Python

foxmula.com

Contents

foxmula.com

Popular

Quora

DISQUS

Google

NASA

Pinterest

Insta

foxmula.com

What is Python ?

A dynamic, **open source programming language** with a focus on **simplicity and productivity**. It has an elegant syntax that is natural to read and easy to write.

foxmula.com

There are so many programming Languages. Why Python ?

Why Python?

- **Python is *Simple & Beautiful***

- **Python is Object-Oriented**
  - Structure supports concepts as *Polymorphism*, *Operation overloading & Multiple Inheritance*

- **Python is Free (Open Source Language)**
  - Downloading & Installing Python is *Free & Easy*
  - *Source Code* is easily *Accessible*

- **Python is Portable & Powerful**
  - Runs virtually on every major Platforms used today
  - *Dynamic Typing, Built-in types & tools, Library Utilities, Automatic Memory Management*

foxmula.com

Java:

```
public class HelloWorld {
    public static void main(String[], args) {
        System.out.println("Hello, World!");
    }
}
```

Python:

foxmula.com

```
print "Hello, World"
```

There are two versions of Python currently available –

Python 2.x

Python 3.x

foxmula.com

Which version to use

Python2 or **Python3?**

foxmula.com

Lets see the basic differences-

Python 2.x is legacy, Python 3.x is the present and future of the language.

The most visible difference is probably the way the “print” statement works.

In python2 it’s a statement

```
        print “Hello World!”
```

In Python3 it’s a function

```
        print(“Hello World!”)
```

foxmula.com

Getting Started with Python

Type the following text at the Python prompt and press the Enter:

```
>>> print "Hello, Python!“
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in

```
print ("Hello, Python!")
```

However in Python version 2.4.3, this produces the following result:

foxmula.com

```
Hello, Python!
```

Python Strings:

Strings in Python are identified as a contiguous set of characters in between quotation marks.

```
str = 'Hello World!'

Print(str) # Prints complete string
print(str[0]) # Prints first character of the string
Print(str[2:5]) # Prints characters starting from 3rd to 6th
print(str[2:]) # Prints string starting from 3rd character
print(str * 2) # Prints string two times
print(str + "TEST“) # Prints concatenated string
```

foxmula.com

Python Lists:

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]).

```
list1 = [ ‘hello', 786 , 4.1, ‘Ram', 702 ]
list2 = [123, ‘Victor']

print(list1) # Prints complete list
print(list1[0]) # Prints first element of the list
print(list1[1:3]) # Prints elements starting from 2nd to 4th
print(list1[2:]) # Prints elements starting from 3rd element
print(list2 * 2) # Prints list two times
print(list1 + list2) # Prints concatenated lists
```

Toxmula.com

Positive and negative indices

```
>>> t = [23, ‘abc’, 4.56, [2,3], ‘def’]
```

Positive index: count from the left, starting with 0

```
>>> t[1]
‘abc’
```

Negative index: count from right, starting with –1

```
>>> t[-3]
4.56
```

foxmula.com

Python Tuples:

A tuple is another sequence data type that is similar to the list. Unlike lists, however, tuples are enclosed within parentheses.

```
tuple1 = ( ‘hello', 786 , 2.23, ‘victor', 70.2 )
tuple2 = (123, ‘jay')
print(tuple1) # Prints complete list
print(tuple1[0]) # Prints first element of the list
print(tuple1[1:3]) # Prints elements starting from 2nd to 4th
print(tuple1[2:]) # Prints elements starting from 3rd element
print(tuple2 * 2) # Prints list two times
print(tuple1 + tuple2) # Prints concatenated lists
```

foxmula.com

Python Dictionary:

Python 's dictionaries consist of key-value pairs.

