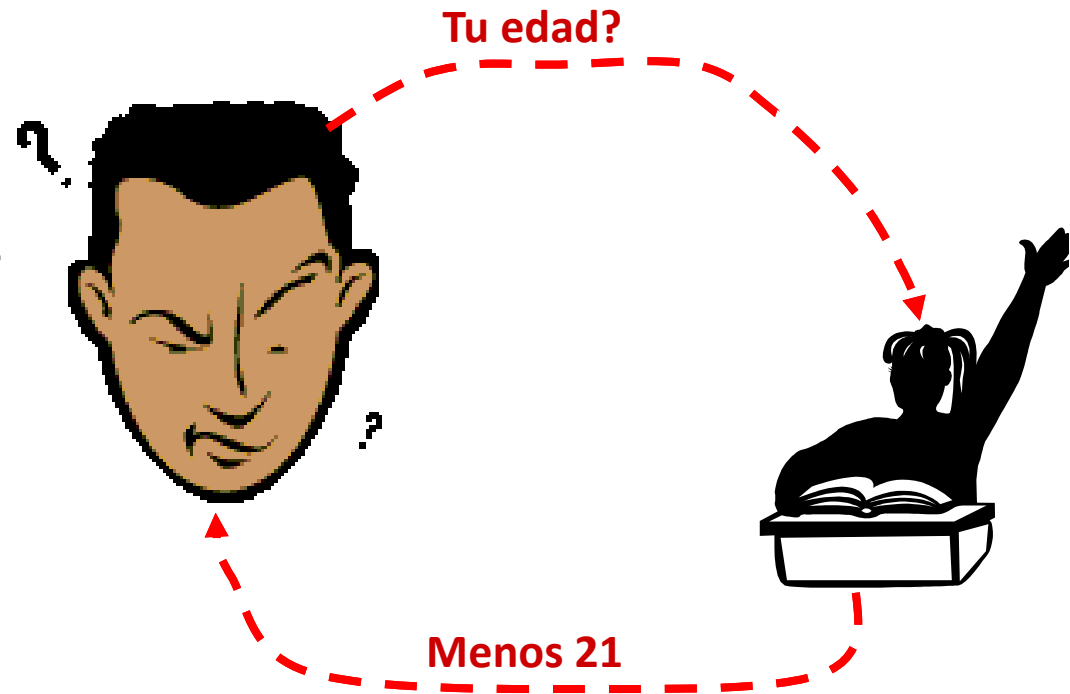


PYTHON

UNI – FIIS
Glen Rodríguez R.

Bucles (loops)

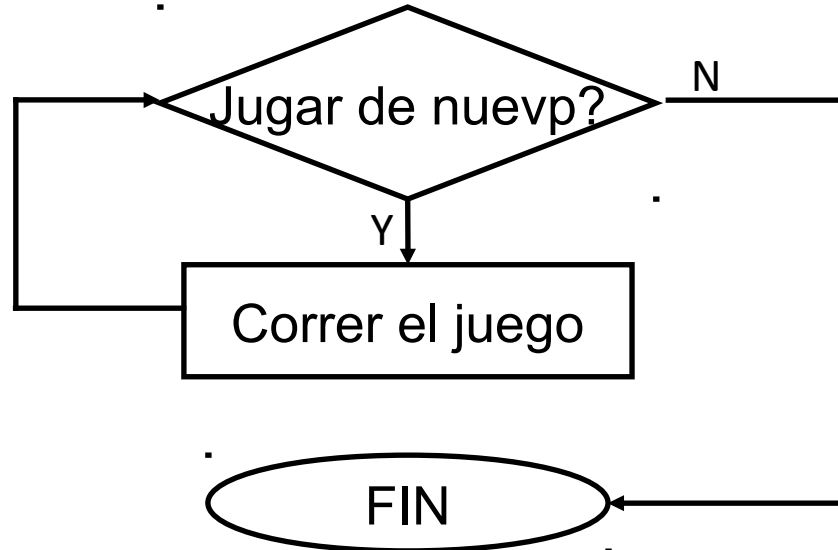
- Desde el punto de vista de una computadora: repetir una acción hasta que cierta condición se cumpla.
- Ejemplo: pedir la edad mientras que la respuesta sea negativa (dejar de repetir cuando la respuesta sea 0 o positiva)





Volver a jugar (volver a correr el programa)

