CHAPTER 2 - Python Data Types

|  |  |  |
| --- | --- | --- |
| **DATA TYPES** | | |
| x = 5  #variable | PI = 3.14 # By convention, constant names are written in UPPERCASE | |
| x = 5 + 5 | | x = 5 - 4 |
| x = 3\*2 | | X = 3\*\*2 |
| x = 3 / 2  #float value | | x = 3 // 2  #not a float value |
| x = 3 % 2  #modulo operator | | X = 4.42  #float |
| X = 4 > 3 #bool  < > == != | | And, or, not  #logical operators |
| x = “Hello World” #String | | |
| x in s #True if object x is in list s | | |
| x not in s | | |
| s\*n, n\*s #Concatenation of n copies of s | | |
| s[i] #The index operator, item at index I of list s | | |
| x[], x[start\_index:end\_index]  #slicing operation | | |
| str[::-1] #prints the string in reverse | | |
| x = [ “1”, “2”,”3”,”4” ] #List | | |
| x = ( “1”, 2, 3, [4], 6 ) #Tuple | | |

WORKING WITH LIBRARIES

* import math #import the entire library
* import numpy as np #import with and alias
* from math import pi, sin #import specific functions
* pip install request #install a library

SOME BUILT-IN FUNCTIONS

* Len( list ) #length of string list
* Min( list ) #smallest item in list
* Max( list ) #largest item in list
* Sum( list ) #sum of items in list
* Abs( x ) #absolute value of x
* Type() #Evaluate the type of the object
* Print(“hello world”)

SOME LIST METHODS

* List.append( item ) #Adds item to the end of list
* List.count( item ) #Returns the number of ocurrences of item in list
* List.index( index, item ) #inserts item into list just before index
* List.pop() #Removes last item in the list
* List.remove( item ) #Removes first occurrence of item in the list
* List. Reverse() #reverses the order of items in the list
* List.sort() #sorts the list
* List.extend([4, 5,6])
* List.copy()
* List.clear()

#Strings and lists both are sequences of objects.

CHAPTER 3 – Imperative Programming

print() #This function prints, within the interactive shell, whatever argument is given to it.

input() #built-in function that prompts the user for input and returns the entered data as a string.

eval() #executes a string or compiled code as a Python expression and returns the result.

|  |  |  |
| --- | --- | --- |
| EXECUTION CONTROL ESTRUCTURES | USER-DEFINED FUNCTION | |
| THE IF STATEMENT | Def square(x, y):  Return x\*\*2 + y\*\*2 | |
| If <condition>:  <indented code block 1>  Elif:  <indented code block 2>  Elif:  <indented code block 3>  else:  <indented code block last>  <non-indented statement> |
| PRINT() VERSUS RETURN | |
| #functions without return can’t be used as expressions  3 \* f(2) + 1 | |
| ITERATION STRUCTURES |
| For <variable> in <sequence>:  <indented code block>  <non-indented code block>  For char in name:  Print(char) |
| DOCSTRINGS | |
| # The comment that can be viewed using function help() in documentation  ‘This is a docstring’ | |
| NESTING CONTROL FLOW STRUCTURES |
| For c in phrase:  If c in ‘’aeiouAEIOU’’:  Print(c) |
| COMMENTS | |
| #This is a comment | |
| FUNCTION RANGE() | PYTHON VARIABLE AND ASSIGNMENTS | |
| For I in range(5):  Print(i ) | <variable> =<expression> | |
| For I in range(2, 5):  Print(i) | IMMUTABLE DATA TYPE | MUTABLE DATA TYPE |
| For I in range(1, 14, 3):  print(i) | * Numbers * Booleans * Strings * Tuples | * Lists * Dictionaries * Sets |

CHAPTER 4 – Text Data, Files, and Exceptions

4.1 STRINGS, REVISITED

#the escape sequence \' or \" is used to indicate that a quote is not the string delimiter but is part of the string value.

string = 'I\'m "sick"'

|  |  |
| --- | --- |
| HE INDEXING OPERATOR, REVISITED  #The last example shows how to get  a slice using negative indexes.  s = 'hello' | 4.2 Formatted Output  FUNCTION PRINT()  #Function print() can take an arbitrary number of input objects  We can insert semicolons between values instead of blank spaces  print(**4, 1.5,** "hello"**,** sep=";")  print(**4, 1.5,** "hello"**,** end=";")  #The argument sep=";" specifies that semicolons should be inserted to separate the printed values of n, r, and name.  #En resumen, sep controla cómo se separan los elementos dentro de una misma llamada a print(), mientras que end determina qué se imprime al final de la salida. Estos parámetros ofrecen flexibilidad en el formato de salida según tus necesidades específicas.  STRING METHOD FORMAT  '{0:5}:{1:5}:{2:5}'.format(h,min,s)  #The numbers 0, 1, and 2 explicitly indicate that the placeholders are for the first, second, and third arguments of the format() function call, respectively.  '{0:3},{1:5}'.format(12, 354)  Everything after the ':' specifies the formatting of the value  '{:ancho.precisión}'.format(número)   **ancho**: El número mínimo de caracteres que ocupará el número formateado.   **precisión**: El número de dígitos después del punto decimal. |
| SLICING LISTS  pets[:2]  ['goldfish', 'cat']    pets[-3:-1]  ['goldfish', 'cat'] |
|  |

the slicing operation.

In Python, x[start\_index:end\_index] refers to the slicing operation. This operation is used to get a subsequence from a sequence, such as a list, tuple, or string. Here’s how it works:

Basic Syntax:

The syntax for slicing is x[start:end], where start is the index where the slice starts, and end is the index where it ends (but is not included).

For example, if you have a list x = [0, 1, 2, 3, 4, 5] and you do x[2:4], you’ll get [2, 3].

Negative Indices:

You can also use negative indices to refer to elements from the end of the sequence.

For example, x[-3:-1] would return the elements from the third-to-last to the second-to-last.

Omitting Indices:

If you omit the start, Python assumes you want to start from the beginning (index 0).

If you omit the end, Python assumes you want to go to the end of the sequence.

For example, x[:3] returns the first three elements, and x[3:] returns all elements from index 3 to the end.

Step or Stride:

You can specify a third parameter step as in x[start:end:step], which determines the interval between elements in the subsequence.

For example, x[1:5:2] would return [1, 3], skipping every second element.

Slicing Strings:

Slicing also works with strings. If s = "Hello World", then s[1:4] would result in 'ell'.

Here’s a code example showing how slicing works on a list and a string:

|  |
| --- |
| **Python**  # Example list  numbers = [0, 1, 2, 3, 4, 5]  print(numbers[2:5]) # Output: [2, 3, 4]  # Example string  message = "Hello World"  print(message[5:10]) # Output: 'World' |