Introduccion a Python
Cheat Sheet

(la maga de Python.)

Variables y Cadenas de texto

Utilizamos variables para almacenar valores. Un string es una cadena de texto, rodeada de comillas dobles o simples.

Hola mundo, nuestra primera línea de código

```
print("Hola mundo")
```

Hola mundo dentro de una variable

```
msj = "Hola mundo"
print(msj)
```

Concatenación (combinando strings/cadenas de texto)

```
primer_nombre = "Mary"
apellido = "Jackson"
nombre_completo = primer_nombre + ' ' + apellido
print(nombre_completo)
```

Listas

Conjunto ordenado y mutable de elementos a los que accedemos mediante su index o a través de un bucle.

Creamos una lista

```
colores = [ 'verde', 'azul', 'rojo']
```

Obtenemos el primer item de la lista

```
primer_color = colores[0]
```

Obtenemos el ultimo item de la lista

```
ultimo_color = colores[-1]
```

Recorremos la lista a través de un bucle

```
for color in colores:
      print(color)
```

Añadimos items a la lista

```
colores = []
colores.append('amarillo')
colores.append('celeste')
colores.append('naranja')
```

Creando una lista numérica

```
numero_cuadrado = []
for x in range(1, 11):
    numero_cuadrado.append(x**2)
```

Comprensión de listas (List Comprehensions)

```
numeros_cuadrados = [x**2 for x in range(1, 11)]
```

Listas (continuación)

Obteniendo una porción de una lista (Slicing)

```
mascotas = ['gato', 'perro', 'tortuga', 'hamster']
segunda_mascota = mascotas[ 1:2 ]
```

Creando la copia de una lista

```
copia_de_mascotas = mascotas[ : ]
```

Tuplas

Las tuplas son similares a las listas, pero a diferencia de las listas, los items de las tuplas no pueden modificarse.

Creamos una tupla

```
info = ('Ana', '34', 'programadora')
```

Accedemos a sus indices

```
print(info[1])
```

Diccionarios

Conjunto no ordenado y mutable de elementos clave-valor a los cuales accedemos mediante sus claves.

Creamos un diccionario

```
curso = {'modalidad': 'online', 'exámenes': '10'}
```

Accediendo a sus valores

```
print("Este curso tendra  " + curso['exámenes'])
```

Añadiendo un nuevo par de clave-valor

```
curso['duracion_horas'] = 200
```

Recorriendo el diccionario a traves de sus claves-valores

```
edades = { 'Melina : 24 , 'Luis' : 26 }
for nombre, edad in edades.items():
   print('La edad de ' + nombre +'es '+ str(edad))
```

Recorriendo el diccionario a traves de sus claves

```
edades = { 'Melina : 24 , 'Luis' : 26 }
for nombre in edades.keys():
  print('La edad de ' + nombre + 'fue registrada')
```

Recorriendo el diccionario a traves de sus valores

```
edades = { 'Melina : 24 , 'Luis' : 26 }
for edad in edades.items():
  print('Hay estudiantes con ' +str(edad)+ 'años')
```

Sentencia If

Esta estructura de control se usa para ejecutar bloques de código si y solo si, se cumple una determinada condición.

Operadores condicionales

```
igual a                 x == 42
distinto de             x != 42
mayor que               x  > 42
mayor o igual que       x >= 42
menor que               x  < 42
menor o igual que       x <= 42
```

Operadores condicionales con listas

```
''gato' in mascotas
'huron' not in mascotas
```

Asignando valores booleanos

```
juego_activo = True
permitido_editar = False
```

Condición con la sentencia If

```
if edad >=  18:
      print("Es mayor de edad")
```

Sentencias if- elif - else

```
if edad < 4:
      precio_pasaje = 50
elif edad < 18:
      precio_pasaje = 80
else:
      precio_pasaje = 100
```

Input de usuario

Python permite solicitar información al usuario a través de un input. Todos los ingresos por input son almacenados como cadenas de texto (strings).

Solicitando el ingreso de un valor

```
nombre = input("Ingrese nombre: ")
apellido = input("Ingrese apellido: ")
print("Hola " + nombre + apellido + "!")
```

Solicitando el ingreso de un valor numérico

```
ingreso_edad = input("Ingrese su edad")
edad = int (ingrese_edad)

ingreso_pi = input("¿Cual es el valor de pi? ")
pi = float (ingreso_pi)
```

Bucle While

El bucle while ejecuta un fragmento de código mientras cierta condición permanece verdadera.

