# 1

Condición inesperada o inusual, que surge durante la ejecución del programa y no puede ser manejada en el contexto local

Una excepción es una situación anómala que se da en la ejecución de un programa y que se supone que ocurre con poca frecuencia.

# 2

* Un modo de definirlas
* Una forma de alcanzarlas, invocarlas
* Una forma de manejarlas
* Un criterio de continuación

Los modernos si, si usas COBOL capaz que no

# 3

Con cualquier error se rompería

Se podrían plantear procesos que se ejecuten al detectar un error, los cuales tendrían el papel de controlador siempre y cuando la unidad que levanto el error tenga acceso al proceso en cuestion

# 4

1. Terminación y Reasuncion

Terminacion: todos menos pl/1

Reasuncion: ninguno menos pl/1

Python

¿Cómo se define?

Los bloques que pueden llegar a levantar excepciones van precedidos por la palabra clave Try y al finalizar el bloque se detallan los manejadores utilizando la palabra clave except(NombreDeLaExcepción)

¿Cómo se lanza?

Con un “raise”

¿Cómo se maneja?

Con el codigo en el bloque except

¿Cuál es su criterio de continuación?

Sigue después del bloque try-except

Pl/1

¿Cómo se define?

Se definen con la instrucción ON CONDITION nombre Manejador

¿Cómo se lanza?

Se levantan con la instrucción SIGNAL CONDITION nombre

¿Cómo se maneja?

Alcance de una excepción y búsqueda del manejador: a medida que se va ejecutando el proceso y se encuentran ON CONDITION se apila el manejador en una pila de manejadores, cunado se levanta una excepción se busca el nombre de la misma desde el tope de la pila hacia abajo, el primer manejador que se encuentra es el que se ejecuta.

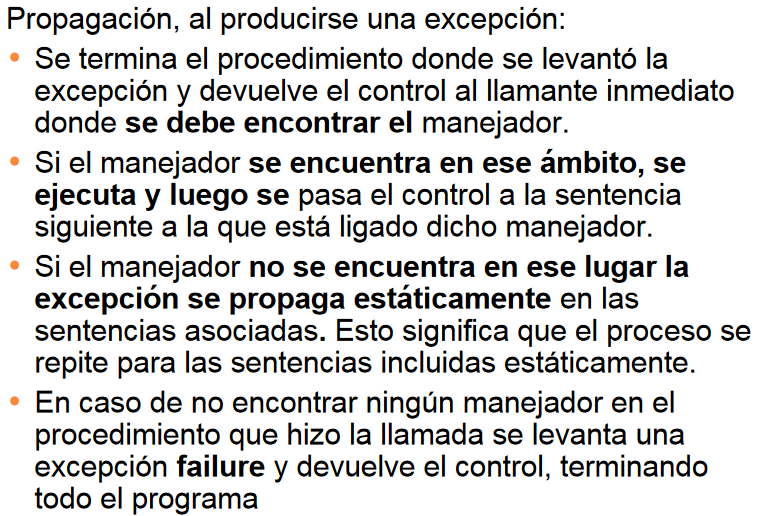
¿Cuál es su criterio de continuación?

Continúa el flujo de ejecución en el punto siguiente a donde se levantó la excepción.

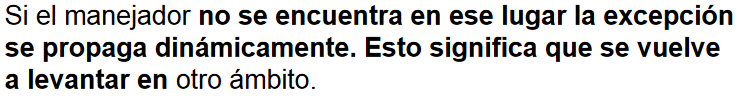
El de reasunción, porque se arregla una expcion concreta y al volver a la unidad esta puede generar la misma mas veces si por ejemplo esto se encontraba en una iteracion

# 5

## CLU



## ADA



## PL/1

Busca el ultimo manejador de la excepción en la pila y si no lo encuentra el programa da error

## Python

si no se encuentra nada que la maneje, es una excepción no manejada, y la ejecución se frena con un mensaje de error

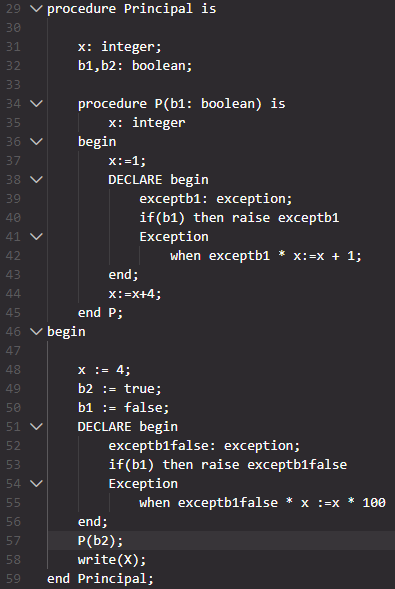
# 6

Se podría decir que sigue el modelo de reasunción ya que en principio no se termina la ejecucion de la unidad que levanta la excepción, en este caso “P”, una forma de si cortar la ejecucion una vez levantado el error, es que todo el codigo siguiente a este se encuentre en la clausula else del if, otra posible forma es una vez ejecutado el manejador “X” hacer un return

# 7

# a

## B



# 8

1. Se llama al proceso 2, luego al proceso 1 donde se levanta la excepción (signal tipo1) por lo que el proceso 1 termina y busca al manejador en el tipo 2 dentro del proceso 2, primero en en el if donde se invoco al proceso 1, como no esta, termina el bloque del if y se busca en el While, como tampoco esta se termina el bloque del While, sale y en el codigo principal al finalizar el While encuentra el manejador y lo ejecuta imprimiendo

“Se produjo un error de tipo2 en Proc\_2”, continua la ejecucion en la línea siguiente al bloque except que contiene el manejador.