Flowchart

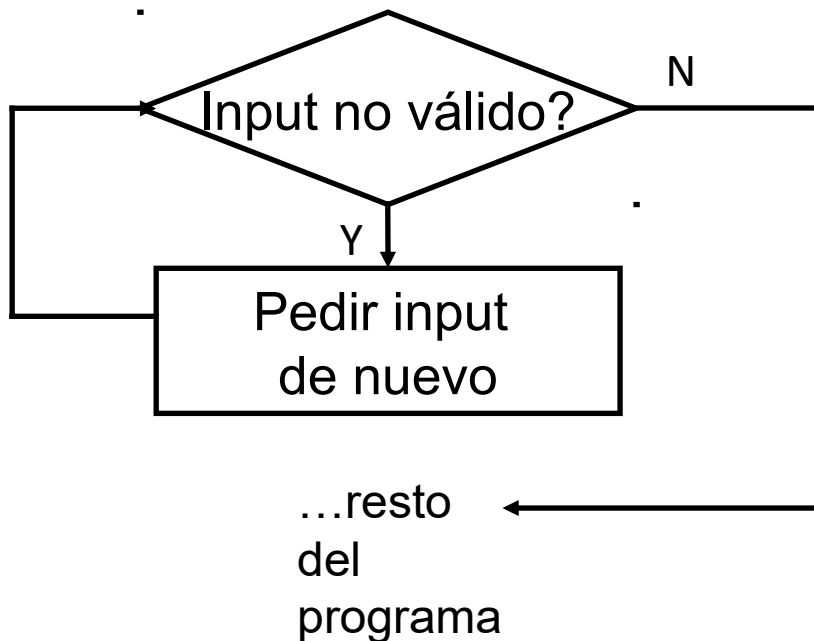


Pseudocódigo

Mientras el jugador quiere volver a jugar
Correr el juego de nuevo

Volver a correr sólo una parte
específica del programa

Flowchart



Pseudocódigo

Mientras input es inválido

Pedir input al usuario

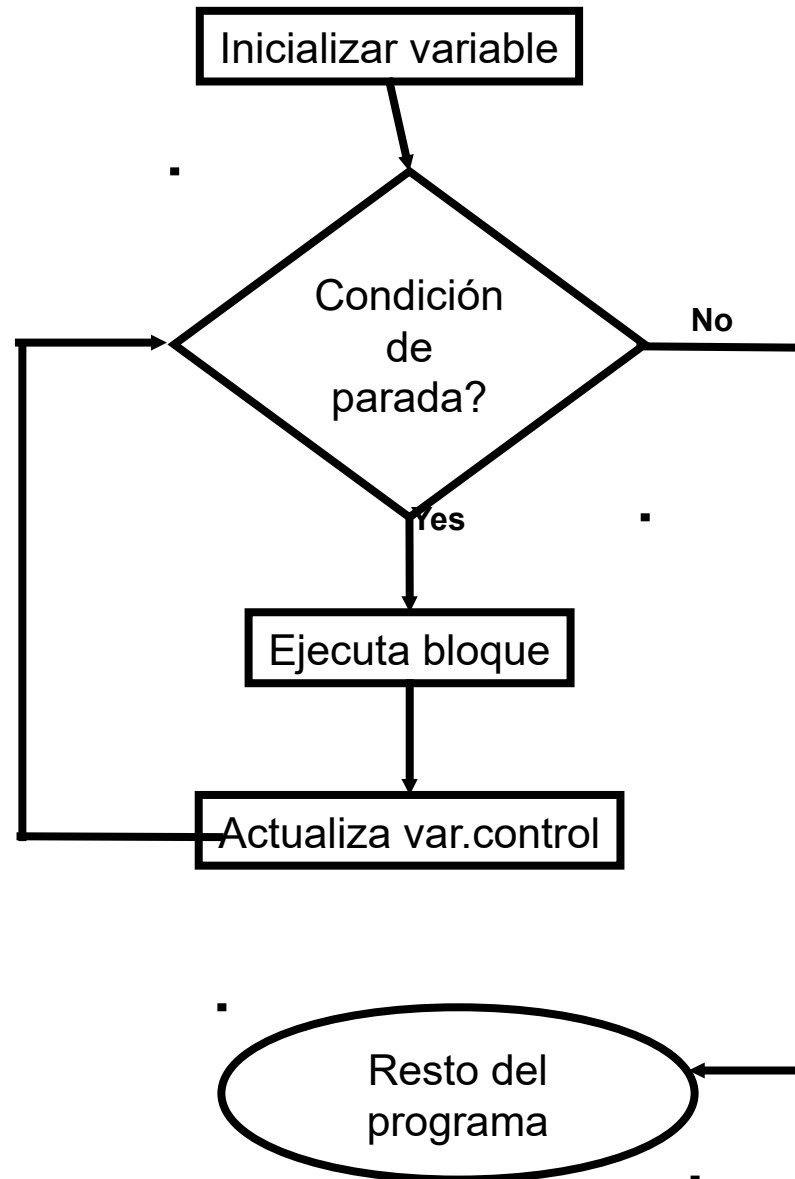
Estructura básica de un bucle

- Control del bucle: por medio de una variable (generalmente)
 - Inicializar la variable a un valor inicial
 - Ver si la variable cumple la condición de parada (espr.booleana)
 - Ejecutar el bloque del bucle (la parte que se repite)
 - Actualizar la variable de control

