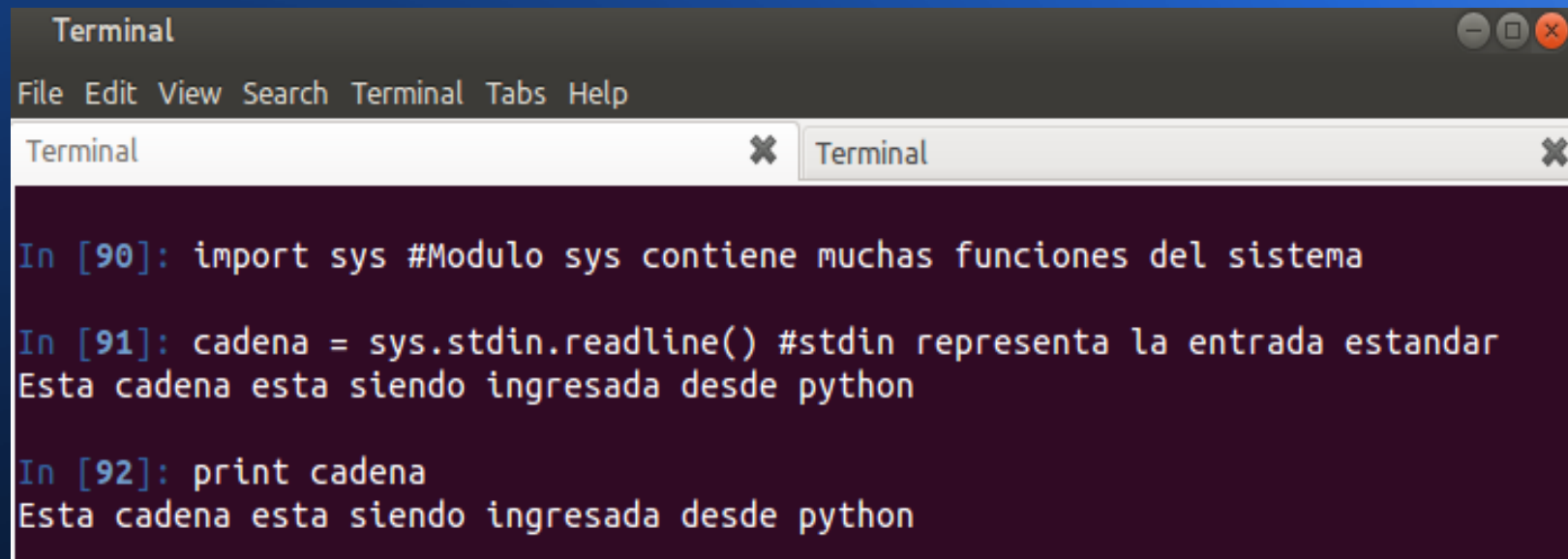


Introducción a la programación con Python (2)

Entrada de datos

- Entrada estándar en tiempo de ejecución
- Parámetros por línea de comandos (al llamar el programa)

Entrada estándar



The image shows a terminal window with a dark background and light-colored text. The window has a title bar with the word "Terminal" and standard window control buttons (minimize, maximize, close). Below the title bar is a menu bar with options: File, Edit, View, Search, Terminal, Tabs, and Help. The main area of the terminal displays three lines of Python code, each preceded by a prompt "In [n]:". The first line imports the sys module. The second line reads a line of input from sys.stdin and stores it in a variable named 'cadena'. The third line prints the value of 'cadena'. Comments are included for each line to explain the code.

```
Terminal
File Edit View Search Terminal Tabs Help

Terminal
Terminal

In [90]: import sys #Modulo sys contiene muchas funciones del sistema

In [91]: cadena = sys.stdin.readline() #stdin representa la entrada estandar
Esta cadena esta siendo ingresada desde python

In [92]: print cadena
Esta cadena esta siendo ingresada desde python
```

Entrada estandar

```
In [94]: valor = int(sys.stdin.readline())
1000

In [95]: type(valor)
Out[95]: int

In [96]: print valor
1000

In [97]: valor = int(sys.stdin.readline())
ola ke ase
-----
ValueError                                Traceback (most recent call last)
/home/jeudy/Proyectos/UCR/intropython0415/<ipython-input-97-7f40246f964b> in <module>()
----> 1 valor = int(sys.stdin.readline())

ValueError: invalid literal for int() with base 10: 'ola ke ase\n'

In [98]: █
```

