Conceptos Avanzados de Programación Python

UNPSJB 2023

Trabajo Práctico 0. Operadores y Expresiones

Identificadores válidos en Python: Un identificador es el nombre empleado en un programa para identificar una variable, una función, etc. Cada lenguaje posee sus reglas para determinar si un identificador en válido. En Python los identificadores comienzan con una letra (de la A a la Z o de la a a la z) o con un guión bajo (_) seguido de cero o más letras, guiones bajos y números. Además, Python diferencia mayúsculas de minúsculas (*case sensitive*) y NO admite signos de puntuación ni símbolos como @, $ y % a excepción del guión bajo.

Tabla de operadores ordenados de mayor a menor prioridad [*].

Operador	Descripción
**	Exponenciación.
+x, -x	positivo, negativo
*, /, //, %	Multiplicación, división, división entera y resto.
+, -	Adición y sustracción.
in, not in, is, is not, <, <=, >, >=, !=, ==	Comparaciones, comprobaciones de membresía y de identidad.
not x	Booleano NOT.
and	Booleano AND.
or	Booleano OR.

1. Escriba un script en lenguaje Python para resolver las siguientes tareas. Tenga en cuenta las funciones que se listan en la columna a su derecha [Built-in].
   a) Muestre por pantalla la cadena *¡Hola Mundo!*.
   b) Almacene la cadena *¡Hola Mundo!* en una variable y luego muestre por pantalla el contenido de la variable.
   c) Pregunte el nombre del usuario en la consola y después de que el usuario lo introduzca muestre por pantalla la cadena *¡Hola* <nombre>*!*, donde <nombre> es el nombre que el usuario haya introducido.

2. Determine el resultado y el tipo de la variable `a` luego de la asignación.

   a) `a = 20 - 5`
   b) `a = 20 - 10.5`
   c) `a = 14/7`
   d) `a = 14//7`
   e) `a = -15//7`
   f) `a = -2**4`
   g) `a = 2**-1`
   h) `a = (50 - 5*6)/2`
   i) `a = 2**3**2`
   j) `a = 10 == 10.0`
   k) `a = 10 != 10.1`
   l) `a = not True`
   m) `a = 0 and True`
   n) `a = 10 > 15`
   ñ) `a = 1 - 0.2 - 0.2 - 0.2 - 0.2 - 0.2`

Documentación

El intérprete de Python tiene una serie de funciones y tipos incluidos en él que están siempre disponibles. Entre ellas:

- `print()`
- `input()`
- `type()`

Funciones Built-in

El capítulo 6 de la documentación explica el significado de los elementos de expresiones en Python.

Expresiones

Volver al índice...

3. Complete las tablas de verdad para las proposiciones dadas [Tablas de verdad].

A	B	C	A *or* B *and* C	A *and not*C
V	V	V		
V	V	F		
V	F	V		
V	F	F		
F	V	V		
F	V	F		
F	F	V		
F	F	F		

4. La desigualdad triangular o desigualdad de Minkowski establece que en todo triángulo, la suma de las longitudes de dos lados cualquiera es siempre mayor a la longitud del lado restante. Utilice esta desigualdad y los conceptos de triángulo isósceles, equilátero y escaleno para escribir expresiones que permitan evaluar si tres longitudes de segmento dadas (L1, L2 y L3) cumplen con las siguientes proposiciones:

   a) Es posible formar un triángulo.
   b) Es un triángulo isósceles.
   c) Es un triángulo equilátero.
   d) Es un triángulo escaleno.

5. Escriba expresiones aritmético-lógicas que permitan evaluar las siguientes propiedades en relación a un número entero `N`:

   a) Tiene dos dígitos.
   b) Pertenece al intervalo semiabierto [0,37).
   c) Tiene al menos tres dígitos.
   d) Es el predecesor del entero M.
   e) Es el sucesor o el predecesor del entero M.
   f) Es par.
   g) Es impar y positivo.
   h) Es divisor de M y múltiplo de P (donde M y P también son números enteros).
   i) No es múltiplo de 3 pero sí de 5.
   j) Es múltiplo de 3 y no de 5 o es múltiplo de 5 y no de 3.

Documentación

Las *tablas de verdad* sirven para determinar las condiciones de verdad de un enunciado. Es decir, su significado, en función de las condiciones de verdad de sus elementos atómicos.