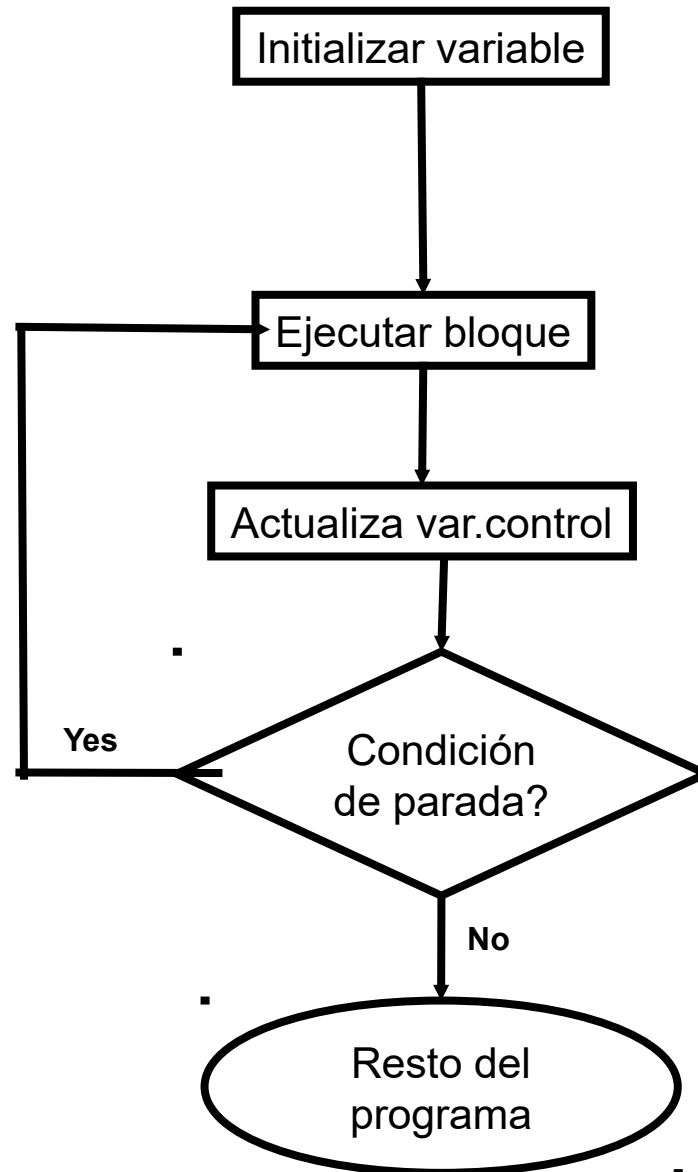
Tipos de bucle

- Pre-test
 - Chequea la condición de parada ANTES de ejecutar el bloque
 - El bucle se ejecuta cero o más veces
- Post-test
 - Chequea la condición de parada DESPUES de ejecutar el bloque
 - El bucle se ejecuta una o más veces

Pre-test



Post-test (no existen en Python)



Bucles en Python

- While
- For
- La condición de parada se chequea antes del bloque
- Puede ejecutarse cero veces si la condición no se cumple en la primera verificación

While

- Se puede usar si no se sabe de antemano cuantas veces se repetirá el bucle (es el tipo más general de loop, el "for" puede reemplazarse por un "while").
- Se repite mientras lo opuesto a la condición de parada sea verdadero
- Formato:

(Condición simple)

```
while (expr.booleana):  
    bloque
```

(Condición compuesta)

```
while (exp.booleana) operador (exp.booleana):  
    bloque
```

Ejemplo

- while1.py

```
i = 1
```

1) Inicializar



```
while (i <= 3):
```

2) Chequea
condición



```
    print("i =", i)
```

3) Ejec.bloque



```
    i = i + 1
```

```
print("Hecho!")
```

4) Actualizar control



Ejemplos

- while2.py

```
i = 3
while (i >= 1):
    print("i =", i)
    i = i - 1
print("Hecho!")
```

