**What is Python?**

* It is a programing language
* It is simple,poweful and general-purpose programing language
* It is very readable language
* Use text editor or IDLE for writing code.
* File extension is .py

**Advantages and Disadvantages?**

**Advantages of Python**

1#- Python is easy to learn for even a novice developer. Its code is easy to read and you can do a lot of things just by looking at it. Also, you can execute a lot of complex functionalities with ease, thanks to the standard library.

#2- Supports multiple systems and platforms.

#3- Object Oriented Programming-driven.

#4- With the introduction of Raspberry Pi, a card sized microcomputer, Python has expanded its reach to unprecedented heights. Developers can now build cameras, radios and games with ease. So, learning Python could open new avenues for you to create some out-of-the box gadgets.

#5- Python has a plethora of frameworks that make web programming very flexible. [**Django**](http://findnerd.com/list/view/Django-Framework-/14165/) is the most famous Python framework for web development.

#6- Gives rise to quick development by using less code. Even a small team can handle Python effectively.

#7- Allows to scale even the most complex applications with ease.

#8- A large number of resources are available for Python.

#9- Offers a built in testing framework to set debugging time and enable fastest workflows.

#10- IT giants like Yahoo, Google, IBM, NASA, Nokia and Disney prefer Python.

1. Python can be easy to pick up due to the flexibility of the language and high-level nature of the data types.

2. Python is an interpreted high-level language and interpreters are available for Windows, Mac OS X, and Unix. Thus the same code can be executed on these different platforms with no changes to the code.

3. The Python Package Index (PyPI) hosts thousands of third-party modules for Python. So you have endless possibilities. ( Web and Internet Development, Database Access, Desktop GUIs, Scientific & Numeric, Education, Network Programming and Software & Game Development.)

4. Python is open-source, you can use it and distribute, even for commercial projects.

5. Python determines the scope of a block based on its indentation. This forces Python's programmers to indent code blocks, which keeps code cleaner.

6. OOPS is well implemented in Python.

7. Python data structure is superior to many other programming languages.

**Disadvantages of Python**

#1- Python is slow.

#2- Python is not a very good language for mobile development.

#3- Python is not a good choice for memory intensive tasks.

#4- It's near impossible to build a high-graphic 3D game using Python.

#5- Has limitations with database access.

#6- Python is not good for multi-processor/multi-core work.

**Compiler & Interpreter?**

|  |  |
| --- | --- |
| Compliler | Interpreter |
| * Compliler takes entire code as input * Intermediate object code is generate * Conditional control statements are executes faster * Program need not be compiled every time. * Errors are dispaled after entire code checked.   Examples – c, java etc... | * Interpreter takes single line code as input. * No intermediate object code is generated. * Conditional control statements are executes slower. * Every high level program converted into low level program (line by line) * Errors are dispaled for every instruction interpreted(If any).   Examples – Basic, python etc... |

**Static and Dynamic programing languages?**

Executable code

Source code

Compiler

Execution

|  |  |
| --- | --- |
| Static | Dynamic |
| * Need to mention data type name before the variable.   + Int a = 5   + string a =”5” * Datatypes decides on compile time. * Developer needs to take care all datatypes.   Examples – c, java | * No need to mention datatype name before the variable.   + a=5   + a=”5” * Datatypes decides on execution time. * Developer do not needs to take all datatypes.   Examples – Basic, python |

