Python®

Notes for Professionals

Chapter 2: Python Data Types

Section 2.1: String Data Type

Section 2.2: Set Data Types

Section 2.3: Numbers data type

Chapter 29: Basic Input and Output

Section 29.1: Using the print function

Section 29.2: Input from a file

Chapter 40: String Formatting

Section 40.1: Basics of String Formatting

700+ pages

of professional hints and tricks

GoalKicker.com
Free Programming Books

Disclaimer
This is an unofficial free book created for educational purposes and is not affiliated with official Python® group(s) or company(s).
All trademarks and registered trademarks are the property of their respective owners

TABLE OF CONTENTS

GETTING STARTED WITH PYTHONLANGUAGE

Computing

Written by : EL BAROUDI Marouane

Section 1.1: Getting StartedPython is a widely used high-level programming language for general-purpose programming, created by Guido vanRossum and first released in 1991. Python features a dynamic type system and automatic memory managementand supports multiple programming paradigms, including object-oriented, imperative, functional programming,and procedural styles. It has a large and comprehensive standard library.Two major versions of Python are currently in active use:Python 3.x is the current version and is under active development.Python 2.x is the legacy version and will receive only security updates until 2020. No new features will beimplemented. Note that many projects still use Python 2, although migrating to Python 3 is getting easier.You can download and install either version of Python here. See Python 3 vs. Python 2 for a comparison betweenthem. In addition, some third-parties offer re-packaged versions of Python that add commonly used libraries andother features to ease setup for common use cases, such as math, data analysis or scientific use. See the list at theofficial site.Verify if Python is installedTo confirm that Python was installed correctly, you can verify that by running the following command in yourfavorite terminal (If you are using Windows OS, you need to add path of python to the environment variable beforeusing it in command prompt):$ python --version GoalKicker.com – Python® Notes for Professionals3Python 3.xVersion ? 3.0If you have Python 3 installed, and it is your default version (see Troubleshooting for more details) you should seesomething like this:$ python --versionPython 3.6.0Python 2.xVersion ? 2.7If you have Python 2 installed, and it is your default version (see Troubleshooting for more details) you should seesomething like this:$ python --versionPython 2.7.13If you have installed Python 3, but $ python --version outputs a Python 2 version, you also have Python 2installed. This is often the case on MacOS, and many Linux distributions. Use $ python3 instead to explicitly use thePython 3 interpreter.Hello, World in Python using IDLEIDLE is a simple editor for Python, that comes bundled with Python.How to create Hello, World program in

IDLEOpen IDLE on your system of choice.In older versions of Windows, it can be found at All Programs under the Windows menu.In Windows 8+, search for IDLE or find it in the apps that are present in your system.On Unix-based (including Mac) systems you can open it from the shell by typing $ idlepython_file.py.It will open a shell with options along the top.In the shell, there is a prompt of three right angle brackets:>>>Now write the following code in the prompt:>>>print("Hello, World")Hit Enter .>>>print("Hello, World")Hello, WorldHello World Python fileCreate a new file hello.py that contains the following line:Python 3.xVersion ? 3.0print('Hello, World')Python 2.xVersion ? 2.6You can use the Python 3 print function in Python 2 with the following import statement: GoalKicker.com – Python® Notes for Professionals4from__future__import print_functionPython 2 has a number of functionalities that can be optionally imported from Python 3 using the __future__module, as discussed here.Python 2.xVersion ? 2.7If using Python 2, you may also type the line below. Note that this is not valid in Python 3 and thus notrecommended because it reduces cross-version code compatibility.print'Hello, World'In your terminal, navigate to the directory containing the file hello.py.Type python hello.py, then hit the Enter key.$ python hello.pyHello, WorldYou should see Hello, World printed to the console.You can also substitute hello.py with the path to your file. For example, if you have the file in your home directoryand your user is "user" on Linux, you can type python /home/user/hello.py.Launch an interactive Python shell