Ejemplo de bucle while

```
valor_actual = 1
while valor_actual <= 5:
        print(valor_actual)
        valor_actual += 1
```

Permitiendo al usuario elegir cuando finalizar un bucle

```
finalizar = ' '
while finalizar  != 'no' :
       finalizar = input("¿Salir del bucle? " )
       print(finalizar)
```

Funciones

Las funciones son bloques de código con un nombre, designadas para una tarea especifica, que pueden ser ejecutados como una unidad funcional. Pueden recibir argumentos opcionalmente, denominados parámetros.

Creamos una función

```
def saludar():
   """ Mostrar un saludo"""
      print("¡Hola!")

saludar()
```

Función con parámetros

```
def saludar(nombre):
   """Mostrando un saludo personalizado"""
       print("¡Hola " + nombre + "!")

saludar('Malena')
```

Parámetros con valores por defecto

```
def preparar_menu(postre = 'brownie con helado'):
       """Mostrar el postre del menu"""
       print("El postre de hoy es " + postre )

preparar_menu()
preparar_menu('flan con crema')
preparar_menu('lemon pie')
```

Retornando un valor

```
def sumar_valores(a, b):
 """Sumando numeros y retornando el resultado"""
      return a + b

sum = sumar_valores(5, 4)
print(sum)
sum = sumar_valores(8, 12)
print(sum)
```

Clases

Una clase define el comportamiento y tipo de información que un objeto puede contener. La información de una clase se almacena en sus atributos y el comportamiento se almacena en sus métodos, funciones propias de la clase.

Creamos la clase Perro

```
class Perro() :
        def __init__(self, nombre):
    """Inicializando el objeto estudiante."""
               self.nombre = nombre

        def alimentar(self):
    """Imprime un saludo en pantalla."""
       print(self.nombre + "ha sido alimentado")

primer_perro = Perro('Huesos')

"""Utilizamos su atributo nombre"""
print(primer_perro.name + 'recibió las vacunas.')

"""Utilizamos su método alimentar"""
primer_perro.alimentar()
```

Herencia

```
class Galgo(Perro):
 """Clase que hereda de la clase Perro
       sus atributos y métodos"""

   def __init__(self, nombre):
  """Inicializa la clase Galgo"""
       super().__init__(nombre)

   def correr(self):
      print(self.name +
          "corre a una velocidad de 63k/h")

perro_galgo = Galgo("Toby")

print(perro_galgo.nombre + " es un perro galgo.")
perro_galgo.alimentar()
perro_galgo.correr()
```

Polimorfismo

```
 """Objetos de diferentes clases pueden tener
un comportamiento o atributo del mismo nombre pero
con diferente valor."""

class Coche():

    ruedas=4

   def desplazamiento(self):
      print("El coche se esta desplazando
            sobre 4 ruedas")

class Moto():

    ruedas=2

   def desplazamiento(self):
       print("La moto se esta desplazando
            sobre 2 ruedas")
```

Trabajando con archivos

A través de nuestros programas podemos leer y escribir información en archivos.
Los archivos son abiertos en modo lectura ('r') por defecto, pero también podemos abrirlos en modo escritura ('w') o ('a') para agregar nuevos elementos a una lista.

Leyendo un archivo y almacenando líneas

```
nombre_archivo = 'articulo.txt'
with open(nombre_archivo) as archivo_objeto:
        lines = archivo_objeto.readlines()

for line in lines:
       print(line)
```

Escribiendo dentro de un archivo

```
nombre_archivo = 'registro.txt'
with open(nombre_archivo, 'w') as archivo_objeto:
    archivo_objeto.write("Registros actualizados")
```

Añadiendo líneas a un archivo

```
nombre_archivo = 'registro.txt'
with open(nombre_archivo, 'a') as archivo_objeto:
   archivo_objeto.write("\nRegistros corregidos")
```

Siempre que abrimos un archivo debemos cerrarlo

```
nombre_archivo = open("mi_archivo.txt", 'r')
nombre_archivo.close()
```

Excepciones

Las excepciones nos ayudan a responder apropiadamente a errores que pueden ocurrir.
En el bloque try debemos colocar código que podría llegar a suceder.
El código que debería ejecutarse en
respuesta a un error se coloca en el bloque except.
El código que debe ejecutarse solo si el
bloque try fue exitoso se coloca en el bloque else.

Utilizando una excepción

```
mensaje = "Cuantos tickets necesita?"
numero_tickets = input(mensaje)

try:
   numero_tickets = int(numero_tickets)

except valorErroneo:
   print("Ingrese nuevamente numero de tickets.")

else:
    print("Tus tickets se estan imprimiendo.")
```