A	B	A *and* B
V	V	V
V	F	F
F	V	F
F	F	F

A	B	A *or* B
V	V	V
V	F	V
F	V	V
F	F	F

A	*not* A
V	F
F	V

El operador lógico *Xor* u *or* exclusivo no está definido en el lenguaje. Sin embargo, es posible definirlo mediante el uso de otros operadores. Su tabla de verdad es:

A	B	A *xor* B
V	V	F
V	F	V
F	V	V
F	F	F

Ejemplo:
$xor = not\ (A == B)$.

Volver al índice...

Conceptos Avanzados de Programación Python

UNPSJB 2023

Trabajo Práctico 1. Bloques Secuenciales

Los siguientes identificadores se utilizan como *palabras reservadas*, o palabras clave del idioma, y no pueden utilizarse como identificadores. Recuerde que el leguaje es sensible a mayúsculas y minúsculas y, por lo tanto, deben escribirse tal cual fueron definidas. [Palabras clave]

Palabras reservadas del lenguaje:

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

Palabras reservadas suaves (versión 3.10 o superior): match case _

Imprimiendo con Formato:

Un *formatted string literal* o `f-string` es un literal de cadena que se prefija con 'f' o 'F'. Estas cadenas pueden contener campos de reemplazo, que son expresiones delimitadas por llaves {}. Mientras que otros literales de cadena siempre tienen un valor constante, las cadenas formateadas son expresiones evaluadas en tiempo de ejecución [f-string].

Ejemplo: `cadena = f"Un literal de string {idVariable}"`

En Python también es posible dar formato a una cadena empleando el método `format()` de la clase `str` [**?**, Pág.27]. La cadena de caracteres sobre la que se está ejecutando este método puede contener texto literal y también marcas de reemplazo definidas mediante llaves {}.

Ejemplo:
```
cadena = "Un literal de string {} bla bla {} bla"
cadena = cadena.format(idVariable, Expresión)
```

1. Sabiendo que una empresa posee 100 autos con capacidad para 4 personas cada uno, y que en el día de la fecha se encuentran disponibles 30 choferes, determine la siguiente información para una lista de espera de 50 pasajeros: La cantidad de autos disponibles para el traslado de pasajeros. ¿Cuántos pasajeros más se podrían sumar durante el día? ¿Cuál es la capacidad de pasajeros ociosa por falta de choferes?
2. Se desea implementar un simulador de plazo fijo. Realice un programa que pregunte al usuario una cantidad a invertir, el interés mensual, el período en meses a depositar y calcule el capital obtenido al finalizar la inversión.
3. Una juguetería tiene mucho éxito con la venta de baldes de ladrillitos en sus presentaciones de 350 y 500 gr. El correo los obliga a armar bultos de no más de 10 unidades cuyo costo final se calcula en función de su peso. Escriba un programa que lea el número de baldes de cada presentación de un pedido y determine en forma automática el costo total del envío sabiendo que cada bulto tiene un costo fijo de 350$ más un proporcional de 125$ por kilo.
4. Dado un entero positivo de cinco cifras `n`, escriba el bloque de sentencias necesarias para determinar si es capicúa. ¿Qué sucedería si no conoce la cantidad de cifras de "`n`"?

Documentación

El intérprete de Python tiene una serie de funciones y tipos incluidos en él que están siempre disponibles.

- `bool()`
- `int()`
- `float()`
- `complex()`
- `str()`

Funciones Built-in

En un literal de string, la barra invertida (\) se utiliza para escapar de los caracteres que de otra manera tienen un significado especial.

\"	Comillas dobles
\'	Comilla simple
\\	Barra invertida
\n	Enter
\t	Tab

Volver al índice...

Conceptos Avanzados de Programación Python

UNPSJB 2023

Trabajo Práctico 2. Ciclo Definido

Los tipos *inmutables* del lenguaje son los tipos de datos: numéricos, booleanos, cadenas de caracteres y Tuplas. Como su propio nombre lo indica, un objeto inmutable una vez creado nunca cambia su valor. Si bien en apariencia es posible modificar la información de una variable string o numérica, en realidad durante la asignación estamos cambiando de objeto. [Inmutable]
Para mayor claridad, se sugiere leer el tema *"The Reference Types and Assignment Statements"* del libro [**?**, Pág.20]. [Jerarquía de tipos estándar]

1. Imprima en pantalla la siguiente serie de enteros:
   a) 1, 4, 9, 16....10000
   b) 25, 16, 9, 4, 1, 0, 1, 4, 9, 16, 25
   c) 100, 95, 90, ..., 0
2. Dado `n` un entero positivo, determine el resultado:
   a) $\sum_{i=0}^{n} \frac{(-1)^i}{1+i}$
   b) $\sum_{i=1}^{n} i^{-i}$
3. Escriba un algoritmo para calcular el valor aproximado de e^x empleando la serie de Maclaurin.

