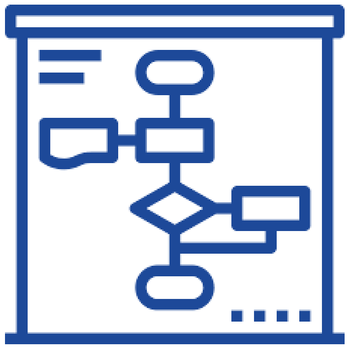
INTRODUCCIÓN A LA PROGRAMACIÓN



OTOÑO, 2022

UNIVERSIDAD TECNOLÓGICA DE CHILE
INSTITUTO PROFESIONAL
CENTRO DE FORMACIÓN TÉCNICA
nacap

UNIDAD III

Arreglos, tuplas y diccionarios

(Tuplas)

Conceptos Generales

```
tuple1=("abc", 34, True, 40, "bien")
```

Tuplas

- Una tupla es una secuencia de valores agrupados, como si fueran un único valor, que, por su naturaleza, deben ir juntos.
- El tipo de datos que representa a las tuplas se llama tuple. El tipo tuple es inmutable: una tupla no puede ser modificada una vez que ha sido creada.
- Desde la perspectiva de Python, las tuplas se definen como objetos con el tipo de datos 'tupla': <class 'tuple'>.
- Una tupla puede ser creada poniendo los valores separados por comas y entre paréntesis. Por ejemplo, podemos crear una tupla que tenga el nombre y el apellido de una persona:

```
persona=('Pancho','Javier')
print(persona)
#salida ('Pancho', 'Javier')
```

Elementos de las tuplas

Los elementos de tupla están ordenados, no se pueden cambiar y permiten valores duplicados. Los elementos de tupla están indexados, el primer elemento tiene índice [0], el segundo elemento tiene índice [1], etc. Nota: Considerar también el operador de rebanado.

```
persona=('Pancho','Javier')
print(persona[1])
#salida Javier
```

Nota: A diferencia de las listas, los elementos no se pueden modificar:
`persona[1]='Pablo'`

TypeError: el objeto 'tuple' no admite la asignación de elementos

El constructor tuple()

También es posible usar el constructor **`tuple()`** para hacer una tupla.

```
frutas = tuple(("manzana", "plátano", "frambuesa"))
print(frutas)
# tenga en cuenta los corchetes redondos dobles
```

Rangos de índices

Puede especificar un rango de índices indicando dónde comenzar y dónde terminar (subconjunto).
Al especificar un rango, el valor de retorno será una nueva tupla con los elementos especificados.

Ejemplo indexación negativa

```
frutas = (“manzana", “plátano", “frambuesa")
print(frutas[-1])
#-1 se refiere al último elemento, -2 se refiere
al penúltimo elemento, etc.
```

```
frutas = (“manzana", “plátano", “frambuesa", “naranja", "kiwi", "melón", "mango")
print(frutas[2:5])
#salida ('frambuesa', 'naranja', 'kiwi')
```

Desempaquetado de tuplas

Los valores individuales de una tupla pueden ser recuperados asignando la tupla a las variables respectivas.
Esto se llama desempaquetar la tupla (en inglés: unpack):

```
persona=('Pancho','Javier’)
nombre1,nombre2=persona
print(nombre1)
#salida Pancho
```

Nota: Si se intenta desempaquetar una cantidad incorrecta de valores, ocurre un error de valor:

```
nombre1,nombre2,nombre3=persona
ValueError: no hay suficientes valores para descomprimir (esperado 3, obtuve 2)
```

Comprobando la existencia de un elemento

Para determinar si un elemento específico está presente en una tupla, use la palabra clave **in**:

```
frutas = ("manzana", "plátano", "frambuesa")
if "manzana" in frutas:
  print("Sí, 'manzana' está en la tupla de frutas")
```

¿Cambiar valores de una tupla?

Una vez que se crea una tupla, no puede cambiar sus valores.
Recordar que las tuplas son inmutables, pero hay una solución: puede convertir la tupla en una lista, cambiar la lista y convertir la lista nuevamente en una tupla.

```
x = ("manzana", "plátano", "frambuesa")
y = list(x)
y[1] = "kiwi"
x = tuple(y)
print(x)
#salida ('manzana', 'kiwi', 'frambuesa')
```

Bucles de tuplas

- Puede recorrer los elementos de la tupla utilizando un bucle **for**:

```
frutas = ("manzana", "plátano", "frambuesa")
for x in frutas:
  print(x)
```

- Recorrer los números de índice.