GETTING STARTED WITH PYTHONLANGUAGE

Computing

Written by : EL BAROUDI Marouane

Section 1.1: Getting StartedPython is a widely used high-level programming language for general-purpose programming, created by Guido vanRossum and first released in 1991. Python features a dynamic type system and automatic memory managementand supports multiple programming paradigms, including object-oriented, imperative, functional programming,and procedural styles. It has a large and comprehensive standard library.Two major versions of Python are currently in active use:Python 3.x is the current version and is under active development.Python 2.x is the legacy version and will receive only security updates until 2020. No new features will beimplemented. Note that many projects still use Python 2, although migrating to Python 3 is getting easier.You can download and install either version of Python here. See Python 3 vs. Python 2 for a comparison betweenthem. In addition, some third-parties offer re-packaged versions of Python that add commonly used libraries andother features to ease setup for common use cases, such as math, data analysis or scientific use. See the list at theofficial site.Verify if Python is installedTo confirm that Python was installed correctly, you can verify that by running the following command in yourfavorite terminal (If you are using Windows OS, you need to add path of python to the environment variable beforeusing it in command prompt):$ python --version GoalKicker.com – Python® Notes for Professionals3Python 3.xVersion ? 3.0If you have Python 3 installed, and it is your default version (see Troubleshooting for more details) you should seesomething like this:$ python --versionPython 3.6.0Python 2.xVersion ? 2.7If you have Python 2 installed, and it is your default version (see Troubleshooting for more details) you should seesomething like this:$ python --versionPython 2.7.13If you have installed Python 3, but $ python --version outputs a Python 2 version, you also have Python 2installed. This is often the case on MacOS, and many Linux distributions. Use $ python3 instead to explicitly use thePython 3 interpreter.Hello, World in Python using IDLEIDLE is a simple editor for Python, that comes bundled with Python.How to create Hello, World program in

IDLEOpen IDLE on your system of choice.In older versions of Windows, it can be found at All Programs under the Windows menu.In Windows 8+, search for IDLE or find it in the apps that are present in your system.On Unix-based (including Mac) systems you can open it from the shell by typing $ idlepython_file.py.It will open a shell with options along the top.In the shell, there is a prompt of three right angle brackets:>>>Now write the following code in the prompt:>>>print("Hello, World")Hit Enter .>>>print("Hello, World")Hello, WorldHello World Python fileCreate a new file hello.py that contains the following line:Python 3.xVersion ? 3.0print('Hello, World')Python 2.xVersion ? 2.6You can use the Python 3 print function in Python 2 with the following import statement: GoalKicker.com – Python® Notes for Professionals4from__future__import print_functionPython 2 has a number of functionalities that can be optionally imported from Python 3 using the __future__module, as discussed here.Python 2.xVersion ? 2.7If using Python 2, you may also type the line below. Note that this is not valid in Python 3 and thus notrecommended because it reduces cross-version code compatibility.print'Hello, World'In your terminal, navigate to the directory containing the file hello.py.Type python hello.py, then hit the Enter key.$ python hello.pyHello, WorldYou should see Hello, World printed to the console.You can also substitute hello.py with the path to your file. For example, if you have the file in your home directoryand your user is "user" on Linux, you can type python /home/user/hello.py.Launch an interactive Python shell