Errores comunes

- No inicializar la variable de control
- Olvidarse de actualizar la variable

```
i = 1
while(i <= 4):
    print("i =", i)
```



```
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
```

Ejercicios

- Pedir la edad hasta que se entre una edad válida. Válida: no negativa y máximo 120 años
- Sumar números hasta entrar “fin” en vez de un número
- Promediar números hasta entrar “fin” en vez de un número

For

- Para “contar” en una secuencia
- Formato

```
for <nombre variable control> in <algo iterable>:  
    bloque
```

- Ejemplo (for1.py)

```
i = 0  
  
total = 0  
  
for i in range (1, 4, 1):  
    total = total + i  
    print("i=", i, "\tttotal=", total)  
  
print("Hecho!")
```

Ejemplo

- for2.py

```
i = 0
```

```
total = 0
```

```
for i in range (3, 0, -1):
```

```
    total = total + i
```

```
    print("i = ", i, "\t total = ", total)
```

```
print("Hecho!")
```


Iterando en una secuencia de caracteres

- for3.py

```
actividad = input("Que haces ahora? ")
```

```
print("Estoy '", end="")
```

```
for c in actividad:
```

```
    print(c + "-", end="")
```

```
print("")
```

Errores en for

- Nunca corre por la condición mal hecha

```
for i in range (5, 0, 1):  
    total = total + i  
    print("i = ", i, "\t total = ", total)  
print("Done!")
```

Los bucles pueden ser variados

```
i = 0
```

```
while (i <= 100):
```

```
    print("i =", i)
```

```
    i = i + 5
```

```
print("Hecho!")
```

```
for i in range (0, 105, 5):
```

```
    print("i =", i)
```

```
print("Hecho!")
```

range()

- Es una función que genera una lista de números
- `range(n)` genera n números del 0 al n-1
- `range([start], stop[, step])` genera números que comienzan en start y se incrementan de step, hasta stop (pero sin incluir stop). Step por default es 1
- `Range(2,5)` genera 2,3,4

Bucles controlados por sentinela

- Una variable sentinela controla el bucle. Cuando se cambia a cierto valor termina el bucle

```
total = 0
temp = 0
while(temp >= 0):
    temp = input ("Entre un número positivo (negativo para terminar): ")
    temp = int(temp)
    if (temp >= 0):
        total = total + temp

print("Suma total:", total)
```

Ejemplo

```
seleccion = " "  
  
while seleccion not in ("a", "A", "r", "R", "m", "M", "s", "S"):  
    print("Menu options")  
    print("(a)ñadir un jugador")  
    print("(r)emover un jugador")  
    print("(m)odificar un jugador")  
    print("(s)alir del juego")  
    seleccion = input("Entre su opcion: ")  
  
if seleccion not in ("a", "A", "r", "R", "m", "M", "s", "S"):  
    print("Solo puede entrar 'a', 'r', 'm' o 's' ")
```

Break

- Usado para salir de un bucle de forma súbita. Debe asociarse a otra condición lógica

<pre>for (Condition 1): if (Condition 2): break</pre>	<pre>while (Condition 1): if (Condition 2): break</pre>
---	---

```
str1 = input("Entre una cadena de letras minúsculas: ")  
for temp in str1:  
    if (temp < 'a') or (temp > 'z'):  
        break  
    print(temp)  
print("Hecho")
```

Ojo

- No usar break a menos que sea absolutamente necesario
- Un bucle debería tener una sola salida (condición de parada). Hacer otra salida hace difícil seguir la lógica del programa (programa spaghetti)

Ojo

- Mejor usar una condición de parada compuesta:

```
while (BE1):  
    if (BE2):  
        break
```

```
while (BE1) and not (BE2)
```

Ojo

- Otra alternativa: usar flags

```
flag = true
while (flag == true):
    if (BE1):
        flag == false
    if (BE2)
        flag == false
# caso contrario el flag sigue siendo true
```

Bucles anidados

Bucle externo (corre n veces)

 Bucle interno (corre m veces)

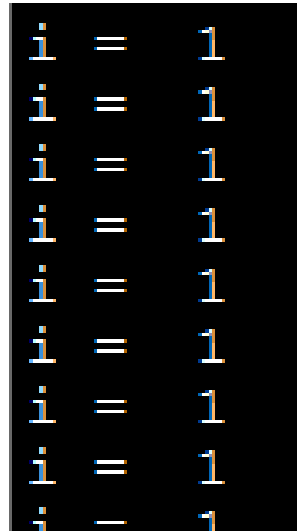
 bloque del bucle interno (corre nxm veces)