$$e^x = \sum_{i=0}^{\text{ínf}} \frac{x^i}{i!}$$

4. Implemente un algoritmo que solicite al usuario un entero positivo `n` mayor que dos y calcule en enésimo término de la Sucesión de Fibonacci,
5. Escribir un script que pida al usuario un número entero y muestre por pantalla un pino de asteriscos cuya altura es el dato introducido. Ejemplo con altura = 3:

```
  *
 ***
*****
```

6. Escribir un script que pida al usuario un dígito y muestre por pantalla un triángulo rectángulo de números cuya altura es el dato introducido. Ejemplo con altura = 4:

```
1
31
531
7531
```

7. Escriba un bloque de sentencias que permita generar todos los números de la forma `ABAB` con $1 \leq A \leq 9$ y $9 \geq B > 0$.

Documentación

La sentencia `for` se usa para iterar sobre los elementos de una secuencia (como strings, tuplas, etc) u otro objeto iterable. El número de iteraciones se conoce antes de ejecutar la instrucción.

```
for<x>in<secuencia>:
  Bloque A
```

El Bloque A de tareas se ejecuta una vez para cada elemento `x` proporcionado por la `<secuencia>`, en el orden establecido en ella.

La función `range()` produce una secuencia de números enteros.

`range(tope)`: secuencia en el rango entre 0 y tope-1 (incremento = 1, `tope>0`)

`range(inicio, tope)`: secuencia en el rango entre inicio y tope-1 (incremento = 1, `inicio<tope`)

`range(inicio, tope, paso)`: secuencia en el rango entre inicio y tope-1 con incremento igual al paso.
Si `paso>0`: `inicio<tope`
Si `paso<0`: `inicio>tope`

Volver al índice...

Conceptos Avanzados de Programación Python

UNPSJB 2023

Trabajo Práctico 3. Condicionales y Tuplas

Las tuplas son secuencias inmutables, usadas normalmente para almacenar colecciones de datos heterogéneos [tuple]. Las tuplas se pueden construir de diferentes maneras:

- Vacía: Usando un par de paréntesis `()` o el constructor `tuple()`
- Único elemento: Usando una coma al final `a,` o `(a,)`
- General: Separando los valores con comas `a,b,c` o `(a,b,c)`
- Con iterables conocidos: `tuple('hola')` o `tuple(range(3))`

1. Implemente un script que lea desde la consola el valor del entero `N` y verifique si cumple con las condiciones del ejercicio 5 del Trabajo Práctico 0.
2. Empleando dos dígitos distintos, `A` y `B`, escriba un script para inicializar al entero `M` siguiendo las siguientes reglas:
   a) El mayor número entero de dos cifras.
   b) El menor número de 4 dígitos intercalando los mismos.
   c) El menor número capicúa de tres cifras.
   d) El mayor número capicúa de 5 dígitos.
3. Implemente nuevamente el ejercicio 4 del Trabajo Práctico 0 empleando condicionales.
4. Determine el cuadrante al cuál pertenece un punto en el plano representado por una tupla de reales de la forma `(x,y)`. ¿Qué resultado se obtendría si el punto se encuentra sobre alguno de los ejes cartesianos o ambos?
5. Implemente un script que a partir de una tupla de la forma (día,mes,año) determine si corresponde a una fecha válida. Recuerde que los meses de abril, junio, septiembre y noviembre tienen 30 días; el mes de febrero puede tener 29 o 28 días dependiendo de si se trata de un año bisiesto o no; y los meses restantes poseen 31 días. Nota: Un año es bisiesto si verifica las siguientes condiciones: es divisible por 4 pero no por 100 o, es múltiplo de 400.
6. Dada la tupla `t=(True,False)`, escriba un bloque de código que le permita verificar los resultados obtenidos en el ejercicio 3 del Trabajo Práctico 0.
7. Dada una tupla de enteros `(-2,14,5,-7,25,18,30)`
   a) Determine la cantidad de enteros positivos.
   b) El mínimo valor.

Documentación

La sentencia `if` se usa para la ejecución condicional de código y su sintaxis es:

```
if <condición>:
  Bloque A
[else:
  Bloque B]
```

Es decir, se ejecuta el bloque A de tareas solo si la condición de cumple, caso contario, se ejecuta el bloque B. Recuerde que la cláusula `else` es opcional.