PYTHON DATA TYPES

Computing

Written by : EL BAROUDI Marouane

Data types are nothing but variables you use to reserve some space in memory. Python variables do not need anexplicit declaration to reserve memory space. The declaration happens automatically when you assign a value to avariable.Section 2.1: String Data TypeString are identified as a contiguous set of characters represented in the quotation marks. Python allows for eitherpairs of single or double quotes. Strings are immutable sequence data type, i.e each time one makes any changesto a string, completely new string object is created.a_str ='Hello World'print(a_str)#output will be whole string. Hello Worldprint(a_str[0])#output will be first character. Hprint(a_str[0:5])#output will be first five characters. HelloSection 2.2: Set Data TypesSets are unordered collections of unique objects, there are two types of set:Sets - They are mutable and new elements can be added once sets are defined1.basket ={'apple','orange','apple','pear','orange','banana'}print(basket)# duplicates will be removed>{'orange','banana','pear','apple'}a =set('abracadabra')print(a)# unique letters in a>{'a','r','b','c','d'}a.add('z')print(a)>{'a','c','r','b','z','d'}Frozen Sets - They are immutable and new elements cannot added after its defined.2.b =frozenset('asdfagsa')print(b)>frozenset({'f','g','d','a','s'})cities =frozenset(["Frankfurt","Basel","Freiburg"])print(cities)>frozenset({'Frankfurt','Basel','Freiburg'})Sec 2.3: Numbers data typeNumbers have four types in Python. Int, float, complex, and long.int_num =10#int valuefloat_num =10.2#float valuecomplex_num = 3.14j #complex valuelong_num = 1234567L #long value GoalKicker.com – Python® Notes for Professionals34Section 2.4: List Data TypeA list contains items separated by commas and enclosed within square brackets [].lists are almost similar to arraysin C. One difference is that all the items belonging to a list can be of different data type.list=[123,'abcd',10.2,'d']#can be an array of any data type or single data type.list1 =['hello','world']print(list)#will output whole list. [123,'abcd',10.2,'d']print(list[0:2])#will output first two element of list. [123,'abcd']print(list1 * 2)#will gave list1 two times. ['hello','world','hello','world']print(list + list1)#will gave concatenation of both the lists.[123,'abcd',10.2,'d','hello','world']Section 2.5: Dictionary Data TypeDictionary consists of key-value pairs. It is enclosed by curly braces {} and values can be assigned and accessedusing square brackets[].dic={'name':'red','age':10}print(dic)#will output all the key-value pairs. {'name':'red','age':10}print(dic['name'])#will output only value with 'name' key. 'red'print(dic.values())#will output list of values in dic. ['red',10]print(dic.keys())#will output list of keys. ['name','age']Section 2.6: Tuple Data TypeLists are enclosed in brackets [ ] and their elements and size can be changed, while tuples are enclosed inparentheses ( ) and cannot be updated. Tuples are immutable.tuple=(123,'hello')tuple1 =('world')print(tuple)#will output whole tuple. (123,'hello')print(tuple[0])#will output first value. (123)print(tuple + tuple1)#will output (123,'hello','world')tuple[1]='update'#this will give you error.

PYTHON DATA TYPES

Computing

Written by : EL BAROUDI Marouane

Data types are nothing but variables you use to reserve some space in memory. Python variables do not need anexplicit declaration to reserve memory space. The declaration happens automatically when you assign a value to avariable.Section 2.1: String Data TypeString are identified as a contiguous set of characters represented in the quotation marks. Python allows for eitherpairs of single or double quotes. Strings are immutable sequence data type, i.e each time one makes any changesto a string, completely new string object is created.a_str ='Hello World'print(a_str)#output will be whole string. Hello Worldprint(a_str[0])#output will be first character. Hprint(a_str[0:5])#output will be first five characters. HelloSection 2.2: Set Data TypesSets are unordered collections of unique objects, there are two types of set:Sets - They are mutable and new elements can be added once sets are defined1.basket ={'apple','orange','apple','pear','orange','banana'}print(basket)# duplicates will be removed>{'orange','banana','pear','apple'}a =set('abracadabra')print(a)# unique letters in a>{'a','r','b','c','d'}a.add('z')print(a)>{'a','c','r','b','z','d'}Frozen Sets - They are immutable and new elements cannot added after its defined.2.b =frozenset('asdfagsa')print(b)>frozenset({'f','g','d','a','s'})cities =frozenset(["Frankfurt","Basel","Freiburg"])print(cities)>frozenset({'Frankfurt','Basel','Freiburg'})Sec 2.3: Numbers data typeNumbers have four types in Python. Int, float, complex, and long.int_num =10#int valuefloat_num =10.2#float valuecomplex_num = 3.14j #complex valuelong_num = 1234567L #long value GoalKicker.com – Python® Notes for Professionals34Section 2.4: List Data TypeA list contains items separated by commas and enclosed within square brackets [].lists are almost similar to arraysin C. One difference is that all the items belonging to a list can be of different data type.list=[123,'abcd',10.2,'d']#can be an array of any data type or single data type.list1 =['hello','world']print(list)#will output whole list. [123,'abcd',10.2,'d']print(list[0:2])#will output first two element of list. [123,'abcd']print(list1 * 2)#will gave list1 two times. ['hello','world','hello','world']print(list + list1)#will gave concatenation of both the lists.[123,'abcd',10.2,'d','hello','world']Section 2.5: Dictionary Data TypeDictionary consists of key-value pairs. It is enclosed by curly braces {} and values can be assigned and accessedusing square brackets[].dic={'name':'red','age':10}print(dic)#will output all the key-value pairs. {'name':'red','age':10}print(dic['name'])#will output only value with 'name' key. 'red'print(dic.values())#will output list of values in dic. ['red',10]print(dic.keys())#will output list of keys. ['name','age']Section 2.6: Tuple Data TypeLists are enclosed in brackets [ ] and their elements and size can be changed, while tuples are enclosed inparentheses ( ) and cannot be updated. Tuples are immutable.tuple=(123,'hello')tuple1 =('world')print(tuple)#will output whole tuple. (123,'hello')print(tuple[0])#will output first value. (123)print(tuple + tuple1)#will output (123,'hello','world')tuple[1]='update'#this will give you error.