  Utilice las funciones **range()** y **len()** para crear un iterable adecuado.

```
frutas = ("manzana", "plátano", "frambuesa")
for i in range(len(frutas)):
    print(frutas[i])
```

- Usando bucle **while**:

```
frutas = ("manzana", "plátano", "frambuesa")
i=0
while i< len(frutas):
    print(frutas[i])
    i=i+1
```

```
#para todos los casos la salida es:
manzana
plátano
frambuesa
```

Operaciones con tuplas

Agregar elementos

Convertir en una lista: al igual que la solución alternativa para cambiar una tupla, puede convertirla en una lista, agregar sus elementos y convertirlos nuevamente en una tupla. Lo anterior puede realizarlo con el método **append()** o **insert()**.

```
frutas = ("manzana", "plátano", "frambuesa")
y = list(frutas)
y.insert(2,"naranja")
frutas = tuple(y)
print(frutas)
#imprime ('manzana', 'plátano', 'naranja', 'frambuesa')
```

Eliminar elementos

Nota: no puede eliminar elementos

Las tuplas no se pueden cambiar, por lo que no puede eliminar elementos de ellas, pero puede usar la misma solución alternativa que usamos para cambiar y agregar elementos de tupla. Lo anterior puede realizarlo con el método **remove()** o **pop()**.

```
frutas = ("manzana", "plátano", "frambuesa")
y = list(frutas)
y.remove("manzana")
frutas=tuple(y)
print(frutas)
#imprime ('plátano', 'frambuesa')
```

Nota: La palabra clave **del** permite eliminar la tupla por completo:

```
del frutas
```

EJERCICIOS: Usos típicos de tuplas

- *Las tuplas se usan siempre que es necesario agrupar valores. Generalmente, conceptos del mundo real son representados como tuplas que agrupan información sobre ellos. Por ejemplo, un partido de fútbol se puede representar como una tupla de los equipos que lo juegan:*

```
partido1= (“Chile”,“Brasil”)
```

- *Para representar puntos en el plano, se puede usar tuplas de dos elementos (x, y).*
  Por ejemplo, podemos crear una función distancia que recibe dos puntos y entrega la distancia entre ellos:

```
def distancia(p1, p2):
    x1, y1 = p1
    x2, y2 = p2
    dx = x2 - x1
    dy = y2 - y1
    return (dx ** 2 + dy ** 2) ** 0.5
```

Al llamar a la función, se le debe pasar dos tuplas:

```
a = (2, 3)
b = (7, 15)
print(distancia(a, b))
```

- *Las fechas generalmente se representan como tuplas agrupando el año, el mes y el día. La ventaja de hacerlo en este orden (el año primero) es que las operaciones relacionales permiten saber en qué orden ocurrieron las fechas:*

```
hoy = (2011, 4, 19)
ayer = (2011, 4, 18)
navidad = (2011, 12, 25)
anno_nuevo = (2012, 1, 1)
f=hoy < ayer
g=hoy < navidad < anno_nuevo
print(f)
print(g)
```

EJERCICIOS: Usos típicos de tuplas

Una tupla puede contener otras tuplas.
Por ejemplo, una persona puede ser descrita por su nombre, su rut y su fecha de nacimiento:

```
persona = ('Pancho', '12345678-9', (1990, 5, 14))
nombre, rut, (a, m, d) = persona
print(m)
```

Una tabla de datos generalmente se representa como una lista de tuplas.
Por ejemplo, la información de los alumnos que están tomando un ramo puede ser representada así:

```
alumnos = [
    ('Pancho', 'Valparaíso', '201199001-5', 'Civil'),
    ('Javier', 'Viña del Mar', '201199002-6', 'Eléctrica'),
    ('Mónica', 'Santiago', '201199003-7', 'Enfermería'),
]
```

En este caso, se puede desempaquetar los valores automáticamente al recorrer la lista en un ciclo for:

```
for nombre, apellido, rol, carrera in alumnos:
    print(nombre, 'estudia', carrera)
```

inacap.cl