```
dict1 = {'name': 'john','code':6734, 'dept': 'sales'}

print(dict1['name’]) # Prints value for 'one' key
print(dict['code’]) # Prints value for 2 key
print(dict1) # Prints complete dictionary
print(dict1.keys()) # Prints all the keys
Print(dict1.values()) # Prints all the values
```

foxmula.com

The ‘in’ Operator

- Boolean test whether a value is inside a container:

```
>>> t = [1, 2, 4, 5]
>>> 3 in t
False
>>> 4 in t
True
>>> 4 not in t
False
```

- For strings, tests for substrings

```
>>> a = 'abcde'
>>> 'c' in a
True
>>> 'cd' in a
True
>>> 'ac' in a
False
```

- Be careful: the *in* keyword is also used in the syntax of *for loops* and *list comprehensions*

foxmula.com

Lists are mutable

```
>>> li = [‘abc’, 23, 4.34, 23]
>>> li[1] = 45
>>> li
    [‘abc’, 45, 4.34, 23]
```

- We can change lists *in place.*
- Name *li* still points to the same memory reference when we’re done.

foxmula.com

Tuples are immutable

```
>>> t = (23, ‘abc’, 4.56, (2,3), ‘def’)
>>> t[2] = 3.14

Traceback (most recent call last):
  File "<pyshell#75>", line 1, in -toplevel-
    tu[2] = 3.14
TypeError: object doesn't support item assignment
```

- You can’t change a tuple.
- You can make a fresh tuple and assign its reference to a previously used name.

```
>>> t = (23, ‘abc’, 3.14, (2,3), ‘def’)
```

- *The immutability of tuples means they’re faster than lists.*

foxmula.com

Operations on Lists Only

```
>>> li = [1, 11, 3, 4, 5]

>>> li.append('a')      # Note the method syntax
>>> li
[1, 11, 3, 4, 5, 'a']

>>> li.insert(2, 'i')
>>>li
[1, 11, 'i', 3, 4, 5, 'a']
```

foxmula.com

Operations on Lists Only

Lists have many methods, including index, count, remove, reverse, sort

```
>>> li = ['a', 'b', 'c', 'b']
>>> li.index('b')   # index of 1st occurrence
1
>>> li.count('b')   # number of occurrences
2
>>> li.remove('b')  # remove 1st occurrence
>>> li
   ['a', 'c', 'b']
```

foxmula.com

If else structure

```
if expression1:
        statement(s)
elif expression2:
        statement(s)
elif expression3:
        statement(s)
else:
        statement(s)
```

foxmula.com

For loop

```
for iterating_var in sequence:
    statements(s)
```

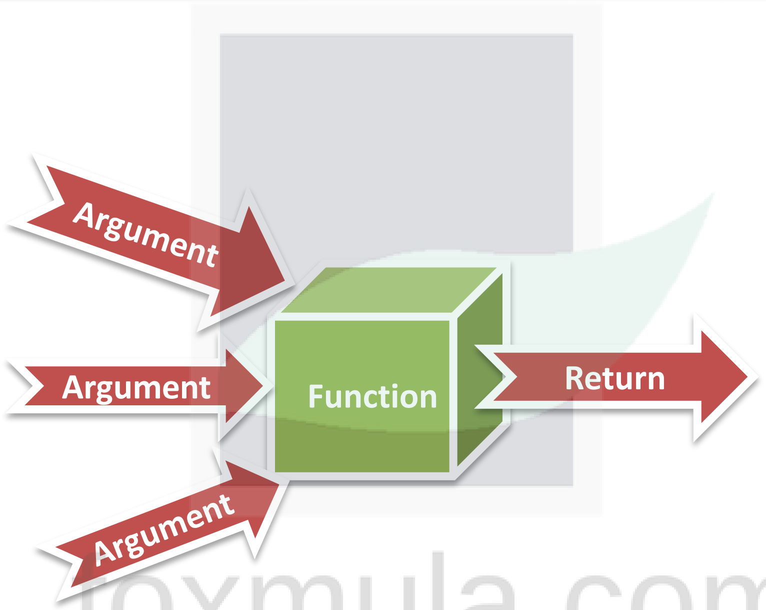
foxmula.com

While loop

```
while expression:
    statements(s)
```

foxmula.com

Defining Functions



foxmula.com

Defining Functions

```
def functionname( parameters ):
     "function_docstring"
    function_suite
    return [expression]
```

Example:

```
def printme( str ):
    "This prints a passed string into this function"
    print(str)
    return
```

foxmula.com

Class

```
>>> class person:
        def sayhi(self):
            print('Hello, How are you?')
>>> p=person()
>>> p.sayhi()
Hello, How are you?
```

foxmula.com

Inheritance :

```
>>> class A:
        def hello(self):
                print('I am Super Class')
>>> class B(A):
        def bye(self):
                print('I am Sub Class')

>>> p=B()
>>> p.hello()
I am Super Class
>>> p.bye()
I am Sub Class
```

foxmula.com

Using in-built Modules

