
SKILLING
CENTER
TECMILENIO

SKILLING CENTER TECMILENIO

Visualización y Programación en Python

Ciclos en Python

Introducción

En la empresa Paso del Norte se tiene la necesidad de determinar la frecuencia de ocurrencia de defectos de calidad de un producto determinado, en un periodo de tiempo, por ejemplo, en un turno laboral. El departamento de calidad estima que tú podrás resolver el problema, para lo cual recurres al lenguaje Python para diseñar el algoritmo apropiado para este caso y ponerlo en el piso de producción para comunicarlo a los interesados.

Aceptas el reto e inmediatamente investigas las opciones que tienes en Python, conocidos como ciclos, para poder seleccionar aquella que mejor se adapta a las circunstancias e implementarla con éxito.

Explicación

Es muy común que en los procesos analíticos se requiera ejecutar un bloque de código en varias ocasiones, por ejemplo, para acumular la sumatoria de los valores enteros menores de 10, en código Python se puede hacer de la siguiente manera:

```
i = 1  #declara la variable 'i' y le asigna un valor
while i < 10:
#Se ejecuta si i es menor a 10
    print(i)
    i = i + 1 #contador
```

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

El proceso inicia con el valor de la variable i en 1. La función “while” establece que mientras que la variable i sea menor de 10, imprime el valor y se incrementa en una unidad, y se termina el ciclo cuando ya no se cumpla la condición.

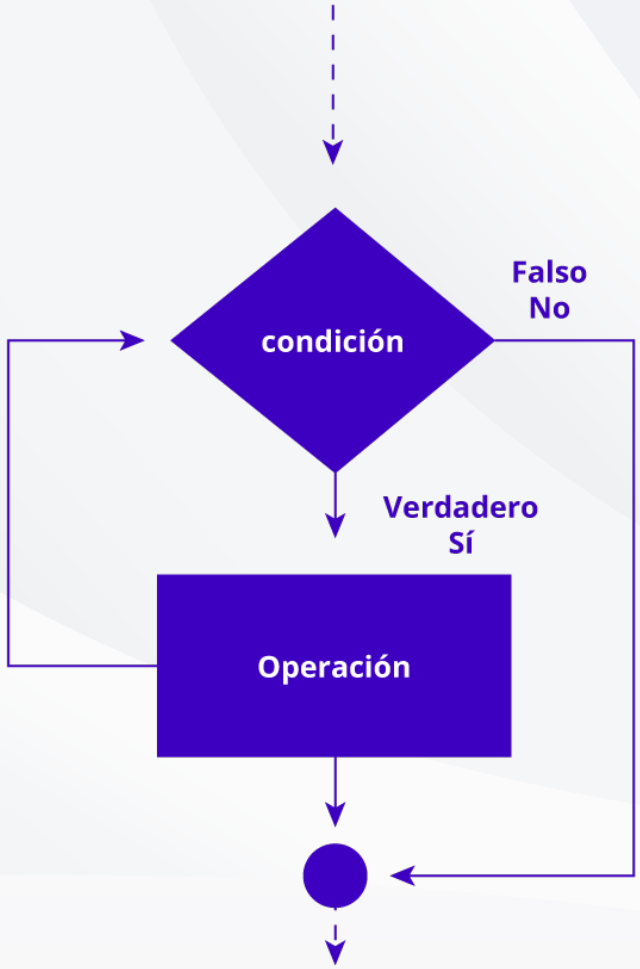
Explicación

Python tiene varias estructuras de control que generan rutas de ejecución, conocidas como ciclos o bucles que permiten ejecutar el mismo código repetidamente mientras se cumpla cierta condición. A continuación se exponen los ciclos “while”, “for” y los ciclos anidados. Lozano (s.f.) define a un iterable como un objeto sobre el que se puede iterar, esto es, que permite recorrer sus elementos uno a uno.

Explicación

- El ciclo “while”

La condición se establece al inicio del proceso y se ejecuta repetidamente mientras se cumple.

