Curso de Python

A) El conocimiento del software es tan importante como saber leer

1. Lenguaje Python dentro del mundo del software

1. Curva de aprendizaje
2. Aprendizaje inclusivo
3. Construir cosas

2. Instalación del Python

1. Windows y Linux
2. Evite el infierno de versiones de Python

3. Terminal python

1. Entrar y salir del interprete
2. Resolver un problema aritmético
3. Usar el help()
   3.1 help(round)
   3.2 help("for")

B) ipython es ideal para la creación de prototipos

3.1. Interprete ipython

1. `<TAB> completacion`
   1.1 Atributos de Objetos
   1.2 Archivos y Directorios
2. Comando **%history**
3. Ejecutar cualquier comando del terminal
   3.1 Mostrar archivos
   3.2 Mostrar permisos

3.2. Aprender Python

C) Principio básico: aprendes mejor lo que haces todos los días

1. Crear directorio /bin
2. Ejecutar un script 20.1
   2.1 parar ejecución con exit()
3. Navegar por sistema de archivos con comandos cd y ls

4. Variables

D) Entender asignación

4.1. Números

1. Números Enteros
   1.1 Operaciones
   1.2 Divisiones y modulo
   1.3 Por lo general son representados en base 10
2. Números con decimales
   2.1 Notación exponencial
   2.2 Números decimales son aproximados

4.2. Cadenas de caracteres

1. Representación de texto en forma literal, entre comillas
2. Cada carácter tiene una representación en la Organización Unicode
3. learn_indexing.py

4.3. Boleamos

1. Tipo booleano: **True** / **False**
2. while-if-elif-else.py

4.4. None

1. **None** es evaluado como False

5. ipython como calculadora

1. 7 operaciones
2. Calculadora mas avanzada
   2.1 import math as m
   2.2 m.sin()

6. Ejemplo crear un modelo interactiva-mente

1. modelar 4 vueltas corriendo y 4 vueltas caminando
2. Ciclos de 6 minutos y tiempo total 24 minutos
3. sin(), abs(), exp()
4. matplotlib

7. Función interna print()

1. Imprime cualquier objeto
2. Salida **print()**
3. Carácter de nueva linea

8. Otros ambientes de Python

E) Prototipo en ipython y recolecto en vim

1. Crear /bin y otros subdirectorios
2. Scripts ejecutables
3. La Revolución del Cuaderno Jupyter (**jupyter notebook**)

9. Sintaxis de Python

1. Identación
2. La indexación de Python
   2.1 Comienza en cero
   2.2 Indexación todas las estructuras de datos
3. Comentarios
4. triple_quotes.py

10. Python estilo

1. pylint

11. Estructuras de datos

F) Hay un mundo ordenado y un mundo desordenado

11.1. Listas primero

G) Lista es la estructura principal de datos para mantener una colección ordenada de valores

1. Crear una lista
2. Modificar una lista
3. **range()**
4. Bucle sobre una lista
5. unicode_ranges.py
6. builtins.py

11.2. Cadenas de caracteres

1. Métodos
   1.1 .rstrip()
   1.2 .join() - metodos_cadenas.py
   1.3 .split()
2. Conversión entre números y cadenas
   2.1 str(), int(), float()
   2.2 type_conversion.py
3. Unicode
   3.1 interrogación invertida - chr(191)
   3.2 letra 'ň' - chr(328)

11.3. Tuplas

1. Inmutable
2. **sorted()** multilevel

11.4. Diccionarios

1. Que es un diccionario ?
2. Crear un diccionario
3. Recuperar un valor usando dic[**key**]
4. Métodos de un diccionario
   4.1 Lista de tuplas - dic.**items()**
   4.2 Lista de claves dic.**keys()**
   4.3 Lista de valores dic.**values()**
5. Bucle sobre el diccionario
6. Guardar diccionario a un archivo
   6.1 dict_menu.py

11.5. Conjuntos

H) Los conjuntos se usan como base en la fundación de la Matemática

1. Construcción de conjuntos
2. Métodos de conjuntos
3. Operaciones de conjuntos
4. set-operations.py

12. Bucles

I) En cada iteración del ciclo se obtienes un nuevo elemento para trabajar

1. Variable de iteración del bucle-for
2. Bucle sobre numeros enteros
3. Con y sin bucle **for** 20.5
4. **while** 20.4

13. Formatos para print()

1. Carácter de escape
2. Formato de salida **format()**
3. Formato de estilo C con operador %

13.1. f-string

1. Formato de f-string
2. Códigos de formato
3. Modificadores comunes ajustan el ancho del campo

14. Leyendo y escribiendo Archivos

1. open, para leer 20.2
2. open, para escribir 21.4
3. with open, bloque para leer 21.3
   3.1 Leer el archivo completo de una vez como una cadena
   3.2 Leer el archivo línea por línea iterando
4. Modismos comunes para escribir en un archivo
   4.1 Escribir una larga cadena de data
   4.2 Redirigir la función print()
5. Con el interprete abra un archivo y salte la primera linea