INDENTATION

Computing

Written by : EL BAROUDI Marouane

For Python, Guido van Rossum based the grouping of statements on indentation. The reasons for this are explainedin the first section of the "Design and History Python FAQ". Colons, :, are used to declare an indented code block,such as the following example:class ExampleClass:#Every function belonging to a class must be indented equallydef__init__(self): name ="example"def someFunction(self, a):#Notice everything belonging to a function must be indentedif a >5:returnTrueelse:returnFalse#If a function is not indented to the same level it will not be considers as part of the parent classdef separateFunction(b):for i in b:#Loops are also indented and nested conditions start a new indentationif i ==1:returnTruereturnFalseseparateFunction([2,3,5,6,1])Spaces or Tabs?The recommended indentation is 4 spaces but tabs or spaces can be used so long as they are consistent. Do notmix tabs and spaces in Python as this will cause an error in Python 3 and can causes errors in Python 2.Section 3.2: How Indentation is ParsedWhitespace is handled by the lexical analyzer before being parsed.The lexical analyzer uses a stack to store indentation levels. At the beginning, the stack contains just the value 0,which is the leftmost position. Whenever a nested block begins, the new indentation level is pushed on the stack,and an "INDENT" token is inserted into the token stream which is passed to the parser. There can never be morethan one "INDENT" token in a row (IndentationError).When a line is encountered with a smaller indentation level, values are popped from the stack until a value is on topwhich is equal to the new indentation level (if none is found, a syntax error occurs). For each value popped, a"DEDENT" token is generated. Obviously, there can be multiple "DEDENT" tokens in a row.The lexical analyzer skips empty lines (those containing only whitespace and possibly comments), and will nevergenerate either "INDENT" or "DEDENT" tokens for them.At the end of the source code, "DEDENT" tokens are generated for each indentation level left on the stack, until justthe 0 is left.

INDENTATION

Computing

Written by : EL BAROUDI Marouane

For Python, Guido van Rossum based the grouping of statements on indentation. The reasons for this are explainedin the first section of the "Design and History Python FAQ". Colons, :, are used to declare an indented code block,such as the following example:class ExampleClass:#Every function belonging to a class must be indented equallydef__init__(self): name ="example"def someFunction(self, a):#Notice everything belonging to a function must be indentedif a >5:returnTrueelse:returnFalse#If a function is not indented to the same level it will not be considers as part of the parent classdef separateFunction(b):for i in b:#Loops are also indented and nested conditions start a new indentationif i ==1:returnTruereturnFalseseparateFunction([2,3,5,6,1])Spaces or Tabs?The recommended indentation is 4 spaces but tabs or spaces can be used so long as they are consistent. Do notmix tabs and spaces in Python as this will cause an error in Python 3 and can causes errors in Python 2.Section 3.2: How Indentation is ParsedWhitespace is handled by the lexical analyzer before being parsed.The lexical analyzer uses a stack to store indentation levels. At the beginning, the stack contains just the value 0,which is the leftmost position. Whenever a nested block begins, the new indentation level is pushed on the stack,and an "INDENT" token is inserted into the token stream which is passed to the parser. There can never be morethan one "INDENT" token in a row (IndentationError).When a line is encountered with a smaller indentation level, values are popped from the stack until a value is on topwhich is equal to the new indentation level (if none is found, a syntax error occurs). For each value popped, a"DEDENT" token is generated. Obviously, there can be multiple "DEDENT" tokens in a row.The lexical analyzer skips empty lines (those containing only whitespace and possibly comments), and will nevergenerate either "INDENT" or "DEDENT" tokens for them.At the end of the source code, "DEDENT" tokens are generated for each indentation level left on the stack, until justthe 0 is left.