Además del método `format()`, los strings poseen otros métodos muy útiles:

- `upper()`
- `lower()`
- `split()`
- `replace()`

En el siguiente link se presenta la lista completa de métodos con su explicación y ejemplos:
[Métodos para strings]

El `string` es un tipo *inmutable*. Sus métodos retornan valores nuevos. Es decir, **no** modifican el string original.

Volver al índice...

Conceptos Avanzados de Programación Python

UNPSJB 2023

Trabajo Práctico 4. Bucles Condicionados

En muchas aplicaciones es necesario ser capaces de almacenar en memoria, un grupo de datos no necesariamente heterogéneos denominados secuencias. Las *secuencias* son contenedores donde los elementos se almacenan siguiendo un orden. Previamente, se han introducido los tipos simples (booleanos, enteros, reales, etc) que conceptualmente poseen un único valor asociado y, las secuencias de datos inmutables `range`, `strings` y `tuples` [Secuencia].
Secuencias inmutables:

- Strings: Secuencia de caracteres.
- Rangos: Secuencia de enteros.
- Tuplas: Secuencia heterogénea de datos inmutables.

1. Implemente nuevamente el ejercicio 1 del Trabajo Práctico 2 empleando la estructura de control `while`.
2. Determine el valor aproximado de e^x con un error menor a un valor ingresado por el usuario (rever ejercicio 3 del Trabajo Práctico 2). Imprima en pantalla la cantidad de términos necesarios para alcanzar la precisión deseada.
3. Escriba bloques de sentencias en un script que a partir de un entero positivo `n` ingresado por el usuario calcule:
   a) Sus dígitos más y menos significativos.
   b) La cantidad de dígitos del número.
   c) La cantidad de dígitos pares e impares.
   d) Un entero con sus dígitos en orden inverso.
   e) Un entero solo con los dígitos pares.
4. Implemente un script para determinar:
   a) Si un número entero positivo es primo.
   b) El máximo divisor entero de un número.
5. Implemente un script para determinar si:
   a) Un número entero positivo es capicúa.
   b) Una cadena de caracteres cumple la condición de ser un palíndromo.
6. Dada una tupla con valores enteros, determine:
   a) Si los valores se encuentran ordenados de mayor a menor.
   b) Si un elemento `n` ingresado por el usuario pertenece a la secuencia.
   c) Si contiene los primeros términos de la serie de fibonacci.

Nota: En los incisos cuya información de entrada es un valor entero complete la tarea trabajando *siempre* con datos numéricos.

Documentación

El intérprete de Python tiene una serie de funciones y tipos incluidos en él que están siempre disponibles. Entre ellas:

- `len()`
- `min()`
- `max()`
- `any()`
- `all()`

Funciones Built-in

Operadores como el $+$ o el $*$ poseen otra función si se los emplea con secuencias. Consulte las operaciones válidas, su funcionamiento y su prioridad en la siguiente [Tabla]

La sentencia `while` se usa para la ejecución repetida de una tarea siempre que la condición sea verdadera:

```
while <condición>:
  Bloque A
```

Es decir, se ejecuta el bloque A de tareas tantas veces mientras la <condición>sea verdadera.

Volver al índice...

Conceptos Avanzados de Programación Python

UNPSJB 2023

Trabajo Práctico 5. Funciones

Un método para resolver un problema complejo consiste en subdividirlo en subproblemas más sencillos que, a su vez, pueden resolverse aplicando la misma metodología. Siguiendo esta estrategia, la resolución comienza con una descomposición modular y luego nuevas descomposiciones de cada módulo en un proceso de refinamiento sucesivo o *stepwise*. Esta técnica se denomina diseño descendente o *top-down* debido a que se comienza en la parte superior con el problema general y se diseñan soluciones específicas para cada subproblema. Una forma de modularizar el código es implementando *funciones* para resolver tareas específicas. [Definición de funciones]

En el momento de la definición de una función es posible declarar *parámetros* como información de entrada a la subrutina. Los identificadores de los parámetros que se utilizan en el momento de la declaración se denominan *parámetros formales* y son los identificadores empleados dentro del cuerpo de ejecución del módulo. Luego, durante la invocación del subprograma los datos literales, las expresiones o las variables con los cuales son llamados se denominan *argumentos o, parámetros actuales, reales o efectivos*. [Parámetros versus Argumentos]

Para mayor claridad, se sugiere leer los temas del capítulo *"Functions in Python"* del libro [**?**, Sección 11.5, pág.119 y Sección 11.8, pág.124; respectivamente].