- Ejemplo (nested.py)

```
i = 1
while (i <= 2):
    j = 1
    while (j <= 3):
        print("i = ", i, " j = ", j)
        j = j + 1
    i = i + 1
print("Fin")
```

Cuidado con el bucle infinito

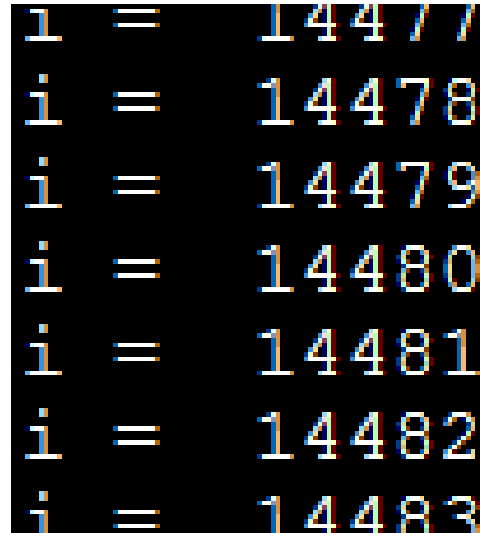
```
i = 1
while (i <= 10):
    print("i = ", i)
    i = i + 1
```



```
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
```

Cuidado con el bucle infinito

```
i = 10
while (i > 0):
    print("i = ", i)
    i = i + 1
print("Done!")
```



```
i = 14477
i = 14478
i = 14479
i = 14480
i = 14481
i = 14482
i = 14483
```

Ejercicio

- Hacer un programa que pide día y mes de nacimiento.
- El mes debe ser de 1 a 12.
- El día no puede menor de 1, ni ser mayor que el máximo de días del mes correspondiente (ejemplo, para el mes 4, máximo es 30)
- Asumir año no bisiesto

Pseudocódigo

While (mes no está entre 1 y 12)

 Pedir mes al usuario

If (mes es uno de 31 días) then

 While (dia no está entre 1 y 31)

 Pedir dia al usuario

If (mes es uno de 30 días) then

 While (dia no está entre 1 y 30)

 Pedir dia al usuario

If (mes es uno de 28 días) then

 While (dia no está entre 1 y 28)

 Pedir dia al usuario

Mostrar mes y dia

Solución

- (calendario.py)

ENE = 1

FEB = 2

MAR = 3

ABR = 4

MAY = 5

JUN = 6

JUL = 7

AGO = 8

SEP = 9

OCT = 10

NOV = 11

DIC = 12

Solución

```
# mes y luego dia
# una vez entrado mes no se puede cambiar
mes = -1
while (mes < ENE) or (mes > DIC):
    mes = int(input("Entre el mes (1 - 12): "))
    if (mes < ENE) or (mes > DIC):
        print("Mes debe estar en el rango 1 - 12")

# Meses de 31 dias
if (mes in (ENE,MAR,MAY,JUL,AGO,OCT,DIC)):
    dia = -1
    while (dia < 1) or (dia > 31):
        dia = int(input("Entre el dia (1 - 31): "))
        if (dia < 1) or (dia > 31):
            print("Dia debe estar enre 1 - 31")
```

Solución

```
elif (mes in (ABR,JUN,SEP,NOV)):  
    dia = -1  
    while (dia < 1) or (dia > 30):  
        dia = int(input("Entre el dia (1 - 30): "))  
        if (dia < 1) or (dia > 30):  
            print("Dia debe ser en el rango 1 - 30")
```

```
elif (mes == FEB):  
    dia = -1  
    while (dia < 1) or (dia > 28):  
        day = int(input("Entre el dia (1 - 28): "))  
        if (dia < 1) or (dia > 29):  
            print("Dia debe ser en el rango 1 - 29")
```

```
print()  
print("Mes de nac.: %d" %mes)  
print("Dia de nac.: %d" %dia)
```

Nota

- Este programa es un ejemplo de uso adecuado de constantes.
- Permiten “autodocumentación” del código fuente.
- Leyendo se entiende más rápido.
- Achica el programa

```
# Months with 31 days  
# January, March, May, July, August, October, December  
if (month in (1,3,5,7,8,10,12)):
```