**Basic funcions in python**

|  |  |
| --- | --- |
| Function Name | Description |
| len() | It is used to get the lenght of elements in datastructure(except integer).  Example -  variable = “python”  print len(variable)  output - 6 |
| dir() | It is used to get the function names which we can perform on given data structure.  Example -  variable = 10  print dir(variable)  output-  ['\_\_abs\_\_', '\_\_add\_\_', '\_\_and\_\_', '\_\_class\_\_', '\_\_cmp\_\_', '\_\_coerce\_\_', '\_\_delattr\_\_', '\_\_div\_\_', '\_\_divmod\_\_', '\_\_doc\_\_', '\_\_float\_\_', '\_\_floordiv\_\_', '\_\_format\_\_', '\_\_getattribute\_\_', '\_\_getnewargs\_\_', '\_\_hash\_\_', '\_\_hex\_\_', '\_\_index\_\_', '\_\_init\_\_', '\_\_int\_\_', '\_\_invert\_\_', '\_\_long\_\_', '\_\_lshift\_\_', '\_\_mod\_\_', '\_\_mul\_\_', '\_\_neg\_\_', '\_\_new\_\_', '\_\_nonzero\_\_', '\_\_oct\_\_', '\_\_or\_\_', '\_\_pos\_\_', '\_\_pow\_\_', '\_\_radd\_\_', '\_\_rand\_\_', '\_\_rdiv\_\_', '\_\_rdivmod\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_rfloordiv\_\_', '\_\_rlshift\_\_', '\_\_rmod\_\_', '\_\_rmul\_\_', '\_\_ror\_\_', '\_\_rpow\_\_', '\_\_rrshift\_\_', '\_\_rshift\_\_', '\_\_rsub\_\_', '\_\_rtruediv\_\_', '\_\_rxor\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_sub\_\_', '\_\_subclasshook\_\_', '\_\_truediv\_\_', '\_\_trunc\_\_', '\_\_xor\_\_', 'bit\_length', 'conjugate', 'denominator', 'imag', 'numerator', 'real'] |
| print() | It is used to display output in command form. |
| input() | It is used to get the required data from the user. |
| raw\_input() | It is also used to get the input data from the users we can give any data type should be string. |
| int() | It is used to convert a string has numbers or integers.  String contains only numbers in it. |

**Data Types**

|  |  |
| --- | --- |
| Function Name | Description |
| Integer | No need to mention data type name before the variable name.  Python memory management allocate the default (or) apropriate datatype based on given value.  Example -  a = 15  print a  output - 15 |
| Float | It is used to get the decimal values.  Example -  a = 10.5  print a  output – 10.5 |

**Data Structures**

1. strings
2. list
3. tuple
4. dictionary
5. set

**Strings:** Group of characters wchich are enclosed by two single quotes or double quotes or triple quotes. Single and double code strings are same but triple code strings are used to define multiline strings.

Syntax:

variable\_name = string

Example -

variable\_name = "hello world"

print variable\_name

output - hello world

String Functions

|  |  |  |  |
| --- | --- | --- | --- |
| Function Name | Description | Syntax | Examples |
| capitalize | It is used to make first letter has capital.  It won't take any parameters. | var.capitalize() | a = "hello world"  print a.capitalize()  out put - Hello world |
| count | It is used to get the repeatings or the occurance of particular element in the string.  It takes one parameter. | var.count('element') | a = "hello world"  print a.count('l')  out put - 3 |
| startswith | It is used check wether the string startswith given element or not.If condition satisfy it will return true otherwise false.  It takes one parameter. | var.startswith('element') | a = "hello world"  print a.startswith('l')  out put – false  a = "hello world"  print a.startswith('h')  out put - true |
| endswith | It is used check wether the string endswith given element or not.If condition satisfy it will return true otherwise false.  It takes one parameter. | var.endswith() | a = "hello world"  print a.endswith('l')  out put – false  a = "hello world"  print a.endswith('d')  out put - true |
| find | It is used to find index position of the element in string.  It takes one parameter. | var.find('element') | a = "hello world"  print a.find('d')  out put – 10 |
| index | It is also used to find index position of the element in string.  It takes one parameter | var.index('element')  OR  var.index('element', 'position') | a = "hello world"  print a.index('e')  out put – 1  a = "hello world"  print a.index('l', 4)  out put – 9 |
|  |  | var.capitalize() |  |
|  |  | var.capitalize() |  |
|  |  | var.capitalize() |  |
|  |  | var.capitalize() |  |
|  |  | var.capitalize() |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

To put it lightly, tuples are lists which can't be edited. Once you create a tuple, you cannot edit it, it is immutable. Lists on the other hand are mutable, you can edit them, they work like the array object in JavaScript or PHP. You can add items, delete items from a list; but you can't do that to a tuple, tuples have a fixed size

Technically, mutability and the syntax are the only differences between lists and tuples in Python. As a consequence of the mutability difference, tuples are easier on memory and processor, meaning you can gain some performance optimization with the use of tuples in the right places.