1. En el proceso 1 habria que levantar la excepción tipo1 usando el “raise”, luego en el proceso dos, para simular la búsqueda estatica por bloques del manejador, habria que poner un declare por cada bloque, también seria necesario definir la excepción como una variable global, asi al levantarla en el proceso 1 no se propagaría como una excepción anónima

# 9

## Diferencias

Python

Las excepciones son objetos de la clase Exception

No hay que hacer nada

Tiene else

Java

Las excepciones son objetos de la clase Throwable

Hay que especificar mediante la clausula throws todas las excepciones que se envía desde un método

No cuenta con una clausula else

## Similitudes

|  |  |
| --- | --- |
| Python | Java |
| Usa bloques try except | Se llaman try catch |
| Tiene bloque finaly |  |
| levanta excepciones con raise | Con throw |
| Si no tiene el manejador va al siguiente bloque try estáticamente luego dinamicamente entre funciones | |

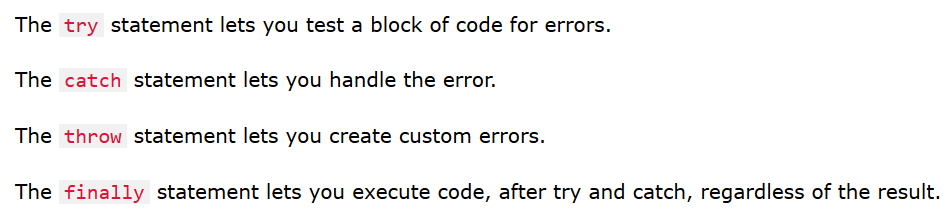
# 10

Implementa el modelo por terminación

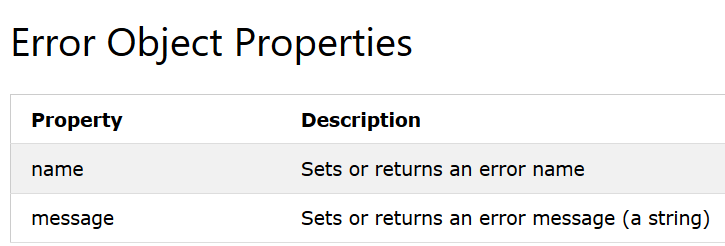
Consta de las instrucciones

* Begin
  + Comienza el bloque donde se podrán levantar excepciones y manejarlas
* Rescue
  + Sirve para captar una excepción y definir un bloque de codigo que la maneje
* Else
  + Capta otras excepciones no definidas en los rescue
* End
  + Termina el bloque begin, luego de ejecutar el bloque rescue se continua despues de esta línea
* Raise
  + Levanta una excepción

# 11



The exception can be a JavaScript String, a Number, a Boolean or an Object



Se propaga estáticamente entre bloques try catch y dinámicamente entre funciones

# 12

Asumiendo que b = true

* Principal
* prueba1(y)
* se levanta la excepción X
* termina prueba 1
* como no hay manejador busca dinámicamente en principal
* ejecuta el manejador X de principal y continua post bloque exception
  + y = 30

asumiendo que b = false

* principal
* prueba1(y)
* prueba2
* levanta la excepción división por 0
* se atiende en el manejador local de prueba 2
* ejecuta el manejador Constraint\_error y continua post bloque exception

para agregar una excepción anónima, se podría declarar X en prueba1 y en principal poner un manejador para others

# 13

Asumiendo m = 0, y = 0

* se ejecuta MAIN
* llama Uno()
* levanta error 1 y va a buscar dinamicamente en Main
* ejecuta el manejador y continua post bloque exception
* se llama a dos
* levanta error 1 y va a buscar dinamicamente en Main
* encuentra manejador vuelve a levantar error1
* termina el bloque y como no hay nada mas termina el programa en error

asumiendo m = 1, y = 0

* se ejecuta MAIN
* llama Uno()
* levanta error 1 y va a buscar dinamicamente en Main
* ejecuta el manejador y continua post bloque exception
* se llama a dos y termina
* termina el main

asumiendo m = 0, y = 1

* se ejecuta MAIN
* llama Uno()
* llama a dos dentro del While
* dos levanta error 1
* temrina dos y busca dinámicamente en UNO

terminar

# 14