TALENTOFUTURO

Docentes:
Pablo Cruz
Roberto Arce

Clase: Introducción a la Programación Avanzada en Python

Módulo 2: Programación Avanzada en Python.

Apoyado por:
CORFO

Clase de hoy

01

Repaso

Tipos de datos y estructuras básicas

02

Estructuras de control

Sentencias condicionales y bucles

03

Funciones

Importancia en la modularidad del código

1A

Repaso de Python

Bloque A

Qué veremos en Bloque A

- Tipos de datos y variables.
- Entradas y salidas.
- Comentarios.

Repaso

Tipos de datos

- Recordemos que las variables son nombres significativos que damos a un espacio de almacenamiento que típicamente se encuentra en la memoria principal.
- Las variables además definen un tipo de dato que ayuda a definir operaciones específicas sobre este.
- Python es un lenguaje de tipificado fuerte, es decir, que el tipo de datos no cambia en forma inesperada durante el código.
- Además, Python es, a diferencia de otros lenguajes, dinámicamente tipificado, es decir, el tipo de datos de una variable se define en tiempo de ejecución (run-time).

Repaso

Tipos de datos en Python

Texto	str
Numéricos	int, float, complex
Secuencias	list, tuple, range
Mapeos	dict
Conjuntos	set, frozenset
Booleano	bool
Binarios	bytes, bytearray, memoryview
None	NoneType

Repaso

Tipos de datos

- Los tipos de datos presentados son los oficial y nativamente soportados por Python.
  - Otros tipos de datos pueden ser soportados por medio de la extensión de la biblioteca nativa de Python.

- No todos los tipos de datos serán usados ya que algunos son de aplicación muy específica.

- Recordemos que un tipo de datos puede ser forzado hacia otro tipo de datos con un mecanismo llamado “cast” el cuan en Python se realiza envolviendo el dato o variable con el tipo de dato y dos paréntesis:

- Ejemplo:

```
int(1.0)
```

Repaso

Tipos de datos: Strings

- String es un tipo de datos que merece atención especial.
- Un “string” es una secuencia de caracteres que representa un texto.
- Incluso si el “string” contiene solo números, su interpretación es como texto por lo que debemos aplicar el mecanismo “cast” si buscamos trabajar el valor como número.
- Cada caracter se indexa desde 0 hasta el tamaño del string (`len(string)`) menos 1.
- Ejemplo:

```
nombre = “Talento Futuro”
print(nombre[0]) # imprime T
```

Repaso

Tipos de datos: Strings

- El tipo de dato “string” se considera “inmutable”, es decir, los caracteres NO pueden ser cambiados una vez que el string se ha definido.
- Si deseamos modificar un string tendremos que construir un nuevo string con las modificaciones necesarias.
- Dos variables de tipo “string” pueden ser concatenadas usando el operador +.
- Ejemplo:

```
palabra_uno = “Talento”
palabra_dos = “Futuro”
print(palabra_uno + “ “ + palabra_dos) # imprime “Talento Futuro”
```

Repaso

Tipos de datos: Strings

- Existe otra operación interesante: “slice”.
- Un “string” puede ser rebanado (“sliced”) usando los corchetes cuadrados indicando el inicio y el final.
- Cada vez que apliquemos la operación “slice” Python retornará un “string” correspondiente a la rebanada desde el valor de inicio hasta el valor final – 1.
- Ejemplo:

```
nombre = “Talento Futuro”
print(nombre[0:7]) # imprime “Talento”
```

Repaso

Tipos de datos: Strings

- Un tercer parámetro especifica la forma de obtener la rebanada.
- Por ahora (más adelante veremos otras formas) diremos que con “-1” indicamos que el slice se debe obtener en forma inversa.
- Por lo tanto, si queremos invertir un string, basta con hacer lo siguiente:


```
nombre = “Talento Futuro”
print(nombre[::-1]) # imprime “orutuF otnelaT”
```

Repaso

Comentarios

- Recordemos que los lenguajes nos permiten definir texto que no es interpretado ni compilado.
- ¡Este texto debe ser bien usado!
    - Es decir, el texto debe ser usado para describir cuestiones no obvias.
    - ¡Y por ningún motivo como excusa para dejar código sin uso!
- En Python los comentarios se definen con el símbolo “#” para una o más líneas, o con tres comillas dobles al comienzo y al final de un texto más largo.

