**CONVENCIONES DE CODIGO DE: PYTHON**

|  |  |
| --- | --- |
| **Nombre:** | Espacios en Blanco |
| **Descripción:** | El uso de espacios en blanco puede resultar clave para mejorar la legibilidad de nuestro código. Se trata de buscar un punto de equilibrio entre un código demasiado disperso y con gran cantidad de espacios, y un código demasiado junto donde no se identifican sus partes. |
| **Ejemplo correcto:** | x = 5 |
| **Ejemplo incorrecto:** | x=5 |

|  |  |
| --- | --- |
| **Nombre:** | Nombres de variables |
| **Descripción:** | Las palabras se escriben en minúscula separadas por guiones bajos. Con el tipo de formato snake\_case, estas también deben de ser lo más explicitas posibles para que los miembros del equipo puedan saber cuál es el objetivo o funciones para las que se usa dicha variable. |
| **Ejemplo correcto:** | definicion\_variable = “” |
| **Ejemplo incorrecto:** | variableX=”” |

|  |  |
| --- | --- |
| **Nombre:** | Sangría |
| **Descripción:** | Se utiliza 4 espacios para la sangría, no tabulaciones. |
| **Ejemplo correcto:** | def funcion(param):  if param > 0:  print("El parámetro es positivo")  else:  print("El parámetro es negativo") |
| **Ejemplo incorrecto:** | def funcion(param):  if param > 0:  print("El parámetro es positivo")  else:  print("El parámetro es negativo") |

|  |  |
| --- | --- |
| **Nombre:** | Importaciones |
| **Descripción:** | Se utilizan importaciones relativas siempre que sea posible. |
| **Ejemplo correcto:** | from .math import sqrt |
| **Ejemplo incorrecto:** | from math import sqrt |

|  |  |
| --- | --- |
| **Nombre:** | Documentación |
| **Descripción:** | Se utilizan docstrings para documentar las funciones, clases y módulos. |
| **Ejemplo correcto:** | def funcion(param):  """Esta función hace algo.  Args:  param: El parámetro de la función.  Returns:  El valor de retorno de la función.  """  return param |
| **Ejemplo incorrecto:** | def funcion(param):  """Esta función hace algo"""  return param |

|  |  |
| --- | --- |
| **Nombre:** | Longitud de linea |
| **Descripción:** | Las líneas de código no deben superar los 79 caracteres. |
| **Ejemplo correcto:** | def funcion\_correcta(param1, param2, param3, param4, param5, param6, param7):  return (  param1  + param2  + param3  + param4  + param5  + param6  + param7  ) |
| **Ejemplo incorrecto:** | def funcion\_muy\_larga(param1, param2, param3, param4, param5, param6, param7):  return param1 + param2 + param3 + param4 + param5 + param6 + param7 |

|  |  |
| --- | --- |
| **Nombre:** | nombres de métodos: |
| **Descripción:** | Usa verbos para los nombres de métodos y sigue el formato snake\_case |
| **Ejemplo correcto:** | def calcular\_area() |
| **Ejemplo incorrecto:** | def AreaCalculada() |

|  |  |
| --- | --- |
| **Nombre:** | nombres de clases: |
| **Descripción:** | Usa sustantivos para los nombres de clases y sigue el formato CamelCase. |
| **Ejemplo correcto:** | class MiClase |
| **Ejemplo incorrecto:** | class mi\_clase |

|  |  |
| --- | --- |
| **Nombre:** | nombres de constantes: |
| **Descripción:** | Usa nombres en mayúsculas separados por guiones bajos para las constantes. |
| **Ejemplo correcto:** | PI = 3.14159 |
| **Ejemplo incorrecto:** | pi = 3.14159 |

|  |  |
| --- | --- |
| **Nombre:** | Evitar comentarios redundantes |
| **Descripción:** | Usa comentarios para explicar el por qué del código, no lo que el código hace. |
| **Ejemplo correcto:** | # Calcula el área del círculo |
| **Ejemplo incorrecto:** | # Esto calcula el área multiplicando pi por el radio |

