**2do Bimestre**

**Programación:**

1. Concepto de programa.
2. Variables objetos con un datos, tipos de datos.-¿Constantes ?
   * Numéricos (int,float,complex) Strings (str) booleanos (True, False)
   * Mutables / inmutable
3. Operadores lógicos y matemáticos.
4. Estructuras de toma de decisiones.
5. Colecciones objetos con mas de un datos
   * listas, tuplas, diccionarios.
   * Mutables / inmutable
6. Estructuras de repeticiones.
7. Procedimientos y funciones.
8. Algoritmos y estructuras de datos básicas.

**Organigrama:**

**Librerías auxiliares**

* módulos externos
* Import random
  + - randint
    - choise
    - shuffle

**Funciones all y any.**

**Operadores ternarios**

* + if
  + else

**Colecciones por comprensión (comprehension).**

* + for
  + if
  + else
* listas y sets.
* diccionario.

**Funciones propias**.

* def return
  + Argumentos posicionales y por omisión
  + Parámetros arbitrarios en Python
  + Funciones con parámetros y operadores
  + Orden de los argumentos
  + Recursividad
  + Anotaciones en funciones
  + docstrings
  + Retorno Empaquetados y desempaquetados.
  + Ámbitos de los distintos objetos.

**Programación funcional**

• Modularidad

• Reutilización de código:

• Encapsulación

• Parámetros

• Retorno de valores

• Ámbito (scope) de las variables

• Funciones como objetos de primera clase

El orden de los argumentos posicionales es importante.

Validación de datos de entrega

Argumentos posicionales y por omisión (palabras clave):

Parámetros arbitrarios en Python \* y \*\*

Funciones con parámetros y operadores

Desempaquetar argumentos con \*:

Separar argumentos posicionales y de palabras clave con /:

**Orden de los argumentos**

1. posicionales
2. \* arbitrarios
3. \*\* kwords -clave valor
4. por omisión

**Recursividad**

**Anotaciones en funciones**

**docstrings**

**Retorno**

* Retorno único:
* Empaquetados y desempaquetados.

**Funciones lambda**

* argumentos
* expresión

**Funciones de orden superior.**

Las funciones de orden superior en Python son aquellas funciones que pueden aceptar otras funciones como argumentos de entrada, manipularlas de manera flexible dentro de otras funciones y al final devolver funciones como resultado.

**map**

**Filter**

**Reduce**

Funciones lambda + funciones de orden superior.

* + lambda + map
  + lambda + filter
  + lambda + reduce (import functools)

**Lazy evaluation.**

**Ámbitos o Scope**

* Ámbito Local (Local Scope):
* Ámbito Encerrado (Enclosing Scope)
* Ámbito Global (Global Scope)
* Ámbito de Nombre Integrado (Built-in Scope)

**Manejo de excepciones**

* + - El bloque try se utiliza para encapsular el código que puede generar una excepción,
    - El bloque except se utiliza para manejar las excepciones capturadas.
    - El bloque else se utiliza tras un evento que no ingresa por excepción.
    - El bloque finally se utiliza como último bloque. Esto es muy importante en una función con return

y

* + - raiser es una declaración para generar manualmente no un error, sino una “excepción” programada, lanzar explícitamente en un lugar o tiempo específico del código.

**Manejo de archivos y directorios:**

directorios mediante cualquier sistema operativo (Windows, Linux o MacOS).

import os

* + - os.path.isfile()
    - os.path.isdir()
    - os.mkdir
    - os.rename
    - os.rmdir
    - os.listdir()
    - os.chdir ()
    - os.remove ()
    - os.getcwd()
    - os.chdir (f"{directorio}\\{archivo}")

(f"{directorio}/{archivo}")

**Manejo de información en archivos externos**

Texto plano txt

Excel xls, xlsx o xlsm

Comma Separated Values csv

JavaScript Object Notation. json

Pickle pkl

eXtensible Markup Language xml

* + Los textos planos usan la t de text o la omiten, los binarios deben usarse con la b de binary
  + Se pueden escribir (write) , leer (read), agregar al final (append) y x para crearlo sin contenido
  + decodificar ASCII, CP1251 / 52 o codificación a UTF-8 o Unicode en Python,
* Escritura
* Lectura
* Lee una línea del archivo
* Lee la totalidad de líneas del archivo y las guarda cada linea como objeto de una lista
* Serialización:
* Deserialización:

**Requests**

El Módulo requests realiza solicitudes HTTP

El protocolo HTTP se basa en dos conceptos principales:

petición (request) y respuesta (response).

REST, REpresentational State Transfer,// RESTful API.

* + GET
  + POST
  + PATCH o PUT
  + DELETE

Trabajos:

Clima [https://openweathermap.org](https://openweathermap.org/)

Banco Central RA <http://estadisticasbcra.com/api/documentacion>

* + apikey
  + token

**Manejo fechas y horas**

ISO (Organización Internacional de Normalización) 8601 sería "2030-01-01T01:30:45.123456

* Orden lógico: "año-mes-día"
* Separadores: (-) (:)
* Precisión: niveles de precisión
* Consistencia internacional
* El método datetime() obtendrás solo la fecha
* El método date() obtendrás solo la fecha
* El método time() obtendrás solo el horario.

**.year,.month,.day,.hour..minute,.second:**

Tabla con los códigos de formato que puede utilizar.

**.datetime()**

**.now()**

**.timedelta()**

**.strftime()**

**.strptime()**

**.timedelta()**

**Time**

* mintime
* maxtime
* .hour
* .minute
* second
* .microsecond
* .isoformat()

**Date:**

* mindate
* maxdate
* .year
* .month
* .day
* .today()
* .weekday()
* .isoweekday()

**Interfaz Gráfica**

**Graphical User Interface– GUI -tkinter():**

Tkinter es una biblioteca estándar de Python que proporciona una interfaz de programación para crear aplicaciones de escritorio multiplataforma con una interfaz gráfica. Tk es una biblioteca gráfica + interface escritas en Tcl (Tool Command Language).

* Tkinter es nativa en la instalación estándar, no se requiere una instalación adicional.
* Es una biblioteca multiplataforma, lo que significa que las aplicaciones creadas con Tkinter pueden ejecutarse en diferentes sistemas operativos, como Windows, macOS y Linux.
* Proporciona una amplia gama de widgets y herramientas para la creación de interfaces gráficas, como botones, etiquetas, campos de entrada, menús desplegables, etc.
* Permite la personalización de la apariencia y el comportamiento de los widgets mediante propiedades y opciones.
* Es altamente integrable con otras bibliotecas y módulos de Python.
  + - windows o root (Ventana principal)
    - Frame (Marco)
    - Label (Etiqueta)
    - Entry (Campo de entrada)
    - Button (Botón)
    - Text (Texto)
    - Listbox (Lista)
    - Checkbutton (Casilla de verificación)
    - Radiobutton (Botón de opción)
    - Menu (Menú)
* Cada widget tiene sus métodos y atributos para personalizar su apariencia y comportamiento en la interfaz de usuario.
* Alguno de ellos como bg – fondo son utilizados en varios widgets diferentes
  + - x
    - y
    - .geometry("WxH")
    - .geometry("WxH+X+Y")
    - height
    - width
    - font
    - bg
    - fg
    - bd
    - cursor
    - relief
    - state
    - padding
    - command
    - justify

Métodos principales para ubicar y organizar widgets.

* pack()
* place()
* grid().