Repaso

Comentarios

```
# Este es un comentario


"""
Este también
es un
comentario
"""
```

Repaso

Entradas y salidas

- En todo algoritmo las entrada y salidas son esenciales.
- Las entradas desde el mecanismo estándar se realizan por medio de `input()`.
- Las entradas capturadas con `input()` son siempre entregadas como strings por los que si debemos trabajar con números tendremos que recurrir al “cast” de estas entradas.
- Las salidas por el mecanismo estándar se realizan por medio de `print()`.
- `print()` en Python está sobrecargado para todo tipo de datos.

Repaso

Entradas y salidas

- Ejemplo:

```
edad = int(input("Ingrese edad: "))
print("La edad es: ", edad)
```

o

```
edad = int(input("Ingrese edad: "))
print("La edad es: " + str(edad))
```

Trabajo grupal – Bloque A

Paralelo 2

G1
Nicolas Mardones
Manuel Denis
Bryan Castillo
GERALDY SUAREZ
Scarlett Espinoza
Ulises Campodónico

G2
Víctor Meza Herrera
Estefania Manriquez
Patricio Vera
Oscar Torres
Braulio Quiroz

G3
Daniela Méndez Gándara
Álvaro Pérez
Pedro Nahum
Javier Gajardo
Angela Proboste Neira

G4
Claudia Blanco
Ariel Inostroza
Héctor Aguayo
Ruben Sanhueza Ramirez
Félix González

G5
Fabian Díaz
Natalia Rivera
Juan Salinas
Rodrigo Pastén Cortés
Flavio Jara R.

G6
Camila Oyarzún
Stefanya Pulgar
Carlos Emilio Azócar Riquelme
Nicolas Rojas
luis.paillan.cnc@gmail.com

G7
Mayerlyn Rodriguez
Sebastian Vega
Efrain Duarte Campos
Bianel Bianchini
Bastián Gamboa Labbé

G8
Daniela Porto
Cristian Chavez Jara
Juan Rodrigo Vega
Rodolfo Cantillana
Abraham Ruiz

Trabajo grupal – Ejercicio #1

Cálculo de la raíz cuadrada

- Elaborar el código para el cálculo de la raíz cuadrada de un número.
- Restricción: no se puede usar sqrt de la biblioteca de Python.
- Debe solicitar el número por la entrada estándar e imprimir el resultado.

Trabajo grupal – Ejercicio #2

Calcular la distancia euclidiana entre dos puntos

- Elaborar un código para el cálculo de la distancia euclidiana entre dos puntos.
- Recuerde que la distancia euclidiana se calcula en base a las coordenadas x e y de ambos puntos.
- Debe solicitar las coordenadas de los puntos por la entrada estándar e imprimir la distancia.
- La fórmula del cálculo es:

$$distancia = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Trabajo grupal – Ejercicio #3

Invertir palabras

- Escriba un programa que solicite al usuario una palabra.
- Invierta la palabra y guárdela en una nueva variable.
- Imprima por pantalla la palabra original y luego la palabra invertida.

Break!

1B Estructuras de Control y Funciones

Bloque B

Qué veremos en Bloque B

- Estructuras de control.
- Funciones.

Estructuras de Control

Estructuras de control

- Python es un lenguaje de alto nivel que soporta diversos paradigmas.
- Uno de los paradigmas base que soporta es el procedural.
- Como todo lenguaje procedural, Python debe soportar las siguientes dos estructuras básicas para control:
    - Estructuras de control por decisiones.
    - Estructuras de control por ciclos.

Estructuras de Control

Decisiones en Python

- Las decisiones en Python se toman con la estructura if.
- Al igual que en otros lenguajes, la estructura if es seguida de una condición que, de ser verdadera, se ejecuta el código abarcado.
- A diferencia de otros lenguajes, Python define el término de la decisión con el signo ":" y luego el código abarcado se establece por la indentación (es decir, no hay llaves u otro mecanismo para encerrar el código abarcado).
- Ejemplo:

```
if a > b:
    print("a es mayor que b")
```

Estructuras de Control

Decisiones en Python

- La estructura “if” en Python se puede completar con dos opciones: un else, o una serie de else-if (elif):

- Ejemplo:

```
if a > b:
      print(“a es mayor que b”)
elif b > c:
       print(“b es mayor que c”)
else:
       print(“No se cumple la condición.”)
```

Estructuras de Control

Ciclos en Python

- En Python tenemos dos mecanismos frecuentemente usados para definir ciclos.
- El primero de ellos es el ciclo tradicional “while” mientras que el segundo es el ciclo “for” que típicamente se usa en conjunto con el patrón de diseño iterador ya que es usado para recorrer elementos iterables.
- Ejemplo:

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

Estructuras de Control

Ciclos en Python

- En el caso de un ciclo for, podemos iterar, por ejemplo, strings, listas, diccionarios.