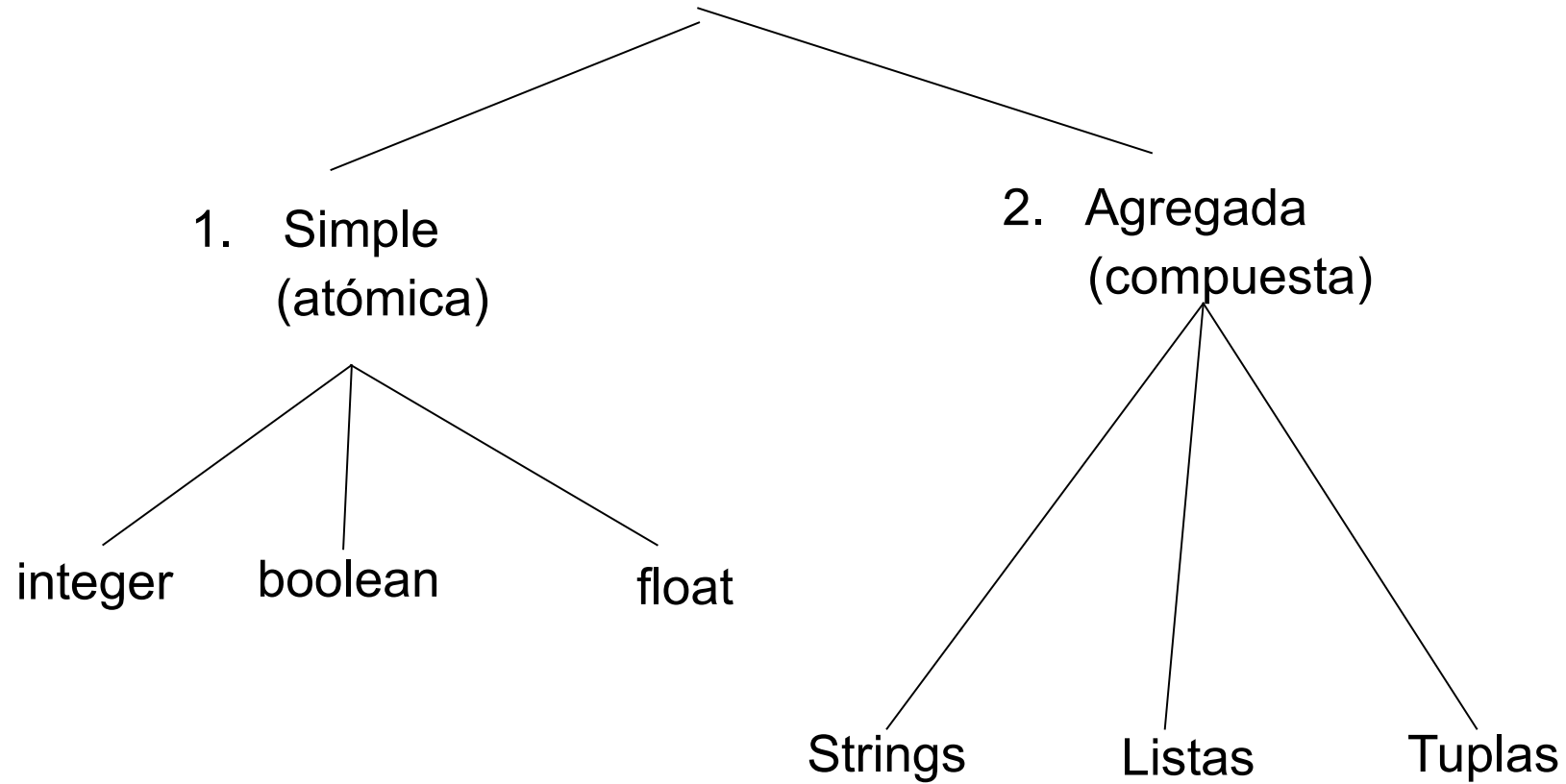
Ejercicio

- Cambie el programa para permitir corregir el mes cuando se está preguntando por el día
 - Si la combinación mes/día es correcta, imprime
 - Pero si no, da la oportunidad de corregir el día o corregir el mes

Tipos compuestos

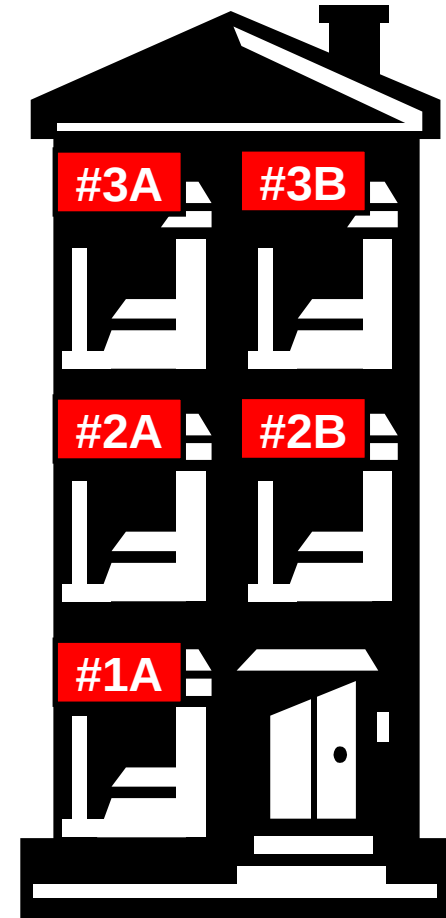
- Variables que están compuestas por otras variables
- Variable no compuesta: `x=567`
- Variable compuesta : `y="hola"`
- El más popular tipo compuesto son las secuencias, que a su vez pueden ser listas, tuplas o strings