COMMENTS AND DOCUMENTATION

Computing

Written by : EL BAROUDI Marouane

Section 4.1: Single line, inline and multiline commentsComments are used to explain code when the basic code itself isn't clear.Python ignores comments, and so will not execute code in there, or raise syntax errors for plain English sentences.Single-line comments begin with the hash character (#) and are terminated by the end of line.Single line comment:# This is a single line comment in PythonInline comment:print("Hello World")# This line prints "Hello World"Comments spanning multiple lines have """ or ''' on either end. This is the same as a multiline string, butthey can be used as comments:"""This type of comment spans multiple lines.These are mostly used for documentation of functions, classes and modules."""Section 4.2: Programmatically accessing docstringsDocstrings are - unlike regular comments - stored as an attribute of the function they document, meaning that youcan access them programmatically.An example functiondef func():"""This is a function that does nothing at all"""returnThe docstring can be accessed using the __doc__ attribute:print(func.__doc__)This is a function that does nothing at allhelp(func)Help on function func in module __main__:func()This is a function that does nothing at allAnother example function GoalKicker.com – Python® Notes for Professionals38function.__doc__ is just the actual docstring as a string, while the help function provides general informationabout a function, including the docstring. Here's a more helpful example:def greet(name, greeting="Hello"):"""Print a greeting to the user `name` Optional parameter `greeting` can change what they're greeted with."""print("{} {}".format(greeting, name))help(greet)Help on function greet in module __main__:greet(name, greeting='Hello')Print a greeting to the user nameOptional parameter greeting can change what they're greeted with.Advantages of docstrings over regular commentsJust putting no docstring or a regular comment in a function makes it a lot less helpful.def greet(name, greeting="Hello"):# Print a greeting to the user `name`# Optional parameter `greeting` can change what they're greeted with.print("{} {}".format(greeting, name))print(greet.__doc__)Nonehelp(greet)Help on function greet in module main:greet(name, greeting='Hello')Section 4.3: Write documentation using docstringsA docstring is a multi-line comment used to document modules, classes, functions and methods. It has to be thefirst statement of the component it describes.def hello(name):"""Greet someone. Print a greeting ("Hello") for the person with the given name. """print("Hello "+name)class Greeter:"""An object used to greet people.

COMMENTS AND DOCUMENTATION

Computing

Written by : EL BAROUDI Marouane

Section 4.1: Single line, inline and multiline commentsComments are used to explain code when the basic code itself isn't clear.Python ignores comments, and so will not execute code in there, or raise syntax errors for plain English sentences.Single-line comments begin with the hash character (#) and are terminated by the end of line.Single line comment:# This is a single line comment in PythonInline comment:print("Hello World")# This line prints "Hello World"Comments spanning multiple lines have """ or ''' on either end. This is the same as a multiline string, butthey can be used as comments:"""This type of comment spans multiple lines.These are mostly used for documentation of functions, classes and modules."""Section 4.2: Programmatically accessing docstringsDocstrings are - unlike regular comments - stored as an attribute of the function they document, meaning that youcan access them programmatically.An example functiondef func():"""This is a function that does nothing at all"""returnThe docstring can be accessed using the __doc__ attribute:print(func.__doc__)This is a function that does nothing at allhelp(func)Help on function func in module __main__:func()This is a function that does nothing at allAnother example function GoalKicker.com – Python® Notes for Professionals38function.__doc__ is just the actual docstring as a string, while the help function provides general informationabout a function, including the docstring. Here's a more helpful example:def greet(name, greeting="Hello"):"""Print a greeting to the user `name` Optional parameter `greeting` can change what they're greeted with."""print("{} {}".format(greeting, name))help(greet)Help on function greet in module __main__:greet(name, greeting='Hello')Print a greeting to the user nameOptional parameter greeting can change what they're greeted with.Advantages of docstrings over regular commentsJust putting no docstring or a regular comment in a function makes it a lot less helpful.def greet(name, greeting="Hello"):# Print a greeting to the user `name`# Optional parameter `greeting` can change what they're greeted with.print("{} {}".format(greeting, name))print(greet.__doc__)Nonehelp(greet)Help on function greet in module main:greet(name, greeting='Hello')Section 4.3: Write documentation using docstringsA docstring is a multi-line comment used to document modules, classes, functions and methods. It has to be thefirst statement of the component it describes.def hello(name):"""Greet someone. Print a greeting ("Hello") for the person with the given name. """print("Hello "+name)class Greeter:"""An object used to greet people.