Entrada estandar

```
In [99]: cadena = raw_input()  
Esta cadena es ingresada usando raw_input  
  
In [100]: print cadena  
Esta cadena es ingresada usando raw_input  
  
In [101]: valor = int(raw_input())  
700  
  
In [102]: type(valor)  
Out[102]: int  
  
In [103]: print valor  
700  
  
In [104]: █
```

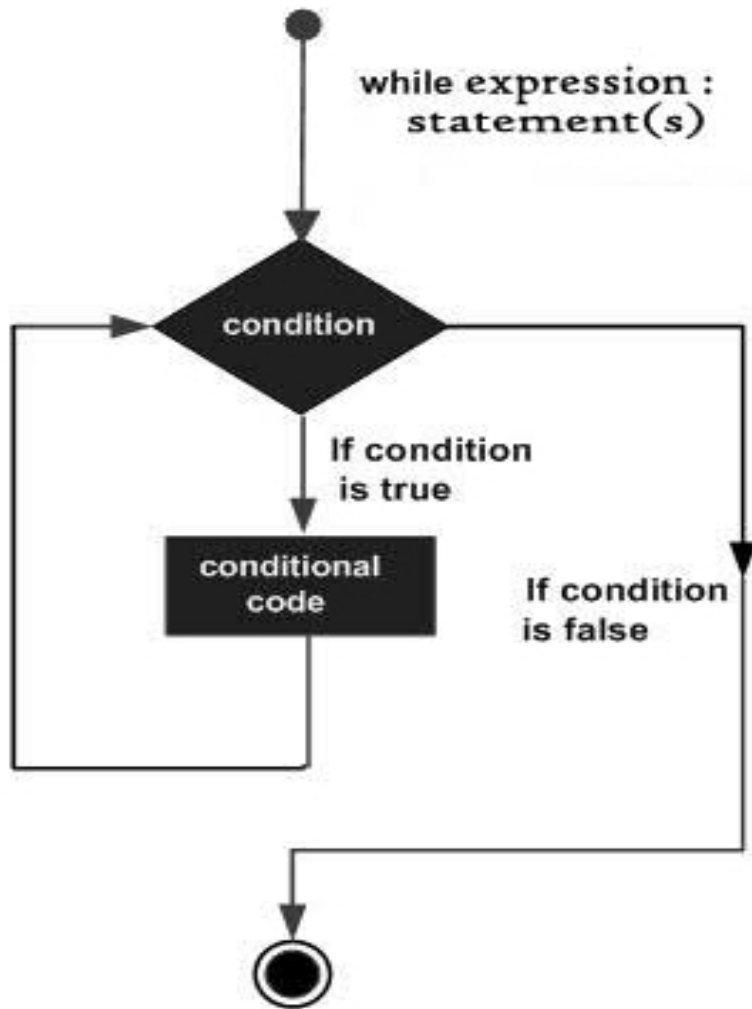
Parámetros de línea de comando

- Para cuando ejecutamos un script desde el shell.
- Utiliza módulo `sys`
- `sys.argv` es una lista que contiene los parámetros enviados. Siempre el primer elemento es el nombre del programa mismo (se puede ignorar).
- Bajar script `ejemplo_parametros.py` del repo.

Control de flujo

- Ejecución condicional de código.
- Si se cumple una condición, entra a un bloque de código.
- Si no se cumple, entra a otro bloque de código.
- Las condiciones se expresan con valores de tipo True y False y operadores lógicos y de comparación.

Control de flujo



Terminal

File Edit View Search Terminal Help

```
In [8]: i = 4
```

```
In [9]: signo = "zero"
```

```
In [10]: if i < 0:
.....:     signo = "negativo"
.....: elif i > 0:
.....:     signo = "positivo"
.....: else:
.....:     signo = "cero"
.....:
```

```
In [11]: print signo
positivo
```

```
In [12]: █
```


Ejercicio

- Existe una lista con todos los números entre 0 y 100 (pista: range?)
- Recorra esa lista (for numero in ...)
- Inserte en una lista todos los números pares.
- Inserte en una lista todos los números impares.
- Dibujemos el diagrama de flujo para plantear el algoritmo.