15. Objeto, métodos, funciones y espacio de nombres

J) Los objetos son contenedores inteligentes

1. Crear objeto mylist
2. Identificar objeto **type(mylist)**
3. Ver sus métodos
   3.1 **dir(mylist)**

4. Ver todas las funciones internas del python
   4.1 builtins.txt
   4.2 **len(mylist)**
   4.3 **help(mylist.apend)**
5. Un espacio de nombres es donde viven los métodos
   5.1 Los espacios de nombres son clave para evitar conflictos entre nombres
   5.2 El problema es que hay muchos métodos

16. Importando módulos

16.1. numpy

1. Posición de numpy en Python
2. numpy proporciona una plataforma para manejar Big Data de manera eficiente
3. numpy en el stack científico de Python
4. Cálculos multi-dimensionales
5. Las matrices son una subclase de los arreglos
6. import numpy as np 20.1
   6.1. Propedeutico numpy 1D
   6.2. Imagenes numpy 2D
   6.3. Doble corchete indica vector fila

16.2. Muchos modulos

1. import os 21.5
2. import sys 21.9
3. import time 21.8
4. import matplotlib.pyplot as plt 21.2
5. import argpaeser 21.11

17. Entrada por teclado

1. Entrada **input()**
2. Opciones 20.3 20.4

18. Sentencias if-elif-else

1. Condicionales if-else 20.3
2. ¿Cómo se indica el final de la estructura?
3. Operadores `<= >=`
4. Declaración de ruptura: break
5. Declaración: continue
6. comparison_operators.py

19. Funciones

19.1. Definiendo funciones

1. def nombre_funcion(argumentos):
2. docstring
   2.1 function_docstring.py
3. **return**
4. argumentos
   4.1 function_args.py
   4.2 function_kwargs.py

19.2. Llamando una function

1. Valores predeterminados para argumentos
2. lambda argumentos : expresión
3. map() 21.6
4. filter() 21.10

19.3. Utilizar funciones

K) Muchos recursos y funciones disponibles

1. Funciones clásicas
2. three-random.py

1. Nuestro propio modelo
2. `guess_a_number.py`

1. Cómo ajustar un polinomio a partir de un conjunto de puntos
2. fit-polynomial.py

19.4. Mejores practicas

1. Crear una función en un archivo
2. Cargar la función y ejecutarla
3. ipython -i function.py
4. convert_temp.py
5. main()
6. hello_nick.py

19.5. Errores de sintaxis y semántica

1. `grep Error texts/builtins.txt |wc`
2. FileNotFoundError: (21.3)
3. TypeError:
4. Manejar excepciones

   4.1 pcost.py

20. Ejemplos de scripts

20.1. scripts

1 script de un solo un comando

   11 test-foo.py

2 Mejores practicas

   21 Funcion main()

   22 hello_nick.py

20.2. Barajar una secuencia de líneas

1 shuffle.py

2 `seq -f 'line %.0f' 20 > texts/20lineas`

20.3. Cómo usar condicionales if-elif-else

1 if_conditional.py

2 dos preguntas usando `input()`

20.4. Bucle infinita

1 grep.py

2 while-if-elif-else.py

3 El comando se repite en un bucle 'while'

20.5. Con o sin bucle for

1 bash-commands.py

2 three-random.py

21. Ejemplos de programas

21.1. Velocidad terminal del paracaidista

1 paracaidista.py
2 input(), round(), str()
3 Uso del ciclo while

21.2. Modelar latidos del corazón al correr y caminar

1 modelar-ciclos.py
2 np.arange()
3 linewidth=, 'r–'
4 Como usar map() function
5 Como usar filter() function

21.3. Leer un archivo CSV

1 Descubrir columnas
2 Remplazar comas por puntos
3 Parser campos como fechas y punto flotante
4 linestyle=, marker=, color='r'

21.4. Escribir un archivo

1 builtins.py
2 pcost.py

21.5. Directorio actual

1 os_comando.py
2 analizar-imagenes.py

21.6. function map()

1 temperatures-map-function.py

21.7. function filter()

1 filter_vowels.py

21.8. import time

1 sleep.py

21.9. import sys

1 orders.py

21.10. import fileinput

1 orders2.py

21.11. import argparser

1 create_note.py

11 Argumentos posicionales vs Argumentos opcionales

21.12. from datetime import datetime

L) ¿Por qué programar con fechas y horas es un desafío?

1 Usos horarios
2 formatos de fechas
3 Maneras de instanciar datetime
4 instancias ingenuas vs instancias cocientes
5 UTC-datetime.py
6 from icalendar import Calendar, Event
7 ics_dateutil.py