

Taller de Programación

Clase 04: Ciclos

Daniela Opitz
dopitz@udd.cl



Basada en presentaciones oficiales de libro Introduction to Programming in Python (Sedgewick, Wayne, Dondero).

Disponible en <https://introcs.cs.princeton.edu/python>

Clase de Hoy

- Pendientes clases anteriores
- Acumuladores y contadores
- Ciclo **for**
- Comparación **while** vs **for**

Acumuladores y Contadores

Dos de las utilidades más comunes en las iteraciones son la acumulación y el conteo de números.

Ejemplo: Sume los primeros **n** números y contar cuántos números hay entre 1 y n (trivial).

inicio variable **suma**
y **contador**
con valor 0

```
#Sumo números desde 1 a 3
#Cuento números desde 1 a 3
n = int(input("Ingrese un numero: "))
suma = 0 #acumulador
contador = 0 #contador

while contador <= n:
    suma = suma + contador
    contador = contador + 1

print(suma)
print(contador-1) #Si no cuento uno más
```

Acumuladores y Contadores

```
#Sumo números desde 1 a 3
#Cuento números desde 1 a 3

n = int(input("Ingrese un numero: "))
suma = 0 #acumulador
contador = 0 #contador

while contador < n:
    suma += contador
    contador += 1

print(suma)
```

```
suma = suma + contador
contador = contador + 1
```

son equivalentes!

Numeros Pares e Impares

- Números pares: números que son divisibles en 2

$$i \% 2 == 0$$

- Números impares: números que no son divisibles en 2

$$i \% 2 \neq 0 \text{ o bien } i \% 2 == 1$$



- **Ejemplo:** Imprimir y contar los número pares entre 1 y un numero n

contador

variable usada para
recorrer la secuencia

```
numero = int(input("Ingrese un numero: "))
i=1
contador = 0
while i <= numero:
    if i % 2 == 0:
        print(i, 'es par')
        contador += 1
    i += 1
print('Numeros de pares entre 1 y 10:', contador)
```

Voy contando los pares

Ciclo **for**

- **for**: Permite repetir un conjunto de instrucciones un numero determinado de veces. La secuencia de instrucciones se recorre en orden.
- Sintaxis:

```
for <variable> in <elemento iterable>:  
    <instrucciones>
```

- Ejemplo: Imprime el texto “Hola número n veces seguido del valor de n donde n va desde 0 a 3”.

Variable usada
para recorrer la
secuencia

```
for i in range(4):  
    print('Hola número', i)
```

Secuencia de
enteros de 0
hasta n - 1

```
$ python3 holas.py  
Hola número 0  
Hola número 1  
Hola número 2  
Hola número 3
```


Ciclo **for**

- Imprime el texto “Hola número n veces seguido del valor de n donde n va desde 4 a 7”.

Inicio

Fin

```
for i in range(4,8):  
    print('Hola número', i)
```

range(start, stop, step)

secuencia de enteros desde start
hasta stop-1, saltándose step
pasos

Salida

```
$ python3 holas2.py  
Hola número 4  
Hola número 5  
Hola número 6  
Hola número 7
```

while vs for

while	for
número desconocido de iteraciones	número conocido de iteraciones
no siempre puede ser sustituido por un ciclo for	puede ser sustituido por un ciclo while
necesita un contador que se inicie antes del loop y que se incremente dentro del loop	usa una variable (contador) para recorrer la secuencia

while vs for

- **Ejemplo:** Imprima todos los números impares menores que n.

Solución 1

```
1 n = int(input('ingrese n: '))
2 if n <= 0:
3     print('Debe ingresar un número mayor a cero')
4 i = 0
5 while i < n:
6     if i % 2 == 1:
7         print(i)
8     i = i+1
```

Solución 2

```
1 n = int(input('ingrese n: '))
2 for i in range(n):
3     if i % 2 == 1:
4         print(i)
```

Solución 3

```
1 n = int(input('ingrese n: '))
2 for i in range(1, n, 2):
3     print(i)
```

while vs for

- Programe un código que permita entregar el resultado de $\sum_{i=1}^n \frac{1}{i}$ donde n es igual a 100

```
1 n = 100 #limite
2 i = 1 #inicio
3
4 suma=0
5 while i <= n:
6     suma=suma + 1/i
7     i=i+1
8 print(suma)
```

```
1 n = 100 #limite
2
3
4 suma=0
5 for i in range(1,n):
6     suma=suma + 1/i
7     i=i+1
8 print(suma)
```

Resumen

Conceptos

- **while**: ejecutar código mientras una condición se cumple
- **for**: ejecutar código al recorrer una secuencia. La secuencia se puede generar con la función `range(...)`