Sintaxis compacta

```
In [71]: num = 4
```

```
In [72]: var = "par" if (num % 2 == 0) else "impar"
```

```
In [73]: print var  
par
```

```
In [74]: num = 5
```

```
In [75]: var = "par" if (num % 2 == 0) else "impar"
```

```
In [76]: print var  
impar
```

```
In [77]: █
```

Ejemplo: expresiones complejas

- Del repositorio `intropython0415`, haga checkout del archivo `ejemplo_controldeflujo.py`
- Ejecútelo varias veces

Ciclos

- Bloques de código que se ejecutan mientras se cumpla una condición (while).
- ¡Cuidado con las condiciones que nunca se cumplen! (ciclos infinitos)
- Recorrer estructuras compuestas (ej. Listas) y ejecutando un bloque de código para cada elemento.

While

```
In [122]: i = 0

In [123]: while i < 10:
.....:     i += 1
.....:     condition = i < 10
.....:     print "Voy por %d. Condicion actual: %s"%(i, condition)
.....:
Voy por 1. Condicion actual: True
Voy por 2. Condicion actual: True
Voy por 3. Condicion actual: True
Voy por 4. Condicion actual: True
Voy por 5. Condicion actual: True
Voy por 6. Condicion actual: True
Voy por 7. Condicion actual: True
Voy por 8. Condicion actual: True
Voy por 9. Condicion actual: True
Voy por 10. Condicion actual: False
```

While

```
In [125]: valor = ""

In [126]: while valor != "stop":
.....:     valor = raw_input()
.....:     print "Valor dado: %s"%(valor)
.....:
estrella
Valor dado: estrella
planeta
Valor dado: planeta
galaxia
Valor dado: galaxia
stop
Valor dado: stop
stop
Valor dado: stop
```

Ejercicio

- Haga un script que reciba 2 parámetros por línea de comandos: una cadena y un número N.
- Imprima N veces la cadena proporcionada.

Break

- Salirse por fuerza del ciclo

```
In [132]: while True:
.....:     cadena = raw_input()
.....:     if cadena == "stop":
.....:         break
.....:     print "Valor ingresado: %s"%(cadena)
.....:
hola
Valor ingresado: hola
mundo
Valor ingresado: mundo
Stop
Valor ingresado: Stop
pare
Valor ingresado: pare
stop
```


Continue

- Suspendir iteración y pasar a la siguiente

```
In [150]: numeros = range(1,20)

In [151]: print numeros
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

In [152]: for n in numeros:
.....:     if n % 2 == 0:
.....:         continue
.....:     print "No me gustan los pares >_<: %d"%(n)
.....:
No me gustan los pares >_<: 1
No me gustan los pares >_<: 3
No me gustan los pares >_<: 5
No me gustan los pares >_<: 7
No me gustan los pares >_<: 9
No me gustan los pares >_<: 11
No me gustan los pares >_<: 13
No me gustan los pares >_<: 15
No me gustan los pares >_<: 17
No me gustan los pares >_<: 19
```

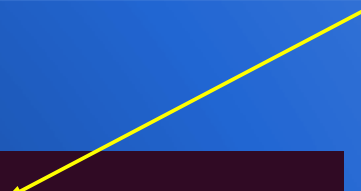
Funciones

- Bloquea el código con un nombre que permite llamarlo
- Puede recibir argumentos
- Puede devolver un valor

```
In [159]: def imprimirN(cadena, n):  
.....:     while n > 0:  
.....:         print cadena  
.....:         n -= 1  
.....:
```

```
In [160]: imprimirN("ola ke ase", 5)  
ola ke ase  
ola ke ase  
ola ke ase  
ola ke ase  
ola ke ase
```


Cantidad variable de parámetros



```
In [175]: def sumatoria(n1, n2, *resto):
.....:     resultado = n1 + n2
.....:     for x in resto:
.....:         resultado += x
.....:     return resultado
.....:

In [176]: print sumatoria(10,20)
30

In [177]: print sumatoria(1,2,3,4,5,6,7,8,9,10)
55
```

Parámetros por valor o referencia

- Tipos primitivos por valor
- Objetos y colecciones (excepto tuplas) por referencia.

```
#Asume que x, y son valores numéricos
def multiplicar(x, y):
    x = x * y
    return x

#Recibe un diccionario al cual, a cada valor, le suma el valor dado.
#Asume que los valores del diccionario dado son numeros
#No devuelve nada, el diccionario se modifica por referencia
def sumar_valor(valor, diccionario):
    for k in diccionario.keys():
        diccionario[k] += valor
```