1. Realice una función para convertir un valor en kilómetros por hora $[km/h]$ a metros por segundo $[m/s]$ y otra con la operación inversa que convierta de $[m/s]$ a $[km/h]$.
2. Implemente funciones para modelar las operaciones lógicas `xor`, `nand` y `nor`. Consulte las tablas de verdad de cada una de ellas en: [Operaciones lógicas]
3. Transforme los bloques de sentencias pedidos en los ejercicios 3 y 6 del Trabajo Práctico 4 en funciones.
4. Implemente el ejercicio 3 del Trabajo Práctico 3 utilizando las funciones `esTriangulo` y `tipoDeTriangulo`. Esta última deberá invocarse solo si las longitudes de los segmentos ingresados por el usuario conforman un triángulo.
5. Algunos números naturales de n dígitos de la forma $N = dj, \ldots, d1, d0$ con $di \neq 0 \forall i \in [0, j]$, cumplen con una propiedad de divisibilidad como se describe a continuación. Sea N=362 verifica la propiedad dado que 362 es divisible por 2, 36 es divisible por 6 y 3 es divisible por 3 mientras que N=168 no la verifica. Implemente una función para determinar si un número entero positivo cumple con esta propiedad.
6. Escriba una función que dada la altura y un caracter dibuje un pinito de dicho caracter siguiendo las especificaciones del ejercicio 5 del Trabajo Práctico 2. En caso de que el usuario no especifique un caracter al momento de la invocación utilice $\star$ por defecto.
7. Implemente una función `fecha_valida`. Rever el ejercicio 5 del Trabajo Práctico 3.

Documentación

Sintaxis para definir una función:

```
def idFuncion(*):
    [return]
```

- Una función puede recibir o no parámetros (*).
- La sentencia **return** es opcional y permite que la función retorne al menos un resultado.

[Asignación aumentada]

Operador	Descripción
x += a	x = x+a
x -= a	x = x-a
x *= a	x = x*a
x /= a	x = x/a
x %= a	x = x %a
x //= a	x = x//a
x **= a	x = x**a

Volver al índice...

8. Escriba una función que realice la operación `CamelCase` de una cadena de caracteres. Su función deberá retornar una nueva cadena de caracteres sin espacios y con las palabras de la cadena original formateada con la primera letra en mayúscula y todas las restantes en minúscula. Ejemplo: Entrada = `"hola mundO"` Salida = `"HolaMundo"`
9. Realice funciones apropiadas para realizar cada una de las siguientes consultas a partir de una tupla de fechas de `nacimientos` que contiene las fechas de nacimiento de un grupo de personas con el formato `(dia, mes, año)`. Ayuda: Es posible identificar unívocamente a una fecha a partir de un número entero de la forma Año*10000+ Mes*100+ Día.
   a) ¿Existe algún mes sin cumpleaños? De no existir retorne **`None`**.
   b) Todos los meses sin cumpleaños.
   c) El promedio de edad del grupo.
   d) El último y el primer cumpleaños del año.
10. Dada una tupla con información de círculos en el plano, implemetente funciones para realizar las siguientes consultas de interés:
    a) Determinar cuál es el círculo cuyo centro se encuentra más cercano a un `(x,y)` dado y a qué cuadrante pertenece.
    b) Determinar cuántos círculos dentro de la estructura se encuentran centrados sobre los ejes de coordenadas.

    `circulos` $= (c_1, c_2, \ldots, c_n)$, donde cada $c_i = (r_i, (x_i, y_i))$ es otra tupla con la magnitud del radio y las coordenadas del centro del círculo en el plano XY.
11. Analice los siguientes bloques de código, verifique la salida en pantalla y determine a qué variable `a` hace referencia en cada caso:

a)
```
def subrutina():
    a = 2
    print(a)
a = 5
subrutina()
print(a)
```

b)
```
def subrutina():
    print(a)
subrutina()
a = 5
print(a)
```

Documentación

[Parámetros]
En el momento de la declaración es posible asignar valores por *defecto o predeterminados* a los parámetros de las funciones. Los parámetros por defecto se declaran luego de los parámetros fijos. En consecuencia, la función podrá recibir menor número de argumentos. Los valores de los parámetros predeterminados se evalúan de izquierda a derecha cuando se ejecuta la definición de la función.

[Cardinalidad]
Si una función espera recibir un número *arbitrario* (desconocido) de argumentos, estos argumentos llegarán en forma de tupla. Para definir un argumento de longitud arbitraria en una función, se antecede al parámetro formal con un asterisco (*). También es posible obtener parámetros arbitrarios como pares del tipo *clave=valor*. En estos casos, los argumentos llegan en forma de diccionario y, al nombre del parámetro deben precederlo dos astericos (**).