But there is an even more significant difference between the two - a philosophical one. I call it philosophical because the python interpreter cannot enforce it.

You might have read somewhere that tuples are heterogenous and lists are homogenous, forget that phrasing because it can be confusing and misleading. The better way to look at it is: "tuples are best used as heterogenous collections and lists are best used as homogenous collections". By heterogenous, the items contained in a tuple express different concepts, while the items in a list they all express the same concept or belong to the same category. Warning: try not to think of the homogeneity in terms of data type, think of them in terms of ideas they express or categories they belong to. Because of the heterogeneity, tuples are semantic - they have an inherent meaning to them.

# Functions:

* Functions consist block of statements which are used to perform some particular task.
* Functions are reusable and recursive.
* We can call functions whenever and whereever.
* By using ‘def’ key word we can create a function.
* Every function should return some thing by using return keyword.

# Syntax:

Def function\_name(parameter1,parameter2.......):

............

............ statements

............

Return....

Example:

1)def sum(a,b)

C=a+b called function

Return c

Print sum(10,20) **callling function**

Output:30

2)def sum(a,b):

Pass

Advanced default parmts:-

* To avoid the problem of unlimited parameters passing we will use advanced default parameters.

It uses two types of “arguments”

1.\*args

2.\*\*kwargs

1.\*args:-

* It will save the values which are passed by the user in the format of “tuple”.

2.\*\*kwargs:-

* This will save the values which are passed by the user in the format of “dictionary”.

Task

1)get the sum of all elements.

def get\_sum(a):

x=0

for i in a:

x=x+i

print x

return x

print get\_sum(range(1,11))

output:55

2)get the multiplication of all the elements.

def get\_sum(a):

x=1

for i in a:

x=x\*i

return x

print get\_sum(range(1,11))

output: 3628800

3)get all the even numbers.

def sum(a):

x=0

print a

for i in a:

x=x+i

return x

print sum(range(0,12,2))

output: [0, 2, 4, 6, 8, 10]

30

4)get all the odd numbers.

def sum(a):

print a

y=0

for i in a:

y=y+i

return y

print sum(range(1,12,2))

output: [1, 3, 5, 7, 9, 11]

36

# Method overloading:

* Two or more functions having a same function name with define signature is known as method overloading function.
* To achieve method overloading python providing default parameter.

Task

1)perform operations on even and odd .

def get\_sum\_even\_odd(l):

even\_list=[]

odd\_list=[]

even\_sum=0

odd\_sum=0

for i in l:

if i%2==0:

even\_list.append(i)

even\_sum=even\_sum+i

else:

odd\_list.append(i)

odd\_sum=odd\_sum+i

return even\_list,odd\_list,even\_sum,odd\_sum

print get\_sum\_even\_odd(range(1,11))

output: ([2, 4, 6, 8, 10], [1, 3, 5, 7, 9], 30, 25)

# programs:

1)check polyndrome.

def is\_palindrome(n):

for i in range(0,len(n)-1):

j = i + 1

if n[i] != n[-j]:

return False

return True

print is\_palindrome('mom')

output:True

2)fibonacci series.

3)prime numbers.

4)factorial

Functional programming tools

# 1)filter:

* It is used to filter the list of values..
* Input will be a list and output also a list.

# Syntax:

Filter(function\_name,arguments)

Example:

1) def get\_even(n):

return n%2==0

print filter(get\_even,range(10))

output: [0, 2, 4, 6, 8]

2) def get\_even(n):

return n%2==1

print filter(get\_even,range(10))

output: [1, 3, 5, 7, 9]

# 2)map:-

* It is used perform the manipulation operations.
* The input will be a list and output also a list.
* Map won’t take condition operations.