Parámetros por valor o referencia

```
In [207]: x = 10
```

```
In [208]: y = 20
```

```
In [209]: print x, y  
10 20
```

```
In [210]: multiplicar(x,y)  
Out[210]: 200
```

```
In [211]: print x, y  
10 20
```

```
In [213]: dic = {"uno":1, "dos":2, "tres":3}
```

```
In [214]: print dic  
{'dos': 2, 'tres': 3, 'uno': 1}
```

```
In [215]: sumar_valor(10, dic)
```

```
In [216]: print dic  
{'dos': 12, 'tres': 13, 'uno': 11}
```

Módulos

- Funciones agrupadas que pueden ser importadas desde otros programas.
- Un módulo se crea fácil: un script de python conteniendo funciones.
- Se importa con ***import*** y el nombre del script, sin la extensión.
- Script debe estar en directorio actual, o dentro de PYTHONPATH (similar a PATH)

Módulos

```
1#!/usr/bin/python
2# -*- coding: UTF-8 -*-
3
4
5"""
6ejemplos_funciones.py
7Modulo con ejemplos de funciones
8"""
9
10def sumatoria(n1, n2, *resto):
11    resultado = n1 + n2
12    for x in resto:
13        resultado += x
14    return resultado
15
16def imprimirN(cadena, n=10):
17    while n > 0:
18        print cadena
19        n -= 1
20
21#Asume que x, y son valores numéricos
22def multiplicar(x, y):
23    x = x * y
24    return x
25
26#Recibe un diccionario al cual, a cada valor, le suma el valor dado.
27#Asume que los valores del diccionario dado son numeros
28#No devuelve nada, el diccionario se modifica por referencia
29def sumar_valor(valor, diccionario):
30    for k in diccionario.keys():
31        diccionario[k] += valor
32
```

```
In [2]: from ejemplos_funciones import sumatoria, imprimirN
```

```
In [3]: imprimirN("Hola",2)
```

```
Hola
```

```
Hola
```

```
In [4]: multiplicar(3,4)
```

```
NameError
```

```
Traceback (most recent call last)
```

```
/home/jeudy/Proyectos/UCR/intropython0415/<ipython-input-4-f8c259c401d5> in <module>()
----> 1 multiplicar(3,4)
```

```
NameError: name 'multiplicar' is not defined
```

```
In [13]: import ejemplos_funciones
```

```
In [14]: dir(ejemplos_funciones)
```

```
Out[14]:
```

```
['__builtins__',
 '__doc__',
 '__file__',
 '__name__',
 '__package__',
 'imprimirN',
 'multiplicar',
 'sumar_valor',
 'sumatoria']
```


Ejercicios

- Cree un nuevo repositorio ejercicios_intro
- Cree un modulo funciones_intro.py
- Programe las siguientes funciones:
 - def es_primo(N): devuelva true o false dependiendo de si el número es primo.
 - def es_bisiesto(N): devuelva true o false dependiendo de si el año dado es bisiesto. Si el año es menor a 1900 o mayor a 3000, devuelva false e imprima un error.

Ejercicios

- Cree un nuevo programa prueba_primos.py que importe del módulo funciones_intro a la función es_primo.
- Agréguelo al repositorio ejercicios_intro
- Dentro del programa defina una función revisar_primos que reciba una cantidad variable de números.
- La función debe recorrer la lista y **devolver** cuantos números dentro de la lista, son primos.
- Por cada primo encontrado, imprímalo.

Ejercicios

- En el cuerpo principal del programa, defina una lista vacía y pídale al usuario que inserte elementos en esta, y se detenga al ingresar “stop”.
- Pase la lista definida por las entradas del usuario, a la función `revisar_primos`.

Ejercicios

- Cree un nuevo programa prueba_bisiestos.py
- El programa imprime los años que son bisiestos entre 2 años dados.
- Importe del módulo funciones_intro, la función es_bisiesto.
- El programa recibe 2 parámetros por línea de comandos. Debe recorrer todos los años intermedios, revisar cuales son bisiestos, e imprimirlos.