[Argumentos]
El lenguaje permite la invocación de subrutinas empleando dos tipos de argumentos, argumentos por *posición* o por *nombre*.

Volver al índice...

Conceptos Avanzados de Programación Python

UNPSJB 2023

Trabajo Práctico 6. Listas y Dicccionarios

Las listas y los diccionarios son estructuras de datos nativas del lenguaje con la capacidad de ser heterogéneas y *mutables*. Es decir, admiten almacenar elementos de distinto tipo y una vez creadas su contenido se puede modificar sin cambiar de objeto. [Estructuras de Datos].
Una propiedad sorprendente de estas estructuras es su capacidad de crear nuevas listas o diccionarios basados en listas o diccionarios pre-existentes de forma inteligente, intuitiva y concisa en una única sentencia. Consultar: [Comprensión Listas] y [Diccionarios].

Importando módulos:
Un módulo es un archivo .py que alberga un conjunto de funciones, variables o clases que pueden ser usados por otros módulos. Importar un módulo requiere de la palabra reservada `import` y su sintaxis es:

```
import <idModulo>
```

Para mayor claridad se sugiere leer el tema *"Importing Python Modules"* del libro [**?**, Sección 25.4, pág.284].

1. Implemente funciones que retornen las siguientes listas de enteros:
   a) Los primeros `n` términos de la sucesión de Fibonacci.
   b) Todos los números primos comprendidos entre 1 y un valor `n`.
   c) Todos los divisores de `n` comprendidos en el intervalo cerrado $[1, n]$.
2. Implemente funciones que retornen listas de enteros de `n` elementos de la forma:
   a) Valores aleatorios entre $[0, 100]$.
   b) Valores aleatorios entre $[-x, x]$.
   c) Los primeros `n` números impares ordenados de mayor a menor.
3. Implemente funciones que dada una lista de la forma del ejercicio 2.b, determine o retorne:
   a) Si todos sus elementos son impares.
   b) Si los valores negativos de la lista solo se encuentran en posiciones pares.
   c) Una nueva lista que contenga sólo los valores positivos ordenados de menor a mayor.
   d) Si el contenido de la lista es monótono creciente.
4. Dada una lista de números enteros con repeticiones ordenado en forma descendente, se desea generar otra lista con pares (número, frecuencia) con la información de la frecuencia de aparición de cada uno. Por ejemplo, si `l` tiene los números [9,7,7,6,5,3,3,3,1,1,1] la lista `frecuencia` debe contener los pares [(9,1),(7,2) ,(6,1) ,(5,1) ,(3,3) ,(1,3)].

Documentación

El intérprete de Python tiene una serie de funciones y tipos incluidos en él que están siempre disponibles. Entre ellas:

- `list()`
- `dict()`
- `sorted()`
- `zip()`

Funciones Built-in

El módulo `random` implementa generadores de números pseudoaleatorios para varias distribuciones. En particular, para números enteros existe una selección uniforme dentro de un rango. [Random]

Volver al índice...

5. Realice funciones apropiadas para realizar cada una de las siguientes operaciones a partir de una lista de fechas `nacimientos` que contiene las fechas de nacimiento de un grupo de personas (dia, mes, año).
   a) Extraer de la lista el cumpleaños de la persona más joven.
   b) Ordenar la lista por orden cronológico. Implemente su propio algoritmo de ordenamiento.
6. Implemente los siguientes diccionarios que a cada clave entera le asocie los siguiente valores:
   a) El máximo divisor entero distinto de si mismo.
   b) La lista completa de divisores del número.
7. Dado un diccionario del tipo (dato: máximo divisor entero distinto de si mismo) construya un nuevo diccionario que no contenga a los números primos.
8. Construya un diccionario a partir de la información contenida en la tupla del ejercicio del ejercicio 9 del Trabajo Práctico 5. Su diccionario será de la forma `nacimientos`={'enero': [(dia, año),..], 'febrero':....}. Vuelva a implementar las funciones teniendo en cuenta la nueva estructura.
9. Se desea almacenar la ruta de viaje de vehículos no tripulados que realizan una travesía de postas definidas a partir de las coordenadas cartesianas (x,y) expresadas en metros. Considerando que el desplazamiento entre postas sucesivas es lineal, defina un diccionario que le permita almacenar la información e implemente funciones apropiadas para identificar cuál de los vehículos:
   a) Recorrió la mayor distancia.
   b) Abarcó la menor superficie rectangular.
10. A partir de los siguientes diccionarios:
    `continentes`={'Africa': ['Angola',...,'Egipto',...], 'America':['Argentina', 'Canada', ...], 'Asia': [...], 'Europa': [...], 'Oceania': [...]}
    Continente es la *clave* del diccionario y su *valor* es una lista con los paises que lo integran.
    `Paises`={..., 'Argentina':[3761274, 40.11, 48,['Bolivia', 'Brasil', 'Chile', 'Paraguay', 'Uruguay']],...}
    Pais es la *clave* del diccionario y su *valor* es una lista que contiene: la superficie en km^2, la cantidad de habitantes en millones, la superficie cubierta por agua en porcentaje y la lista de sus países limítrofes.