```
>>> import time
>>> print(‘The sleep started’)
The Sleep started
>>> time.sleep(3)
>>> print(‘The sleep Finished’)
The sleep Finished
```

foxmula.com

Serial Communication

```
Import serial

Ser=serial.Serial('COM4',9600)

#for reading serial data
print(ser.readline())

# for sending serial data
Ser.write('s')
```

foxmula.com

Python Arithmetic Operators

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

foxmula.com

Python Comparison Operators

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

foxmula.com

Python Assignment Operators

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a

foxmula.com

Python Logical Operators

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

foxmula.com

Python Membership Operators

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not find a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

foxmula.com

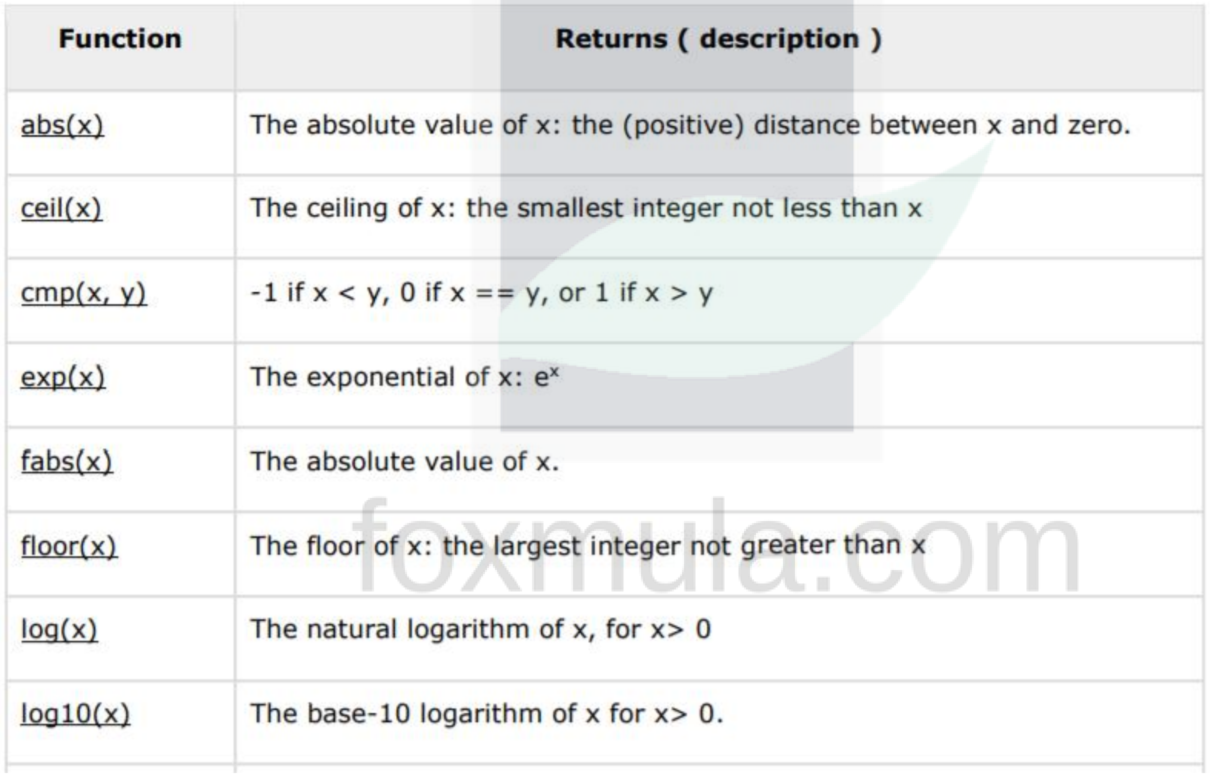
NUMBERS

Python Number Conversion

- Type int(x) to convert x to a plain integer.
- Type long(x) to convert x to a long integer.
- Type float(x) to convert x to a floating-point number.
- Type complex(x) to convert x to a complex number with real part x and imaginary part zero.
- Type complex(x, y) to convert x and y to a complex number with real part x and imaginary part y. x and y are numeric expressions

foxmula.com

NUMBERS



Function	Returns (description)
abs(x)	The absolute value of x: the (positive) distance between x and zero.
ceil(x)	The ceiling of x: the smallest integer not less than x
cmp(x, y)	-1 if x < y, 0 if x == y, or 1 if x > y
exp(x)	The exponential of x: e^x
fabs(x)	The absolute value of x.
floor(x)	The floor of x: the largest integer not greater than x
log(x)	The natural logarithm of x, for x> 0
log10(x)	The base-10 logarithm of x for x> 0.

foxmula.com

String Parsing

- Print("My name is %s and weight is %d kg!" % ('Zara', 21))

Format Symbol	Conversion
%c	character
%s	string conversion via str() prior to formatting
%i	signed decimal integer
%d	signed decimal integer
%u	unsigned decimal integer
%o	octal integer
%x	hexadecimal integer (lowercase letters)
%X	hexadecimal integer (UPPERcase letters)
%e	exponential notation (with lowercase 'e')
%E	exponential notation (with UPPERcase 'E')

foxmula.com

Sets

- A set contains an unordered collection of unique and immutable objects. The set data type is, as the name implies, a Python implementation of the sets as they are known from mathematics. This explains, why sets unlike lists or tuples can't have multiple occurrences of the same element.