- Ejemplo:

```
nombre = “Juan”
for caracter in nombre:
        print(caracter)

nombres = [“Juan”, “Camila”]
for nombre in nombres:
        print(nombre)
```

Funciones

¿Qué son las funciones?

- Las funciones son una de las estructuras que proveen los lenguajes estructurados para modularizar código.

- La modularización de código se hace necesaria cuando la complejidad de este hace que sea difícil entender como un todo.

- Las funciones son llamadas en algún punto del programa y momentáneamente se interrumpe la ejecución del programa principal para ejecutar el código de la función.

- Una vez que la función ha terminado su ejecución, se continúa con la ejecución del programa principal (si no ha habido un fallo en la función).

Funciones

Funciones en Python

- Las funciones en Python se definen usando la palabra reservada “def”.
- Luego sigue el nombre de la función y dos paréntesis como en las funciones matemáticas.
- Ejemplo:

```
def mi_funcion():
        print(“Hola desde una función!”)
```

- A diferencia de otros lenguajes, las funciones en Python definen el contenido por medio de la indentación y el signo “:”.
- Las funciones pueden definir variables en su interior y el alcance de estas es solo a nivel de la función.

Funciones

Funciones en Python: Parámetros

- En Python no se precisa del tipo de dato para los parámetros.
- Los parámetros se definen separados por coma entre los paréntesis de la función.
- Ejemplo:

```
def mi_funcion(nombre, apellido):
        print(nombre + " " + apellido)
```

- El llamado a la función queda entonces como: `mi_nombre("Juan", "Pérez")` y el resultado sería la impresión de 'Juan Pérez'.
- Las funciones pueden definir variables en su interior y el alcance de estas es solo a nivel de la función.

Trabajo grupal – Bloque B

Paralelo 2

G1
Nicolas Mardones
Manuel Denis
Bryan Castillo
GERALDY SUAREZ
Scarlett Espinoza
Ulises Campodónico

G2
Víctor Meza Herrera
Estefania Manriquez
Patricio Vera
Oscar Torres
Braulio Quiroz

G3
Daniela Méndez Gándara
Álvaro Pérez
Pedro Nahum
Javier Gajardo
Angela Proboste Neira

G4
Claudia Blanco
Ariel Inostroza
Héctor Aguayo
Ruben Sanhueza Ramirez
Félix González

G5
Fabian Díaz
Natalia Rivera
Juan Salinas
Rodrigo Pastén Cortés
Flavio Jara R.

G6
Camila Oyarzún
Stefanya Pulgar
Carlos Emilio Azócar Riquelme
Nicolas Rojas
luis.paillan.cnc@gmail.com

G7
Mayerlyn Rodriguez
Sebastian Vega
Efrain Duarte Campos
Bianel Bianchini
Bastián Gamboa Labbé

G8
Daniela Porto
Cristian Chavez Jara
Juan Rodrigo Vega
Rodolfo Cantillana
Abraham Ruiz

Trabajo grupal – Ejercicio #1

Modularizar código

- Tomar los tres casos del trabajo grupal anterior y re escribirlos como funciones.

Trabajo grupal – Ejercicio #2

¿Es palíndromo?

- Escriba un programa reciba como entrada una palabra.
- Determine si la palabra ingresada es palíndroma.
- Imprima por pantalla si la palabra es palíndroma.

Trabajo grupal – Ejercicio #3

¿Está en el rango?

- Escriba una función que permita determinar si un número se encuentra en un rango determinado.
- La función debe tener como parámetros el número a testear, el valor inferior del rango, el valor superior del rango.
- La función debe retornar True si el número está en el rango; caso contrario retorna False.

Trabajo grupal – Ejercicio #4

¿Cuál es el número mayor?

- Escriba un programa que continuamente pregunte por un número hasta que se escriba la palabra “FIN”.
- En cada ciclo el programa debe actualizar el valor mayor y reportarlo por pantalla.
- Si no ingresa un valor mayor al anterior, el programa debe continuar mostrando el último valor considerado como mayor.
- ¿Podría reescribir el programa para determinar el número menor?
- Nota: no se puede usar instrucción de tipo break.

¿Preguntas?

¡Hemos llegado al final de la clase!

TALENTOFUTURO