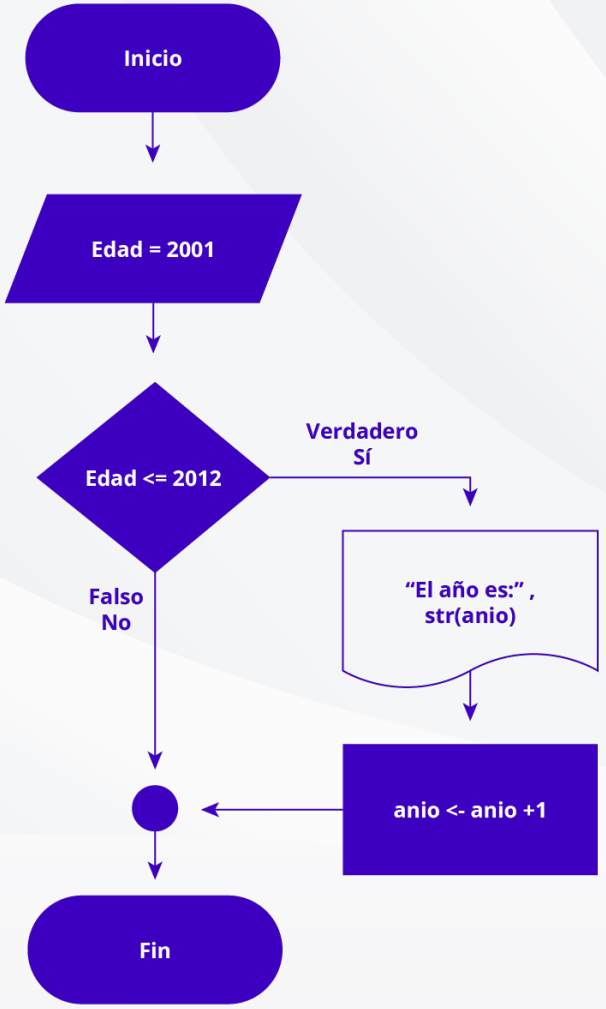


Explicación

Como ejemplo del ciclo while, imprime los años 2001 al 2012.
Se inicia el proceso con la variable anio = 2001, se evalúa la condición de que la variable sea menor de 2012, se imprime el año y se incrementa el valor en 1 unidad.

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-----------------------------------------------------")
print("INCREMENTAR EL AÑO HASTA 2012.")
print("-----------------------------------------------------")
#Entradas
anio= 2001
#Proceso
while anio <=2012:
#Salida
    print("El año es:", str(anio))
    anio += 1
```

```
-----------------------------------------------------
INCREMENTAR EL AÑO HASTA 2012.
-----------------------------------------------------
El año es: 2001
El año es: 2002
El año es: 2003
El año es: 2004
El año es: 2005
El año es: 2006
El año es: 2007
El año es: 2008
El año es: 2009
El año es: 2010
El año es: 2011
El año es: 2012
```



Estas pantallas se obtuvieron directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

- El ciclo “for”

El ciclo *for* es similar que el *while*, sin embargo, es adecuado para problemas de clasificación de valores de variables para luego asignarle operaciones o simplemente acciones de agrupación. Heinerman (2021) menciona que el ciclo *for* permite agrupar varios objetos en un bloque continuo de memoria, si bien es cierto que se puede leer con un bucle *while*, es más adecuado recorrerlo con un bucle *for*.

Por ejemplo, dado el conjunto de precios de venta semanal [155, 171, 148, 161, 158, 153, 162], imprime el “Precio actual” y el “Precio Promedio”

```
precio_semanal = [155, 171, 148, 161, 158, 153, 162]
precio_sum = 0

for precio in precio_semanal:
    precio_sum = precio_sum + precio
    print("Precio actual:",precio)

promedio = precio_sum // len(precio_semanal)
print("Precio Promedio:", promedio)
```

```
Precio actual: 155
Precio actual: 171
Precio actual: 148
Precio actual: 161
Precio actual: 158
Precio actual: 153
Precio actual: 162
Precio Promedio: 158
```

Estas pantallas se obtuvieron directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Es posible implementar el ciclo *for* con una secuencia, utilizando la clase “range”, el cual devuelve una iteración que va desde una posición inicial hasta la posición final. Hay que recordar que Python considera la posición inicial como la número 0.

Por ejemplo, imprime los valores menores al 11.