Variables en Python



Ejemplo

- Una dirección (Av. Universitaria 1234) es una variable compuesta
- En realidad en esa dirección hay un edificio de departamentos. Cada departamento es una variable simple



Strings

- Secuencia de caracteres
- Se puede acceder a un carácter usando [] y el índice del carácter (desde 0 → 1er carácter)
- Si un string tiene n caracteres, y trato de acceder el [n] o superior: error

Ejemplo de strings

```
index = int(input("Entre posición (0-13): "))
name = "Homero Simpson"
print(name)
if (index >= 0) and (index <= 13):
    print(name[index])
else:
    print("%d está fuera del rango 0 - 13" %index)
```

Inmutabilidad

- Una vez creado, un string en Python no se puede modificar
- Si hago
 - `var="hola"`
 - `var=var+ " alumno"`
- Se destruye el hola y se crea otra variable (casilla de memoria) con "hola alumno"

Sub-strings

```
var = "hello world"
```

```
#str[start:end]
```

```
print(var[0])
```

```
print(var[1:5])
```

```
print(var[2:])
```

```
print(var[:5])
```

```
#output
```

```
.....
```

```
h
```

```
ello
```

```
llo world
```

```
hello
```

```
.....
```

Separando strings

- `str.split([sep [,maxsplit]])`
- Por default el separador es el espacio en blanco
- El 2do parámetro es el máximo número de pedazos en que se parte el string, por defecto es ilimitado

Ejemplo

```
name = "James"  
name = name + "T. Kirk"  
print(name)  
print(name[0:5])  
print(name[2:])  
print(name[:3])  
first, second = name.split('.')  
print(second,first)
```

Ejemplo

```
aString = "man who smiles"  
  
# Default split character is a space  
  
one, two, three = aString.split()  
print(one)  
print(two)  
print(three)  
  
aString = "James,Tam"  
first, last = aString.split(',')  
nic = first + " \"The Bullet\" " + last  
print(nic)
```

Iterando (bucle) en un string

```
sentencia = "todo tiene su final"
```

```
for aChar in sentencia:
```

```
    print(aChar)
```

```
x=3*sentencia
```

```
print(x)
```

Mayúsculas y minúsculas

```
saludo="Hola gente UNI"
```

```
x=saludo.upper()
```

```
print(x)
```

```
x=saludo.lower()
```

```
print(x)
```

```
x=saludo[3].upper()
```

```
print(x)
```


Ejercicio

- Hacer un programa que pida una frase y cuente cuantas vocales hay en esa frase
- Usar “replace” para cambiar “Hola” por “Chau”

Tamaño de una secuencia

```
MAX_FILE_LENGTH = 256
```

```
SUFFIX_LENGTH = 3
```

```
filename = input("Enter new file name (max 256 characters): ")
```

```
if (len(filename) > MAX_FILE_LENGTH):
```

```
    print("File name exceeded the max size of %d characters," %(MAX_FILE_LENGTH))
```

```
else:
```

```
    # Find file type, last three characters in string e.g., resume.txt
```

```
        endSuffix = len(filename)
```

```
        startSuffix = endSuffix - SUFFIX_LENGTH
```

```
        suffix = filename[startSuffix:endSuffix]
```

```
        if (suffix == "txt"):
```

```
            print("Text file")
```

```
            print("%d:%d %s" %(startSuffix,endSuffix,suffix))
```

Funciones varias

Función booleana	Descripción
<code>isalpha()</code>	V si son puras letras.
<code>isdigit()</code>	V si son puros dígitos
<code>isalnum()</code>	V si son letras y/o dígitos
<code>islower()</code>	V si las letras son minúsculas
<code>isspace()</code>	V si son puros espacios en blanco, tabuladores o saltos de línea
<code>isupper()</code>	V si las letras son mayúsculas.