PYTHON ANTI-PATTERNS

Computing

Written by : EL BAROUDI Marouane

Section 199.1: Overzealous except clauseExceptions are powerful, but a single overzealous except clause can take it all away in a single line.try: res = get_result() res = res[0] log('got result: %r' % res) except: if not res: res = '' print('got exception')This example demonstrates 3 symptoms of the antipattern:The except with no exception type (line 5) will catch even healthy exceptions, including KeyboardInterrupt.1.That will prevent the program from exiting in some cases.The except block does not reraise the error, meaning that we won't be able to tell if the exception came from2.within get_result or because res was an empty list.Worst of all, if we were worried about result being empty, we've caused something much worse. If3.get_result fails, res will stay completely unset, and the reference to res in the except block, will raiseNameError, completely masking the original error.Always think about the type of exception you're trying to handle. Give the exceptions page a read and get a feel forwhat basic exceptions exist.Here is a fixed version of the example above:import traceback try: res = get_result() except Exception: log_exception(traceback.format_exc()) raise try: res = res[0]except IndexError: res = '' log('got result: %r' % res)We catch more specific exceptions, reraising where necessary. A few more lines, but infinitely more correct.Section 199.2: Looking before you leap with processor-intensive functionA program can easily waste time by calling a processor-intensive function multiple times.For example, take a function which looks like this: it returns an integer if the input value can produce one, elseNone:def intensive_f(value): # int -> Optional[int]# complex, and time-consuming codeif process_has_failed:returnNonereturn integer_outputAnd it could be used in the following way:x =5if intensive_f(x)isnotNone:print(intensive_f(x) / 2)else:print(x,"could not be processed")print(x)Whilst this will work, it has the problem of calling intensive_f, which doubles the length of time for the code torun. A better solution would be to get the return value of the function beforehand

PYTHON ANTI-PATTERNS

Computing

Written by : EL BAROUDI Marouane

Section 199.1: Overzealous except clauseExceptions are powerful, but a single overzealous except clause can take it all away in a single line.try: res = get_result() res = res[0] log('got result: %r' % res) except: if not res: res = '' print('got exception')This example demonstrates 3 symptoms of the antipattern:The except with no exception type (line 5) will catch even healthy exceptions, including KeyboardInterrupt.1.That will prevent the program from exiting in some cases.The except block does not reraise the error, meaning that we won't be able to tell if the exception came from2.within get_result or because res was an empty list.Worst of all, if we were worried about result being empty, we've caused something much worse. If3.get_result fails, res will stay completely unset, and the reference to res in the except block, will raiseNameError, completely masking the original error.Always think about the type of exception you're trying to handle. Give the exceptions page a read and get a feel forwhat basic exceptions exist.Here is a fixed version of the example above:import traceback try: res = get_result() except Exception: log_exception(traceback.format_exc()) raise try: res = res[0]except IndexError: res = '' log('got result: %r' % res)We catch more specific exceptions, reraising where necessary. A few more lines, but infinitely more correct.Section 199.2: Looking before you leap with processor-intensive functionA program can easily waste time by calling a processor-intensive function multiple times.For example, take a function which looks like this: it returns an integer if the input value can produce one, elseNone:def intensive_f(value): # int -> Optional[int]# complex, and time-consuming codeif process_has_failed:returnNonereturn integer_outputAnd it could be used in the following way:x =5if intensive_f(x)isnotNone:print(intensive_f(x) / 2)else:print(x,"could not be processed")print(x)Whilst this will work, it has the problem of calling intensive_f, which doubles the length of time for the code torun. A better solution would be to get the return value of the function beforehand