    Implemente funciones para realizar las siguientes consultas de interés.
    a) Determine la lista de países con un único país limítrofe independientemente del continente al que pertenezca.
    b) ¿Cuál es el país con menor densidad poblacional de cada continente?
    c) ¿Cúal es el continente con más países con mayor porcentaje de agua que de tierra?

Documentación

Métodos de las listas:

- `append()`
- `count()`
- `index()`
- `insert()`
- `remove()`

Encontrará su explicación y ejemplos en el link: [Métodos de Listas]

Métodos de los diccionarios:

- `get()`
- `items()`
- `keys()`
- `popitem()`
- `update()`
- `values()`

Encontrará su explicación y ejemplos en el link: [Métodos de Diccionarios]

Métodos compartidos:

- `pop()`
- `clear()`
- `copy()`

Volver al índice...

Conceptos Avanzados de Programación Python

UNPSJB 2023

Trabajo Práctico 7. Archivos

Python nos permite trabajar con el sistema de archivos y directorios brindando una opción de entrada y salida de información de un programa. Un método simple es trabajar con *archivos manipulando* su lectura y escritura a nivel de la aplicación. Los archivos en Python son objetos de tipo `file` creados mediante la función `open()`. Esta función toma como parámetros una cadena con el *path* o ruta al archivo a abrir, que puede ser relativa o absoluta, una cadena opcional indicando el modo de acceso, entre otros parámetros opcionales [open()].
Ejemplo:

```
archivo = open('miarchivo.txt', mode ='x', encoding ='utf-8')
```

Consultar [Text I/O] y, para mayor claridad, se sugiere leer el tema en el capítulo 18 *"Reading and Writing Files"* del libro [**?**, pág.215].

1. Implemente un script que a partir de un archivo de texto con números enteros positivos genere una archivo de salida con la siguiente información:

   a) Cada uno de los enteros del archivo de entrada junto a su máximo divisor entero distinto de si mismo.
   b) Cada uno de los enteros del archivo de entrada junto a la lista completa de divisores del número.

   Recuerde que en el ejercicio 6 del Trabajo Práctico 6 implementó funciones apropiadas para ambas tareas.

2. Dada una lista de enteros positivos ordenada con repeticiones (Ver el ejercicio 4 del Trabajo Práctico 6, implemente una función que guarde en un archivo la información de los pares dato-frecuencia.

3. Implemente un script que apartir de un archivo de texto con fechas de la forma dia/mes/año $31/10/2013\backslash n$ $31/11/2002\backslash n$ $29/2/2019\backslash n$... realice las siguientes operaciones:

   a) Cargue la lista `nacimientos` solo con las fechas válidas incluidas en el archivo.

   Los meses de abril, junio, septiembre y noviembre tienen 30 días; el mes de febrero puede tener 29 o 28 días dependiendo de si se trata de un año bisiesto o no; y los meses restantes poseen 31 días. Recuerde que un año es bisiesto si cumple con la condición de ser divisible por 4 pero no por 100 o, es múltiplo de 400.
   b) Ordene la lista `nacimientos` con las fechas ordenadas cronológicamente.
   c) Guarde la lista ordenada en un archivo de texto.

Documentación

El intérprete de Python tiene una serie de funciones y tipos incluidos en él que están siempre disponibles. Entre ellas:

- `open()`

Funciones Built-in

Parámetro `mode`: es una cadena de caracteres opcional que especifica el *modo* en el cual el archivo es abierto. Por defecto, su valor es `'rt'`.

- `'r'`: abierto para lectura (por defecto)
- `'w'`: abierto para escritura
- `'x'`: crea el archivo y falla si existe otro con el mismo nombre
- `'a'`: abierto para escritura, añadiendo al final del archivo si este fue creado previamente
- `'b'`: modo binario
- `'t'`: modo texto (por defecto)
- `'+'`: abierto para su actualización

Volver al índice...