Más funciones

Función	Descripción
<code>endswith</code> <code>(substring)</code>	V si el string termina en el substring
<code>startswith</code> <code>(substring)</code>	V si el strin empieza con el substring
<code>find</code> <code>(substring)</code>	La primera ocurrencia del substring en el string (o -1 si no lo encuentra)
<code>replace</code> <code>(oldstring,</code> <code>newstring)</code>	Reemplaza las apariciones de un viejo substring con un nuevo substring

Listas

- Semejante a los arreglos de C o de Java
- Aparte de len (como los strings), también tiene la función max y min

Ejercicio

- Hacer un programa que pida las notas de los alumnos de una clase. Luego debe imprimir todas las notas y el total de notas
- Sin listas:

```
SIZE = 5
```

```
stu1 = float(input("Enter grade for student no. 1: "))  
stu2 = float(input("Enter grade for student no. 2: "))  
stu3 = float(input("Enter grade for student no. 3: "))  
stu4 = float(input("Enter grade for student no. 4: "))  
stu5 = float(input("Enter grade for student no. 5: "))
```

Sin listas

```
total = stu1 + stu2 + stu3 + stu4 + stu5
```

```
average = total / CLASS_SIZE
```

```
print()
```

```
print("GRADES")
```

```
print("The average grade is %.2f%%", %average)
```

```
print("Student no. 1: %.2f", %stu1)
```

```
print("Student no. 2: %.2f", %stu2)
```

```
print("Student no. 3: %.2f", %stu3)
```

```
print("Student no. 4: %.2f", %stu4)
```

```
print("Student no. 5: %.2f", %stu5)
```

Creando una lista

Formato (lista de 'n' elementos):

```
<nombre_lista> = [<valor 1>, <valor 2>, ... <valor n>]
```

Ejemplo:

```
# Lista con 5 elementos
```

```
percentages = [50.0, 100.0, 78.5, 99.9, 65.1]
```

```
# otros ejemplos
```

```
letters = ['A', 'B', 'A']
```

```
names = ["Juan Carlos", "Luisa", "Ax'1", "Nadia"]
```


Accesando la lista

La lista completa

```
print(percentages)
```

```
>>> print(percentages)
[50.0, 100.0, 78.5, 99.9, 65.1]
```

Un elemento

```
print(percentages[1])
```

```
>>> print(percentages[1])
100.0
```

Indices negativos

- Python permite usar índices negativos (-1 es el último elemento, -2 el penúltimo, etc.)
- Pero mejor no usarlo, la gente que recién aprende Python pero ya sabe C o Java se puede confundir

Creando una lista sin tamaño fijo

- Paso 1: crear la variable

Formato:

```
<list name> = []
```

Ejemplo:

```
classGrades = []
```

Creando lista de tamaño no fijo

- Paso 2: inicializar la lista con elementos
- Usar función “append”

```
for i in range (0, 4, 1):
```

```
    # Each time through the loop: create new element = -1
```

```
    # Add new element to the end of the list
```

```
    classGrades.append(-1)
```

Programa de los alumnos con listas

```
def read(classGrades):  
    total = 0  
    average = 0  
    for i in range (0, CLASS_SIZE, 1):  
        temp = i + 1  
        print("Enter grade for student no.", temp, ":")  
        classGrades[i] = float(input(">"))  
        total = total + classGrades[i]  
    average = total / CLASS_SIZE  
    return(classGrades, average)
```

Continuación

```
def display(classGrades, average):  
    print()  
    print("GRADES")  
    print("The average grade is %.2f%%" %average)  
    for i in range (0, CLASS_SIZE, 1):  
        temp = i + 1  
        print("Student No. %d: %.2f%%"  
              %(temp,classGrades[i]))  
  
def start():  
    classGrades = initialize()  
    classGrades, average = read(classGrades)  
    display(classGrades,average)  
  
start()
```

Buscando / modificando

```
grades = ['A','B','C','A','B','C','A']  
last = len(grades) - 1  
i = 0  
while (i <= last):  
    if (grades[i] == 'A'):    # Search for matches  
        grades[i] = 'A+'    # Modify element  
  
    i = i + 1
```

Ojo

- Qué pasa?

```
list1 = [1,2]
```

```
list2 = [2,1]
```

```
print (list1, list2)
```

Two ref to one list

```
list1 = list2
```

```
print (list1, list2)
```

```
list1[0] = 99
```

```
print (list1, list2)
```


Y acá?

```
list1 = [1,2,3]
```

```
list2 = []
```

```
for i in range (0, 3, 1):
```

```
    list2.append(list1[i])
```

```
print(list1, list2)
```

```
list1[1] = 99
```

```
print(list1, list2)
```