, set display

Python –If.. Else.. Statements

In this tutorial, we will discuss about the "if" Condition statement. Let us understand how this "if" statements work. There will be 2 parts in the "if" statement. They are "if" and "elif", "else" which is optional. when the "if" condition satisfies then it executes the program inside that else it execute the program inside "elif" or "else" statements.

Syntax:

```
if Condition:

   program of if

elif test expression:

   program of elif

else:

   program of else
```

Below is the example of If.. Else:

Example 1:

```
>>> x=int(input("Please enter a number:"))

Please enter a number:10
```

```
>>> if x<0:



...     print("negative is zero")

... elif x==1:

...
...
...    print("positive and greater than 1")
        print("si

    ngle") else:
positive and greater than 1
```

Example 2:

```
>>> x=int(input("Please enter a number:"))

Please enter a number:1

>>> if x<0:
...     print("negative is zero")

... elif x==1:

...     print("single")

     else:
...

...    print("positive and greater than 1")

...

Single
```

Python – For Loop

In this tutorial, we will learn about for loop. In Python, for loop is used to iterate over the sequence of elements (the sequence may be list, tuple or strings.. etc). Below is the syntax.

Syntax:

```
for_stmt ::=  "for" target_list "in" expression_list ":" suite

              ["else" ":" suite]
```

In the below example, we have given the list of strings as courses and the for loop created to iterate through all the strings to print the course and the length of the course name.

Example:

```
>>> courses=['html','c','java','css']

>>> for i in courses:

...     print(i, len(i))

...

html 4
c 1

java 4

css 3

>>>
```

In the below example, for loop iterates through the list of numbers. In the immediate step, if statement filters only the numbers less than 50, else it will display "no values" for rest of the iterations.

Example:

```
>>> x=[10,20,30,40,50,60]

>>> x

[10, 20, 30, 40, 50, 60]
```

```
>>> for i in x:
...     if i<50:
...         print(i)
...     else:
...         print("no values")
```

Below is the output:

```
10
20
30
40
no values
no values
```

Python – While Loop

In this tutorial, we will learn about while loop. In python, while loop is used to iterate until the condition is satisfied. If the condition given is not satisfied in the first iteration itself, the block of code inside the loop will not get executed.

In the below example, we have assigned the value of x as zero and started the while loop until the value of x is less than 10 and print the values.

Example:

```
>>> x=0
>>> while x<10:
...     x=x+1
...     print(x)
...
```

Below is the Output:

```
1

2

3

4

5

6

7

8

9

10
```

Just changed the values for the above example and below is the output.

```
>>> x=100

>>> while x<110:

...     x=x+1

...     print(x)

...
```

Note:

 This is a **preview Python e-Book** containing **only 30 pages**.
 It is provided to help you understand **how the Python e-Book and is structured**.
 The **complete Python** includes detailed concepts, real-world examples, and career guidance.
 Buy now from our official website: https://topitcourses.com/python-core-advanced/