**CONVENCIONES DE CODIGO DE: JAVA**

|  |  |
| --- | --- |
| **Nombre:** | Variables |
| **Descripción:** | Los nombres de variables deberían ser cortos y llenos de significado. La elección de una variable debería ser mnemónica-es decir, diseñada para indicar al observador casual su utilización. Se deben evitar los nombres de variable de un sólo carácter, excepto para variables temporales. Algunos nombres comunes de este tipo de variables son: i, j, k, m, y n para enteros. |
| **Ejemplo correcto:** | int i = 5;  String nombreAlumno; |
| **Ejemplo incorrecto:** | String a = “”;  int variable1=0; |

|  |  |
| --- | --- |
| **Nombre:** | Nombres de variables, métodos y clases |
| **Descripción:** | Las palabras se escriben en minúscula, con la primera letra de cada palabra en mayúscula, excepto la primera palabra que se escribe en minúscula |
| **Ejemplo correcto:** | int variableCorrecta;  public void metodoCorrecto() {  // ...  }  class ClaseCorrecta {  // ...  } |
| **Ejemplo incorrecto:** | int variableIncorrecta;  public void metodoIncorrecto() {  // ...  }  class ClaseIncorrecta {  // ...  } |

|  |  |
| --- | --- |
| **Nombre:** | Sangria |
| **Descripción:** | Se utiliza una tabulación para la sangría |
| **Ejemplo correcto:** | public class Clase {  public void metodo() {  if (condicion) {  System.out.println("Hola mundo");  } else {  System.out.println("Adiós mundo");  }  }  } |
| **Ejemplo incorrecto:** | public class Clase {  public void metodo() {  if (condicion) {  System.out.println("Hola mundo");  } else {  System.out.println("Adiós mundo");  }  }  } |

|  |  |
| --- | --- |
| **Nombre:** | Metodos |
| **Descripción:** | Los métodos, la primer letra debe ser minúscula, utiliza nomenclatura camelCase, los nombres deben conformarse por el par verbo + sustantivo , el nombre va después del tipo de método (void, int, double, String), al finalizar el nombre del método debe indicarse mediante paréntesis con o sin argumentos () |
| **Ejemplo correcto:** | import java.util.Scanner;  import java.util.Date;  public class Clase {  public static void main(String[] args) {  int mi\_metodo(){  }  }  } |
| **Ejemplo incorrecto:** | import java.util.Scanner;  import java.util.Date;  public class Clase {  public static void main(String[] args) {  int miMetodo(int a, int b){  //Info…  return 1;  }  }  } |

|  |  |
| --- | --- |
| **Nombre:** | Constantes |
| **Descripción:** | Las constantes, todas las letras de cada palabra deben estar en mayúsculas, se separa cada palabra con un \_, se declaran similar a las variables, con la diferencia de que el tipo de dato va después de la palabra reservada final. |
| **Ejemplo correcto:** | final int EDAD |
| **Ejemplo incorrecto:** | final int edad |

|  |  |
| --- | --- |
| **Nombre:** | Los paquetes |
| **Descripción:** | Los paquetes., deben ser escritos todo en minúscula, van después de la palabra reservada package, si se van a usar paquetes dentro de otros paquetes, se unen mediante un punto (.), finalizan con ; |
| **Ejemplo correcto:** | package ventanas;  package imagenes.iconos; |
| **Ejemplo incorrecto:** | package PaquetesJava; |

|  |  |
| --- | --- |
| **Nombre:** | Clases e interfaces |
| **Descripción:** | Las clases, la primer letra debe ser mayúscula, utiliza nomenclatura camelCase, para las clases, los nombres deben de ser sustantivos (Sujeto) y van después de la palabra reservada class, Para las interfaces, los nombres deben de ser adjetivos (Califica el sustantivo) y van después de la palabra reservada interface |
| **Ejemplo correcto:** | class VentanaRegistro interface ActionListener |
| **Ejemplo incorrecto:** | class ventanaRegistro interface actionListener |

|  |  |
| --- | --- |
| **Nombre:** | Métodos |
| **Descripción:** | Los métodos deben seguir la convención de nomenclatura camelCase, comenzando con una letra minúscula y utilizando un verbo seguido de un sustantivo. El nombre del método debe ir después del tipo de retorno (void, int, double, String) y debe finalizar con paréntesis con o sin argumentos. |
| **Ejemplo correcto:** | import java.util.Scanner;  import java.util.Date;  public class Clase {  public static void main(String[] args) {  int mi\_metodo(){  }  }  } |
| **Ejemplo incorrecto:** | import java.util.Scanner;  import java.util.Date;  public class Clase {  public static void main(String[] args) {  int miMetodo(int a, int b){  // Información...  return 1;  }  }  } |

|  |  |
| --- | --- |
| **Nombre:** | Clases e Interfaces |
| **Descripción:** | Para las clases, la primera letra debe ser mayúscula y seguir la convención de nomenclatura camelCase. Los nombres deben ser sustantivos y seguir la palabra reservada "class". Para las interfaces, los nombres deben ser adjetivos que califiquen el sustantivo y seguir la palabra reservada "interface". |
| **Ejemplo correcto:** | class VentanaRegistro  interface ActionListener |
| **Ejemplo incorrecto:** | class ventanaRegistro  interface actionListener |