```
>>> adjectives = {"cheap","expensive","inexpensive","economical"}
>>> adjectives
set(['inexpensive', 'cheap', 'expensive', 'economical']) >>>
```

```
>>> x = set("A Python Tutorial")
>>> x
set(['A', ' ', 'i', 'h', 'l', 'o', 'n', 'P', 'r', 'u', 't', 'a', 'y', 'T'])
```

```
>>> cities = set(("Paris", "Lyon", "London","Berlin","Paris","Birmingham"))
>>> cities
set(['Paris', 'Birmingham', 'Lyon', 'London', 'Berlin']) >>>
```

foxmula.com

This Week

- Socket Programming – Networking with python
- Email Access
- Web Access
- MySQL Integration
- Os, timieit, math and shutil
- Python Regular Expressions
- Testing with python

foxmula.com

Socket Programming –?

- Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.
- Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on. The socket library provides specific classes for handling the common transports as well as a generic interface for handling the rest.

foxmula.com

TCP	UDP
TCP is a connection-oriented protocol	UDP is a connectionless protocol.
TCP is suited for applications that require high reliability, and transmission time is relatively less critical.	UDP is suitable for applications that need fast, efficient transmission, such as games. UDP's stateless nature is also useful for servers that answer small queries from huge numbers of clients.
HTTP, HTTPs, FTP, SMTP, Telnet	DNS, DHCP, VOIP.
The speed for TCP is slower than UDP.	UDP is faster because error recovery is not attempted. It is a "best effort" protocol.
There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent.	There is no guarantee that the messages or packets sent would reach at all.

foxmula.com

Socket programming using TCP

domain

The family of protocols that is used as the transport mechanism. These values are constants such as AF_INET, PF_INET, PF_UNIX, PF_X25, and so on.

type

The type of communications between the two endpoints, typically SOCK_STREAM for connection-oriented protocols and SOCK_DGRAM for connectionless protocols.

foxmula.com

Server socket method

- s.bind()

This method binds address (hostname, port number pair) to socket.

- s.listen()

This method sets up and start TCP listener.

- s.accept()

This passively accept TCP client connection, waiting until connection arrives (blocking).

foxmula.com

Client socket method

- s.connect()

This method actively initiates TCP server connection

foxmula.com

Genral socket programming

s.recv()

- This method receives TCP message

s.send()

- This method transmits TCP message

s.recvfrom()

- This method receives UDP message

s.sendto()

- This method transmits UDP message

s.close()

- This method closes socket

socket.gethostname()

- Returns the hostname.

foxmula.com

Exceptions - Python

- Python provides two very important features to handle any unexpected error in your Python programs and to add debugging capabilities in them:

- Exception Handling
- Assertions:

foxmula.com

What is Exception?

- An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error.

- When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

foxmula.com

OS

- The main purpose of the OS module is to interact with your operating system. The primary use is to create folders, remove folders, move folders, and sometimes change the working directory.

foxmula.com

RE

- Identifiers:
- \d = any number
- \D = anything but a number
- \s = space
- \S = anything but a space
- \w = any letter
- \W = anything but a letter
- . = any character, except for a new line
- \b = space around whole words
- \. = period. must use backslash, because . normally means any character.

foxmula.com

RE

- Modifiers:
- {1,3} = for digits, u expect 1-3 counts of digits, or "places"
- + = match 1 or more
- ? = match 0 or 1 repetitions.
- * = match 0 or MORE repetitions
- $ = matches at the end of string
- ^ = matches start of a string
- | = matches either/or. Example x|y = will match either x or y
- [] = range, or "variance"
- {x} = expect to see this amount of the preceding code.
- {x,y} = expect to see this x-y amounts of the precedng code

foxmula.com

RE

- White Space Charts:
- \n = new line
- \s = space
- \t = tab
- \e = escape
- \f = form feed
- \r = carriage return
- Characters to REMEMBER TO ESCAPE IF USED!
- . + * ? [ ] $ ^ ( ) { } | \
- Brackets:
- [] = quant[ia]tative = will find either quantitative, or quantatative.
- [a-z] = return any lowercase letter a-z
- [1-5a-qA-Z] = return all numbers 1-5, lowercase letters a-q and uppercase A-Z

foxmula.com

Features and philosophy

- Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability, and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

foxmula.com

Exercise

- Given an array of ints length 3, return an array with the elements "rotated left" so [1, 2, 3] yields [2, 3, 1].
-

rotate_left3([1, 2, 3]) → [2, 3, 1]
rotate_left3([5, 11, 9]) → [11, 9, 5]
rotate_left3([7, 0, 0]) → [0, 0, 7]

foxmula.com

solution