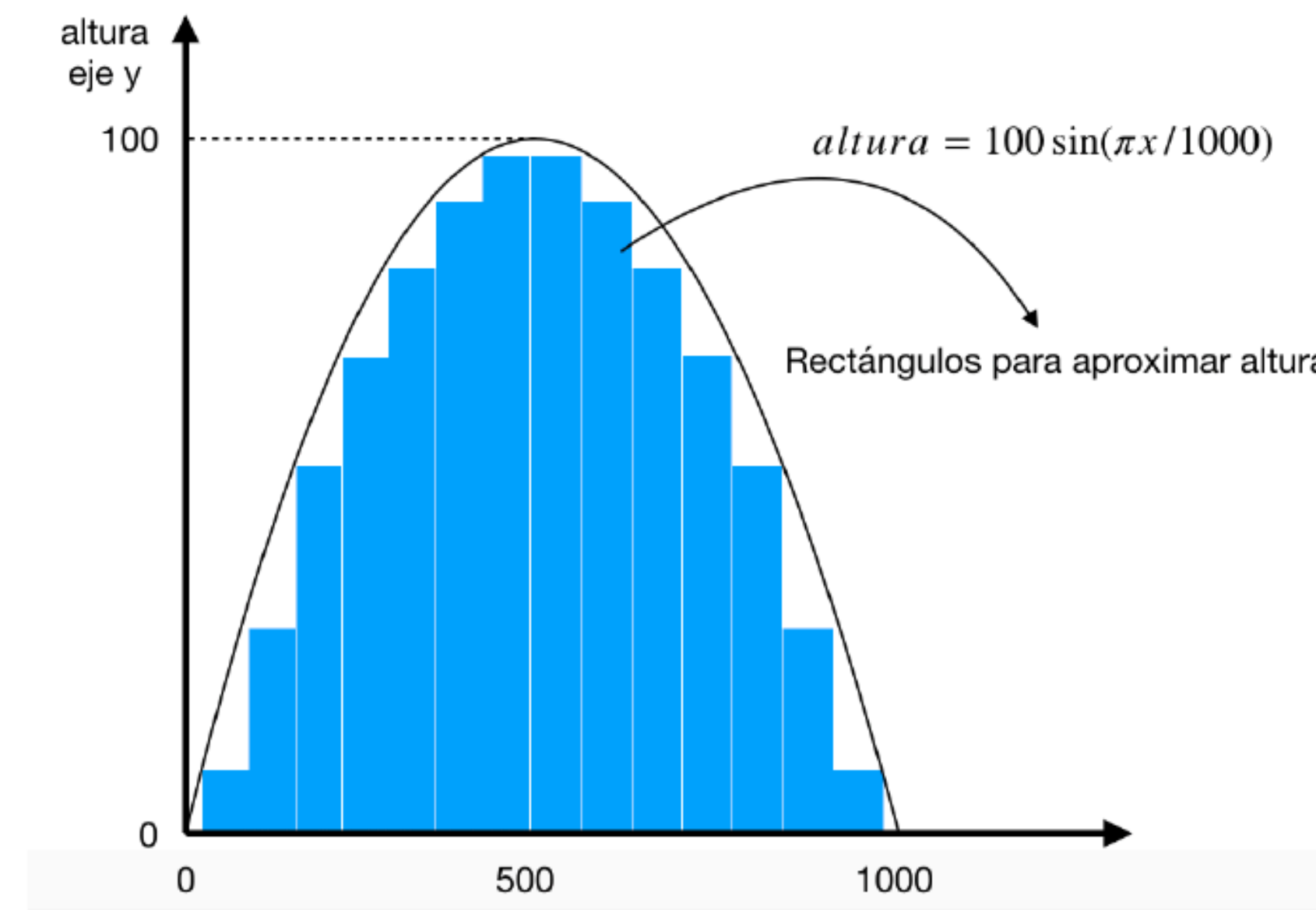
Funciones

- **range(stop)**: secuencias de enteros hasta `stop-1`
- **range(start, stop, step)**: secuencia de enteros desde `start` hasta `stop-1`, saltándose `step` pasos

Actividad 1

Programa un código que calcule el área del perfil de un cerro donde se construirá un túnel que pasa por la cuesta Lo Prado considerando que la forma del cerro se puede aproximar usando la altura h , tal que $h = 100\sin(\pi x/1000)$, donde x es la posición en la base del cerro y está en metros.

Para calcular el área de la sección del cerro, utilice la estrategia de suma de rectángulos de ancho 1 bajo la curva que describen el perfil del cerro. Puede importar \sin y π de la librería `math`.



Actividad 2

Los números de la progresión de Diego son los favoritos de los amantes de la simetría, pues son aquellos que se leen de la misma manera tanto de izquierda a derecha, como de derecha a izquierda. **Por ejemplo:** 1,2,3,4,5,6,7,8,9,22, 33, 2882, 5005, 292, 2882, etc.

Programe un código que imprima todos los números pertenecientes a la progresión de Diego contenidos en el intervalo cerrado de [1, 3000].

Hint: `s[::-1]` invierte un **string**. Por ejemplo si `s` es igual a `'hola'`, `s[::-1]` será `'aloh'`.

Actividad 3

Escriba un programa en Python que estime el valor de π usando la siguiente sumatoria infinita:

$$\pi = 4x \sum_{k=0}^{\infty} s_k, \text{ donde } s_k = \frac{(-1)^k}{2k+1}, \text{ es decir:}$$

$$\pi = 4x \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$$

El programa debe entregar el valor de π , cuando encuentre el término s_k de la sumatoria cuyo valor absoluto sea **menor o igual** a 10^{-5} .

Hint: puede usar el método `abs()` retorna el valor absoluto de un número.

Resumen

¿En dónde estamos?

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

https://docs.python.org/3/reference/lexical_analysis.html

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

<https://docs.python.org/3/library/functions.html>