Trabajo Práctico 8. Conjuntos

En Python un objeto del tipo *conjunto* o *set* es una colección no ordenada de objetos diferentes entre sí. En la actualidad hay dos tipos básicos de conjuntos: *set* y *frozenset*. La clase set es mutable, es decir, el contenido del conjunto puede ser modificado a través de sus métodos. En cambio, la clase frozenset es inmutable y, por tanto, sus datos pueden ser utilizados como claves de diccionario o como elementos de otro conjunto.
Consultar [Sets] y, para mayor claridad, se sugiere leer el tema en el capítulo 32 *"Sets"* del libro [**?**, pág.379].

1. Dados dos sets con nombres de estudiantes, uno con los anotados para rendir el siguiente examen y el otro con los inscriptos para rendir el proyecto final, implemente funciones para responder las siguientes consultas:
   a) ¿Cuáles son los estudiantes anotados en el examen y en el proyecto final?
   b) ¿Cuáles son los estudiantes que solo se presentan a una de las instancias de evaluación?
   c) ¿Cuántos son los estudiantes anotados en total?
   d) ¿Cuáles son los estudiantes que solo presentan el proyecto final?
2. Escriba funciones para retornar el conjunto de:
   a) Palabras que aparecen en dos archivos de texto distintos.
   b) Todas las posiciones en las que un caracter dado aparece en una cadena.
   c) Las letras que aparecen en todas las líneas de un archivo de texto.
3. La Criba de Eratóstenes es un algoritmo que permite hallar todos los números primos menores que un número natural dado. A partir del conjunto de números naturales comprendidos entre 2 y n escriba una función que retorne el conjunto de números primos menores que n empleando dicho algoritmo.
4. Se desea representar un mapa con algunas de las principales ciudades del país y la comunicación entre las mismas. A partir de una lista de ciudades y un diccionario con una ciudad como clave y un conjunto de cuidades aledañas a ella como valor, implemente una función para determinar todas las ciudades que se pueden alcanzar desde una ciudad dada.

Documentación

El intérprete de Python tiene una serie de funciones y tipos incluidos en él que están siempre disponibles. Entre ellas:

- `set()`

Funciones Built-in

Los conjuntos poseen métodos muy útiles:

- `add()`
- `remove()`
- `clear()`
- `pop()`
- `update()`

Y con operaciones matemáticas de conjuntos típicas a disposición:

- `difference()`
- `intersection()`
- `union()`

En el siguiente link se presenta la lista completa de métodos con su explicación y ejemplos:

[Métodos de Conjuntos]

Volver al índice...

Trabajo Práctico 9. Recursión

La *recursión* o recurrencia, es una manera de especificar una definición basándose en si misma. En particular, en programación se dice que una función es recursiva si se invoca a sí misma. En otras palabras, el problema se resuelve mediante recursión cuando la solución depende de los resultados de instancias más pequeñas de sí mismo hasta alcanzar el caso base.

Consultar el tema en el capítulo 9 *"Recursion"* del libro [**?**, pág.99].

1. Escriba una función recursiva y otra iterativa para obtener el n-ésimo elemento de la sucesión de Fibonacci. Compare ambas implementaciones de acuerdo a la utilización de memoria, el tiempo de ejecución y la claridad del código en relación a la definición del problema.
2. Dado un número natural `n`, escriba funciones recursivas para:
   a) Imprimir en pantalla su cuenta regresiva.
   b) Sumar todos sus dígitos.
   c) Obtener su dígito más significativo.
   d) Determinar si todos sus dígitos están ordenados de menor a mayor.
   e) La cantidad de dígitos pares e impares que contiene.
   f) La cantidad de veces que se repite su dígito más significativo.
3. Escriba una función recursiva que implemente la búsqueda binaria.
4. Dada una cadena de caracteres, escriba funciones recursivas para:
   a) Determinar si es un palíndromo.
   b) Contar la cantidad de apariciones de una letra dada.
   c) Retornar la lista con las posiciones en las que aparece la letra.
   d) Retornar el conjunto de enteros con las posiciones en las que aparece la letra.
5. Realice una función recursiva para verificar si una lista de N elementos enteros esta ordenada de mayor a menor.
6. Implemente una función recursiva para el ejercicio 5 del Trabajo Práctico 5

Documentación

Estructura usual de una función recursiva:

```
def idfuncion(*):
  if <caso base>:
    Bloque A
  else:
    Bloque B
```

- `(*)` Parámetros formales de la función.
- `Bloque A`

Tarea de resolución directa que retorna un valor que depende sólo de la instancia actual.

- `Bloque B`

Se resuelve la instancia previa a partir de la cual se obtiene el resultado para la instancia actual.

Volver al índice...