```
for i in range (11)
    print(i)
```

La salida del algoritmo es una lista del 0 al 10, en incrementos de 1.

Explicación

- Ciclos anidados

Sintes (2023) explica que un bloque está anidado cuando un ciclo (interior) está en el cuerpo de instrucciones de otro ciclo (exterior).

Es útil para resolver problemas más complejos, como iterar algún objeto donde cada elemento tiene otra clase iterable, como una lista de listas.

Para evitar complicaciones en la lógica del código, se recomienda integrar un máximo de tres ciclos.

Se pueden tener ciclos anidados tanto con el ciclo “for” como con el ciclo “while” y combinados, según sea el caso.

Explicación

Ejemplo:

Imprime los números primos comenzando con el 2 y terminado con el 29. Además, imprime el valor correspondiente elevado al cubo.

```
# -*- coding: utf-8 -*-
#Decoración: primos_y_cubos
print("-----------------------------------------------------")
print("CUBO DE UN NÚMERO PRIMO.")
print("-----------------------------------------------------")
#Inicializar
n2 = 2
#Proceso
for i in range(1, 29):
    c = 0
    for n1 in range(2, n2//2):
        if n2 % n1 == 0 :
            c += 1
            n1 = n2
    if c == 0 :
        print("El cubo de", n2," es: ", n2**3)
    n2 = n2 + 1
```