```
def rotate_left3(nums):
        l=len(nums)
        k=[]
        for i in range(1,l):
                k.append(nums[i])
        k.append(nums[0])
        return k
```

foxmula.com

Exercise

- #Given 2 int arrays, a and b, each length 3, return a new array length 2 containing their middle elements.
- middle_way([1, 2, 3], [4, 5, 6]) → [2, 5]
- middle_way([7, 7, 7], [3, 8, 0]) → [7, 8]
- middle_way([5, 2, 9], [1, 4, 5]) → [2, 4]

foxmula.com

Exercise

#Given 2 arrays of ints, a and b, return True if they have the same first element or they have the same last element. Both arrays will be length 1 or more.

```
common_end([1, 2, 3], [7, 3]) → True
common_end([1, 2, 3], [7, 3, 2]) → False
common_end([1, 2, 3], [1, 3]) → True
```

foxmula.com

Exercise

#Given an array of ints, return True if the sequence of numbers 1, 2, 3 appears in the array somewhere.

array123([1, 1, 2, 3, 1]) → **True**

array123([1, 1, 2, 4, 1]) → **False**

array123([1, 1, 2, 1, 2, 3]) → **True**

foxmula.com

solution

```python
def array123(nums):
        # Note: iterate with length-2, so can use i+1 and i+2 in the loop
        for i in range(len(nums)-2):
                if nums[i]==1 and nums[i+1]==2 and nums[i+2]==3:
                        return True
        return False
```

foxmula.com

Exercise

#Given a non-empty string like "Code" return a string like "CCoCodCode".

string_splosion('Code') → 'CCoCodCode’

string_splosion('abc') → 'aababc’

string_splosion('ab') → 'aab'

foxmula.com

solution

```
def string_splosion(str):
        result = ""
        # On each iteration, add the substring of the chars 0..i
        for i in range(len(str)):
                result = result + str[:i+1]
        return result
```

foxmula.com

Exercise
The password generator

- Write a password generator in Python. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. The passwords should be random, generating a new password every time the user asks for a new password. Include your run-time code in a main method.
- Extra:
- Ask the user how strong they want their password to be. For weak passwords, pick a word or two from a list.

foxmula.com

Exercise the hangman

- In the game of Hangman, a clue word is given by the program that the player has to guess, letter by letter. The player guesses one letter at a time until the entire word has been guessed. (In the actual game, the player can only guess 6 letters incorrectly before losing).
- Let’s say the word the player has to guess is “EVAPORATE”. For this exercise, write the logic that asks a player to guess a letter and displays letters in the clue word that were guessed correctly. For now, let the player guess an infinite number of times until they get the entire word. As a bonus, keep track of the letters the player guessed and display a different message if the player tries to guess that letter again. Remember to stop the game when all the letters have been guessed correctly!

foxmula.com

- https://www.google.com/settings/security/lesssecureapps

foxmula.com