# Syntax:-

Map(function\_name,arguments)

Example:

1) def get\_even(n):

return n\*2

print map(get\_even,range(10))

output: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

# 3)reduce:-

* It is used to perform any mathmatical operations on each and every element of list.
* The input will be a list but output will be a single.

# Syntax:

Reduce(function\_name,input list)

Def function\_name(p1,p2)

Reduce p1()p2

Example:

1) def get\_sum(a,b):

return a+b

print reduce(get\_sum,range(1,10))

output:45

# 4)zip:-

* It is used to create a dictionary from two lists.

# Syntax:

Zip(list1,list2)

Example:

1) d=dict(zip(['a','b','c','d'],[1,2,3,4]))

print d

output: {'a': 1, 'c': 3, 'b': 2, 'd': 4}

Type coversion

# 1)list:

* This method is used to convert group of elements into a list format.

# Syntax:

List (group of elements)

Example:

1) li=list('gopal')

print li

li=list([1,2,3,4])

print li

li=list((1,2,3))

print li

li=list({'a':1,'b':2})

print li

output: ['g', 'o', 'p', 'a', 'l']

[1, 2, 3, 4]

[1, 2, 3]

['a', 'b']

# 2)tuple:-

* This method is used to convert group of elements into a tuple format.

# Syntax:

tuple(group of elements)

Example:

1) tu=tuple('gopal')

print tu

tu=tuple([1,2,3,4])

print tu

tu=tuple((1,2,3))

print tu

tu=tuple({'a':1,'b':2})

print tu

output: ('g', 'o', 'p', 'a', 'l')

(1, 2, 3, 4)

(1, 2, 3)

('a', 'b')

# 3)dict:

* It is used to convert a list of items into dictionary.

# Syntax:

Dict([(k,v),(k,v),(k,v)])

Example:

1) d=dict([('a',1),('b',2)])

print d

output: {'a': 1, 'b': 2}

2)e=dict(zip([1,2,3],[2,3,4]))

print e

output: {1: 2, 2: 3, 3: 4}

list comprehension

* It is concise way(easyer way)to create a list based on the requirement.
* This is advanced mothodology for map,filter,loops and conditions.
* The output of list comprehension is always a list.

# Syntax:

[ expression for name in list ]

Examples:

1) print [i for i in range(100)]

**Output:** **[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99]**

2) print [i\*2 for i in range(1,11)]

Output: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

3) l1=[1,2,3]

l2=[1,2,3]

print [l1[i]+l2[i] for i in range(len(l1))]

output: [2, 4, 6]

lambda functions

* Lambda functions are one line in code and these are not reusable so the scope of lambda functions is very very less in python.
* Lambda keywords is used to create a lambda functions.

# Syntax:

Function\_name=lambda var1,var2=var1\*var2

Modules

* Every python file treated has a module.
* To access the module we can used the command import.
* Two types of modules are there in python

1)system modules

2)user defined modules

* The main purpuse of using modules is to reuse the complete python file with the functionality.

# Importing modules

# Type 1:

Import math

Print math.sqrt(25)

# Type 2:

From math import sqrt

print sqrt(25)

# user defined modules

* Create a .py file

Functionns.py

|  |
| --- |
| def sum(a,b):  Return a+b  def mul(a,b):  Return a\*b  def div(a,b):  Return a/b |

* Take new file in that we write the program.

Import functions from functions import \*

Print sum(10,20) print sum(10,20)

Output:30 print mul(10,20)

Print div(20,10)

Output: 30

200,2

Files:

* we can work on any type of files in python by using file operations.
* Python has inbuilt mechanisum (method) to open,read,write and closing file.
* By using open method we can open a file here open takes two parameters.

1. File name with path
2. Read mode

* If we are not providing read mode parameter by default the mode will be in read.

# Modes

# Read(‘r’):-

* Used only for reading the files.
* The file shouid need to exist.
* We cannot do any operations which affects the file.

# Write(‘w’):-

* Write is used to add the data or text to a particular file.

# Syntax:

Open(‘new.txt’,’w’)

# Read functions

1. Read()
2. Readline()
3. Readlines()

# Write functions

1. Write()
2. Writelines()