```
-----------------------------------------------------
CUBO DE UN NÚMERO PRIMO.
-----------------------------------------------------
El cubo de 2  es:  8
El cubo de 3  es:  27
El cubo de 4  es:  64
El cubo de 5  es:  125
El cubo de 7  es:  343
El cubo de 11  es:  1331
El cubo de 13  es:  2197
El cubo de 17  es:  4913
El cubo de 19  es:  6859
El cubo de 23  es:  12167
El cubo de 29  es:  24389
```

Estas pantallas se obtuvieron directamente del software que se está explicando en la computadora, para fines educativos.

Actividad

Para incrementar tu aprendizaje del tema, contesta las siguientes preguntas:

1. Imagina que estás en la posición de inspector de recibo de compras de un producto y quieres utilizar ciclos de Python para determinar la variedad y cantidad de productos recibidos de cada uno. Considera el recibo de un proveedor que te envió un pedido con 5 tipos de productos (p1, p2, ...p5) y en total con 20 artículos, los cuales pueden ser repetidos o diferentes.
2. Genera aleatoriamente una lista del recibo con el tipo y la cantidad en unidades recibidas y desarrolla el algoritmo en Python para determinar la cantidad total de artículos recibidos de cada tipo.
3. Genera aleatoriamente una lista con 20 números. Para cada uno, imprime el número, calcula el cuadrado y la suma del número más el siguiente en la lista. Desarrolla el algoritmo en Python e imprime los resultados. Resuelve el problema utilizando cada uno de los ciclos “for” y “while”.

Cierre

Con el aprendizaje de este tema tienes las herramientas necesarias para facilitar la tarea de clasificar los defectos de calidad, analizar su frecuencia y comunicar los resultados. Incluso agregas gráficas a tu análisis, como histogramas en donde se puedan identificar fácilmente los defectos con mayor ocurrencia, además de proponer posibles acciones que puedan reducirlos.

Como reconocimiento a tu labor, el jefe del departamento te ha pedido que este algoritmo se replique en todas las líneas de producción.

Bibliografía

- Heineman, G. (2021). *Learning Algorithms: A Programmer's Guide to Writing Better Code.* Estados Unidos: O'Reilly.
- Lozano, J. (s.f.). *Tutorial de Python.* Recuperado de https://j2logo.com/python/tutorial/
- Sintes, B. (2023). *Introducción a la programación con Python.* Recuperado de https://www.mclibre.org/consultar/python/

SKILLING CENTER TECMILENIO

Visualización y Programación en Python

Archivos, funciones, módulos y paquetes

Introducción

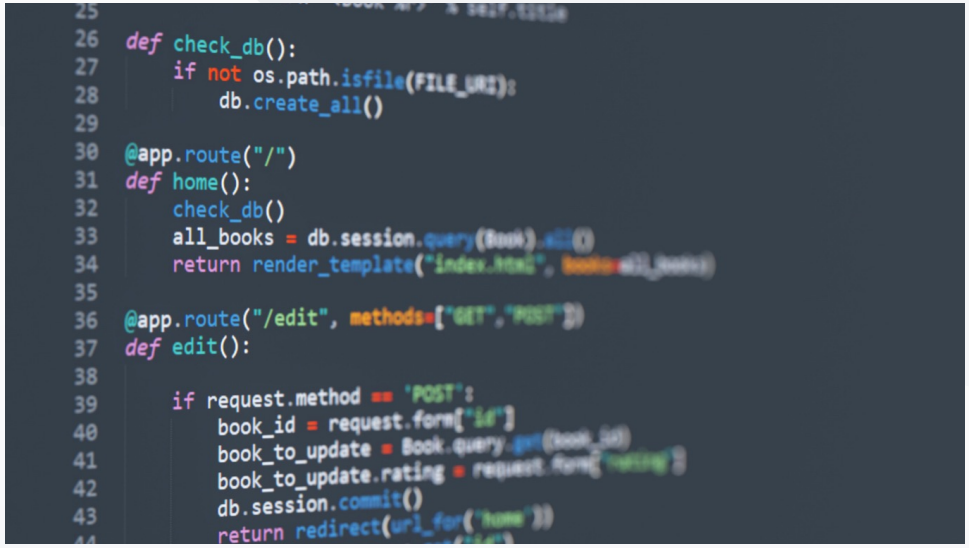
Conforme vas conociendo las cualidades del lenguaje Python, vas descubriendo los beneficios que tiene para resolver problemas y aplicarlos a tu área de interés. En este tema, vas a aprender la manera en que puedes establecer algoritmos que son utilizados repetidamente y, por lo tanto, puedes llamar a estos algoritmos cada vez que lo necesites, sin tener que escribirlos nuevamente.

Adicionalmente, aprenderás a utilizar uno de los beneficios más poderosos de Python, que son las librerías de código que ya están establecidas y listas para que las uses en tus aplicaciones, lo cual simplifica ampliamente el desarrollo de cualquier modelo analítico.

Explicación

Funciones en Python

- Permiten definir un bloque de código para realizar una tarea específica.
- Se definen utilizando la palabra reservada “def”.
- Debe tener un nombre. Se recomienda usar minúsculas y separar palabras con guion bajo.
- La sintaxis general es la siguiente:
  - **def nombre(parámetro1, parámetro 2…):**
    <cuerpo_funcion:>
    Return <valor de salida>



Explicación

Como ejemplo, considera la función básica que ejecuta una instrucción.

Para esta función, el código Python, no tiene parámetros y se define de la siguiente forma:

- **def di_hola():**
- **print(“Hola”)**

Y para ejecutarla se le llama di_hola() y como resultado imprime Hola.

Con la misma función, puedes agregar tu nombre para que imprima Hola + nombre:

- **def di_hola(nombre)**
- **print(“Hola ”, nombre)**

Y se ejecuta con di_hola(“Jose”), y como resultado imprime Hola Jose.

Explicación

Otro ejemplo es cuando se quiere realizar una operación aritmética, en este caso, multiplicar por 2 un número. Se define la función con el siguiente código:

- **def f(x)**
- **return 2*x**

Y para usarla, sólo le das el valor de x, como y = f(3) y el resultado es 6.

Explicación

De acuerdo con el Libro de Python (s.f.), las funciones parten de los siguientes principios:

- **Reusabilidad.** Fragmentos de código para repetirse sin escribirlo nuevamente, y se actualiza con mayor facilidad.
- **Modularidad.** Se desarrolla el código en módulos, facilitando su lógica y legibilidad.

Ejemplo de función para multiplicar un conjunto de números:

```
def multiplica(numeros):
    total = 1
    for n in numeros:
        total = total * n
    return total
```

Para ejecutarla, se llama con el grupo de números multiplica([1, 3, 5, 4]), y la salida es 60.
Con otro grupo de números, por ejemplo, multiplica(10, 5), la salida es 50 y multiplica([1, 20 , 3, 5, 4, 7]), la salida es 8400.

La sentencia *return* te permite hacer dos cosas:

- Salir de la función.
- Regresar los resultados generados en la ejecución de la función.

Explicación

Módulos

Lozano (s.f.) define un espacio de nombres como una colección aislada de nombres que referencian a objetos.

- El uso de módulos genera un archivo que contiene código con funciones, variables o clases.
- Se identifica y guarda con un nombre y su extensión es .py.
- Pueden ser accedidos desde otros módulos, lo cual permite el reúso del código.
- Para poder utilizarlos se utiliza la palabra reservada import.

Por ejemplo, generar el código para mostrar la serie de Fibonacci:

```
def fibo(n): # se limita la serie de fibonacci hasta el número n
    a, b = 0, 1
    while a < n:
        print(a, end=' -  ')
        a, b = b, a+b
    print()
```

Una vez creado el módulo, se guarda como archivo fibonacci.py en la misma carpeta de trabajo.

Para usarlo, se llama de la siguiente manera:

```
Import fibonacci
fibonacci.fibo(1000)
```

Y el resultado es la serie calculada hasta antes del valor de 1000:
0 - 1 - 1 - 2 - 3 - 5 - 8 - 13 - 21 - 34 - 55 - 89 - 144 - 233 - 377 - 610 - 987 -

Explicación

Paquetes

- Similar a los módulos, un paquete es un directorio que contiene un conjunto de módulos.
- Para utilizarlo, también se emplea la palabra reservada *import*.

Por ejemplo, en una aplicación para pedidos en línea, puedes crear como paquete los módulos de usuarios y de pedidos, y utilizarlos de forma individual o conjunta.

Librerías

- Son un conjunto de funcionalidades muy completas.
- Se utilizan para propósitos específicos como configurar y organizar listas, series y tablas, para visualización, realizar cálculos matemáticos y estadísticos, entre otros (Heineman, 2021).
- Hay una gran variedad de librerías, y se pueden utilizar gratuitamente.
- Las librerías más populares son Matplotlib, Pandas, Seaborn, Numpy, Plotly, entre otras.
- También se invocan con la palabra reservada *import* y se abrevia para luego llamarla.

Por ejemplo, el siguiente código:

- **import numpy as np**
- **print(np.arange(0,5))**

Genera una serie de 5 números del 0 al 4.

Explicación

La librería Pandas

- Permite leer archivos con diferentes formatos, los más comunes son de Excel tipo .csv y .xlsx.
- Pandas convierte los datos en un “dataframe” (similar a una hoja de cálculo).
- En primer lugar, se debe importar la librería con el código:
  - **import pandas as pd**
- Y en seguida la lectura del archivo:
  - **pd.read_csv(“nombre del archivo.csv”)**

