Errores y excepciones

Hay (al menos) dos tipos diferentes de errores: *errores de sintaxis* y *excepciones*.

Errores de Sintaxis:

Los errores de sintaxis, también conocidos como errores de interpretación.

El intérprete repite la línea culpable y muestra una pequeña 'flecha' que apunta al primer lugar donde se detectó el error. Este es causado por (o al menos detectado en) el símbolo que precede a la flecha: en el ejemplo, el error se detecta en el print, ya que faltan dos puntos (':') antes del mismo. Se muestran el nombre del archivo y el número de línea para que sepas dónde mirar en caso de que la entrada venga de un programa.

```
>>> while True print 'Hola mundo'
Traceback (most recent call last):
...
    while True print 'Hola mundo'
                   ^
SyntaxError: invalid syntax
```

Hay (al menos) dos tipos diferentes de errores: *errores de sintaxis* y *excepciones*.

Excepciones:

Los errores detectados durante la ejecución se llaman excepciones, y no son incondicionalmente fatales. Sin embargo, la mayoría de las excepciones no son manejadas por los programas, y resultan en mensajes de error como los mostrados aquí:

```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
ZeroDivisionError: integer division or modulo by zero
>>> 4 + spam*3
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: name 'spam' is not defined
>>> '2' + 2
```

Errores y excepciones

Manejando excepciones:

Es posible escribir programas que manejen determinadas excepciones, para ello se declara try.

La declaración try funciona de la siguiente manera:

- **Primero, se ejecuta el bloque try (el código entre las declaración try y except).**
- **Si no ocurre ninguna excepción, el bloque except se saltea y termina la ejecución de la declaración try.**
- **Si ocurre una excepción durante la ejecución del bloque try, el resto del bloque se saltea. Luego, si su tipo coincide con la excepción nombrada luego de la palabra reservada except, se ejecuta el bloque except, y la ejecución continúa luego de la declaración try.**
- **Si ocurre una excepción que no coincide con la excepción nombrada en el except, esta se pasa a declaraciones try de más afuera; si no se encuentra nada que la maneje, es una excepción no manejada, y la ejecución se frena con un mensaje**

```
import sys

try:
    f = open('miarchivo.txt')
    s = f.readline()
    i = int(s.strip())
except IOError as (errno, strerror):
    print "Error E/S ({0}): {1}".format(errno, strerror)
except ValueError:
    print "No pude convertir el dato a un entero."
except:
    print "Error inesperado:", sys.exc_info()[0]
    raise
```

Funciones

Una función es una sección de un programa que calcula un valor de manera independiente al resto del programa.

Una función tiene tres componentes importantes:

- **los parámetros, que son los valores que recibe la función como entrada;**
- **el código de la función, que son las operaciones que hace la función; y**
- **el resultado (o valor de retorno), que es el valor final que entrega la función.**

En esencia, una función es un mini programa.

La palabra reservada def se usa para definir funciones. Debe seguirle el nombre de la función y la lista de parámetros formales entre paréntesis. Las sentencias que forman el cuerpo de la función empiezan en la línea siguiente, y deben estar con sangría.

```
def sumador(a,b):
    c=a+b
    return c

#Programa
x=raw_input('Primer Numero: ')
y=raw_input('Segundo Numero: ')
suma=sumador(x,y)
```

Consideraciones sobre los parámetros y retornos:

- **Una función puede o no recibir parámetros**
- **Puede recibir mas de un parámetro y se los indica separando por comas**
- **Para llamar la función se debe poner el nombre de la misma y entre paréntesis los parámetros. La cantidad de parámetros que se envían debe coincidir con las solicitadas.**
- **No es necesario que los nombres de parámetros en la llamada y en la función, solo se debe respetar el orden definido**
- **Una función no necesariamente debe retornar un valor**
- **Puede retornar mas de un valor y estos se devuelven como una lista**

Funciones - Argumentos

```
#Variables directas
def f(a, b, c):
    return a + b*c

#Llamada
>>> f(2, 5)
TypeError: f() takes exactly 3 arguments (2 given)
>>> f()
TypeError: f() takes exactly 3 arguments (0 given)
>>> f(2, 5, 3)
17

#Con valores por defecto
def h(a, b=4, c=2):
    return a + b*c

#Llamada
>>> h(1)   # a=1, b=4 y c=2
9
>>> h(1, 5, 6)   # a=1, b=5 y c=6
31
>>> h()
TypeError: h() takes at least 1 argument (0 given)
```

Funciones - Argumentos

```
"""
Para que la función pueda tomar una cantidad
indefinida de argumentos se puede utilizar :
"""

*args

def f(*args):
    return args
#LLamada
>>> f(1, 5, True, False, "Hello, world!")
(1, 5, True, False, 'Hello, world!')

"""
Para enviar una cantidad indefinida de argumentos pero
que puedan ser identificados por clave se usa :
"""
**kwargs

def f(**kwargs):
    return kwargs

#Llamada
>>> f(a=1, b=True, h=50, z="Hello, world!")
{'a': 1, 'h': 50, 'b': True, 'z': 'Hello, world!'}
```

Ejercicio con archivos de texto 02

Definición

Generar un modulo en Python denominado filtroTexto.py que:
Cree 2 archivos universitarios.csv y certificados.csv
Lea el contenido del archivo asistentes.csv

- Desglose el contenido de cada línea cuyo carácter de separación es el ;(punto y coma) en los campos : nom, ape, uni, cer
- Analizar el campo **uni** , que indica si la persona es universitaria(SI) o no(NO) y si es universitaroia grabar la linea completa en el archivo universitarios.csv
- Analizar el campo **cer** , que indica si el participante quiere(SI) o no(NO) certificado, si lo quiere grabar la línea completa en el archivo certificados.csv

Ejercicio con funciones

Ejercicio
Funciones:

- **Lectura de archivo de texto para desplegar Menu(Menu)**

Opciones

1. **Solicitud de Texto y comprobación si es PALINDROMO(**Palabra o expresión que es igual si se lee de izquierda a derecha que de derecha a izquierda**)**
2. **Solicitud de Texto y calculo de promedio de letras en las palabras del texto ingresado con precisión de 2 decimales**

(Ambos desarrollos deben hacerse en funciones)

Procesos Principal:

- **-En un loop infinito:**
  - **Despliega el menú por medio de una función**
  - **Solicita se ingrese una opción**
  - **Analiza la opción**
  - **Si es 99, termina el programa**
  - **Si es menor a 99 , llama a la función que corresponde donde se solicitan datos y despliega resultado**
  - **Solicita <enter> para continuar**
  - **Vuelve al despliegue del menú hasta que la opción sea 99**