UNIVERSIDAD NACIONAL DE HURLINGHAM

Estructuras de datos

Profesor

Sergio Gonzalez

Modo de cursada

- Tema nuevo cada semana
- Miércoles previo habilitamos pestaña en campus con video y ejercicios
- Lunes: Clase teórico-práctica presencial
- Viernes: Clase práctica virtual sincrónica
- Consultas asincrónicas por Discord en la semana (**matricularse-materias-discord**)

Temas

- Primer Parcial:
  - Tipos de datos abstractos
  - Recursividad
  - Arreglos uni y multidimensionales
  - Pilas y Colas dinámicas
- Segundo Parcial:
  - Diccionarios y conjuntos
  - Listas enlazadas
  - Arboles

Fechas importantes

- Cuatrimestre: 25/03 al 02/08
- Primer Parcial: 13 de Mayo.
- Segundo Parcial: 1 de Julio.

- Recus: 8 y 12 de Julio

Regimen académico

- Parciales se aprueban con 4.
- Promoción
  - Promedio mayor o igual a 7
  - 6 o más en ambos parciales o recuperatorios
  - Nota: 6 y 7 **NO** promociona
- Regularización
  - Promedio entre 4 y 6
  - Integrador
  - Final

UNIVERSIDAD NACIONAL DE HURLINGHAM

Introducción a Python
Parte 1

Elementos básicos de lenguajes de programación

- Variables
- Palabras reservadas
- Entrada / Salida
- Expresiones / Operadores
- Comentarios

Que es PYTHON?

- Python es un lenguaje de programación interpretado con una sintaxis que favorece la legibilidad del código.

Lenguaje de programación Python

- Creado a finales de los 80s por Guido Van Rossum
- Multiparadigma: Orientado a objetos, programación funcional y programación imperativa.
- Licencia GNU GPL
- Código legible y transparente

Elementos básicos de un lenguaje de programación

- Variables
- Entrada – Salida
- Operadores
- Estructuras de control
- Comentarios

Variables

- Contenedores de datos
- Ocupan espacio en RAM
- Definición:
  - Tipo
  - Nombre / Identificador
    - Letras
    - Números
    - Caracteres especiales

Nombre / Identificador

- **NO** pueden empezar con un numero
- **NO** se pueden usar palabras reservadas

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

Tipos de datos

- Al programar, elegimos los tipos de datos a utilizar para cada variable
  - Define rango acotado (Ahorro de memoria)
  - Operaciones permitidas

Tipos de datos primitivos

- Implementados en el lenguaje
- Representación en la computadora de datos enteros, reales, lógicos, caracteres, etc.
- Interpretación de un patrón de bits

Booleanos

- “bool”

- Lógica booleana
  - Verdadero (True)
  - Falso (False)

Cadenas de caracteres

- “string”

- Serie de caracteres
  - Palabras
  - Se definen entre comillas (dobles o simples)

Enteros

- “int”

- Números enteros con signo
  - En general el rango posible depende del tamaño de la variable en memoria.

Punto flotante

- “float”

- Números reales con decimales
- Precisión: Cantidad de cifras decimales

Declaración de variables

- **NO** se declaran definiendo el tipo
- El tipado es automático y dinámico
- Funciones para el cambio de tipo (casteo):
  - int()
  - float()
  - str()

Declaración de variables

```
nombreVariable = valor
```

Ejemplos:

```
dias = 2
decision = True
letra = "C"
radio = 25.63
dias = "lunes"

x , y , z = 34 , 25 , 12
x = y = "Hola"
```

Entrada / Salida

- Por pantalla / teclado

Imprimir cosas por pantalla

Función "print"

```
print(<numero fijo>)

print("texto a imprimir")

print(variable) -> Imprime contenido de la variable
```

Ingreso de datos por teclado

- Para leer los datos que se introducen en el teclado se utiliza la función **input()**

- Se puede utilizar de la siguiente manera:

```
variable = input("Ingrese un numero: ")
```

Ingreso de otros tipos de datos

- Por defecto, la función convierte la entrada a una variable de tipo “string”
- Pero es posible “castear” las variables

```
variable = int(input("Ingrese un numero entero: "))
```

Expresiones / Operadores

- Operar con tipos de datos primitivos
  - Asignación
  - Aritméticos
  - Comparación
  - Lógicos

Operadores de asignación

- Asignar valores en variables
  - =
  - += / -=

Operadores aritméticos

operador	significado
+	Suma
-	Resta
*	Producto
/	División
//	División entera
%	Módulo (resto)

- Diferente función con diferentes tipos de datos

Operadores de comparación

operador	significado
<	Menor
>	Mayor
>=	Mayor o igual
<=	Menor o igual
==	Igual
!=	Distinto

- Resultado booleano

Operadores lógicos

operador	significado
not	No lógico (NOT)
and	"Y" lógico (AND)
or	"O" lógico (OR)

- Resultado booleano

Expresiones

- Formadas por operadores de comparación y lógicos
- Resultado booleano
- Condiciones complejas

Comentarios

- Lineas del programa que no se ejecutan
- Si son de una sola línea debe comenzar con "#"
- Si ocupan más de una línea van entre ‘ ’ ’

Ejercicios

- Podemos hacer los ejercicios de la primer parte

Estructuras de control

- Ya sabemos declarar variables, hacer operaciones de distintos tipos entre ellas, ingresar y sacar datos del programa
- Nos falta estructurar el código
- Ejecución condicional o repetitiva de líneas de código según resultado de expresiones

Estructuras de control

- Modificar el orden de ejecución de los pasos del algoritmo
  - Selectivas
  - Repetitivas

Estructuras de control selectivas

- Bifurcaciones en el flujo del programa.
- Decidir que hacer a partir de evaluar una condición.
- Uso de operadores y expresiones lógicas.

Alternativas simples

- Si – Entonces / If – then.
- Ejecuta una acción solo cuando se cumple una condición

```
if condicion:
    S1
    S2
    ...
    Sn
```

Alternativas dobles

- Si – Entonces – si_no / If – then – else.

```
if condicion:
    acciones S1
else:
    acciones S2
```

Alternativas múltiples

- Si – Entonces – si_no – entonces ...
- If – then – else if – then ...

```
if condicion1:
    acciones S1
elif condicion2:
    acciones S2
elif  condicion3:
    acciones S3
```

Estructuras de control repetitivas

- Conjunto de operaciones que se deben repetir varias veces.
- Ciclo o bucle: Parte de un programa que se repite (iteración) un número dado de veces o mientras se cumpla una condición.

Estructuras de control repetitivas

- Desde – Hasta / For

```
for variable in elemento_iterable:
    <acciones>
```

Estructuras de control repetitivas

- Desde – Hasta / For
- Uso alternativo a iterar colecciones:
  - Repetición de bloque de código 10 veces usando la función range:

```
for var in range(10):
    <bloque codigo>
```

Estructuras de control repetitivas

- Mientras / While

```
while condicion:
    <accion>
```