```
import pandas as pd

pd.read_csv('surveys.csv')
```

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Explicación

Por ejemplo, tienes la información de una encuesta en un archivo de Excel en formato .csv y lo vas a leer en Python:

- Nombre del archivo: surveys.csv
- El código que utilizas es el siguiente:
  - **import pandas as pd**
  - **pd.read_csv('^surveys.csv')**

	record_id	month	day	year	plot_id	species_id	sex	hindfoot_length	weight
0	1	7	16	1977	2	NL	M	32.0	NaN
1	2	7	16	1977	3	NL	M	33.0	NaN
2	3	7	16	1977	2	DM	F	37.0	NaN
3	4	7	16	1977	7	DM	M	36.0	NaN
4	5	7	16	1977	3	DM	M	35.0	NaN
...	...	...	...	...	...	...	...	...	...
35544	35545	12	31	2002	15	AH	NaN	NaN	NaN
35545	35546	12	31	2002	15	AH	NaN	NaN	NaN
35546	35547	12	31	2002	10	RM	F	15.0	14.0
35547	35548	12	31	2002	7	DO	M	36.0	51.0
35548	35549	12	31	2002	5	NaN	NaN	NaN	NaN

35549 rows × 9 columns

Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Este tipo de herramientas permiten manipular datos en Python.
Las librerías constituyen la principal fuente de procesos analíticos.
Con el uso de las librerías puedes realizar análisis muy efectivos de manera sencilla.

Explicación

Manejo de archivos

- Python puede leer o escribir información en un archivo.
- La lectura es para obtener información actualizada y la escritura es cuando nos interesa utilizarla para otros fines.
- Las operaciones más habituales son:
  - Crear un archivo.
  - Escribir sobre un archivo.
  - Leer un archivo.
  - Borrar un archivo.

Para crear un archivo se utiliza la función open con la siguiente sintaxis, con la instrucción de escritura 'w':
- **open(ruta del archivo, 'w')**

Para escribir contenido se utiliza la función "write":
- **f.write('Hola mundo desde Python!')**

Al terminar se cierra el archivo:
- **f.close()**

Para leer e imprimir el archivo se requiere abrir el archivo en modo lectura 'r':
- **f = open('hola.txt', 'r')**
- **print(f.read())**

La salida es "Hola mundo desde Python".

Explicación

Para agregar contenido se utiliza el parámetro “append” con la abreviación ‘a’:

- **f = open(‘hola.txt’, ‘a’)**
- **f.write(‘\n, Hasta pronto!’) # Agrega este texto en la siguiente línea.**

Para leerlo, se vuelve a utilizar el modo ‘r’:

- **f = open(‘hola.txt’, ‘r’)**
- **print(f.read())**
- **f.close()**

Y la salida es la siguiente:
Hola mundo desde Pyhton
Hasta pronto!

Otra manera de hacerlo y no tener que cerrarlo explícitamente es la siguiente sintaxis:

- **with open('hola.txt', 'a') as f:**
- **f.write(‘\nHola de nuevo’)**

Para imprimir, es necesario abrirlo de nuevo en modo lectura,
y la salida es la siguiente:
Hola mundo desde Pyhton
Hasta pronto!
Hola de nuevo

Actividad

Para reforzar tu aprendizaje del tema, contesta las siguientes preguntas:

1. Elabora el código para una función en Python que calcule la raíz cuadrada de un conjunto de números. Debes crear el conjunto de números aleatorios, aplicar la función e imprimir los resultados.
2. Elabora el código para un módulo que calcule la longitud de la hipotenusa de un triángulo rectángulo, dadas las longitudes de los dos catetos adyacentes.
3. Genera una tabla con datos de precios del 2024 con el cierre semanal de una acción de tu preferencia en la bolsa de valores. La tabla debe tener como columnas la fecha, precio de apertura, precio más alto, precio más bajo, precio de cierre y volumen operado. Guarda el documento como un archivo de Excel en formato .csv y utiliza la librería Pandas para leer e imprimir la información. Muestra el código en Python y los resultados.

Cierre

Con este tema te puedes dar cuenta de las inmensas posibilidades que tiene el lenguaje Python para resolver problemas en tu área de trabajo y en toda la organización en donde se toman decisiones basadas en datos. Con esto en mente, te has propuesto estudiar y comprender las aplicaciones de las librerías de Python para utilizarlas en las circunstancias adecuadas en tus modelos analíticos.

Bibliografía

- El Libro de Python. (s.f.). *Módulos en Python.* Recuperado de https://ellibrodepython.com/modulos-python
- Heineman, G. (2021). *Learning Algorithms: A Programmer's Guide to Writing Better Code.* Estados Unidos: O'Reilly.
- Lozano, J. (s.f.). *Espacios de nombres, módulos y paquetes en Python.* Recuperado de https://j2logo.com/python/tutorial/espacios-de-nombres-modulos-y-paquetes/

Tecmilenio no guarda relación alguna con las marcas mencionadas como ejemplo. Las marcas son propiedad de sus titulares conforme a la legislación aplicable, estas se utilizan con fines académicos y didácticos, por lo que no existen fines de lucro, relación publicitaria o de patrocinio.
