

CS1111 - Programación 1

Laboratorio 1B

Profesor: Julio Yarasca



Profesores:

De Teoría:

Jesus Edwin Bellido Angulo

Vicente Machaca Arceda

De Laboratorio:

María Hilda Bermejo Ríos

José Chavez Alvarez

Henry Giovanni Gallegos Velgara

Hugo Allain Mori Paiva

Julio Yarasca Moscol

Wilder Nina Choquehuayta

Fernando Nuñez Calderón

Percy Quevedo Vega

Jonathan Markham Silva Mercado

Carlos Sotomayor Beltrán

Jorge Villavicencio

Unidad 2: Python

UTEC

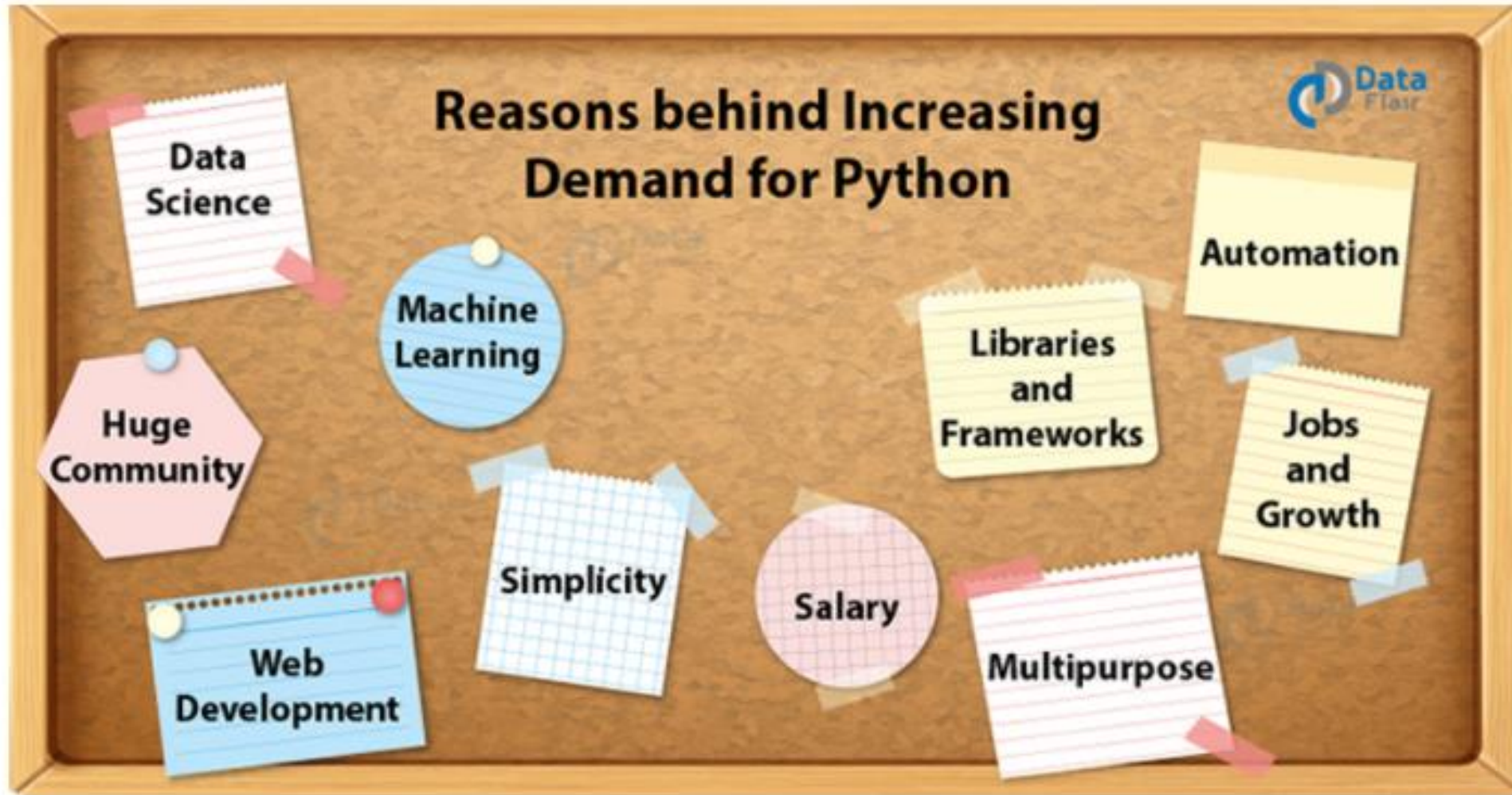
Logro de la sesión:

Al finalizar esta sesión, conocerás el ambiente integrado de desarrollo Pycharm, crearás proyectos y codificarás tus algoritmos.

Contenido:

- Por qué usar Python
- Recomendaciones para instalar el Pycharm
- Crear un proyecto
- Conocer y utilizar instrucciones para entrada y salida de datos en Python

Por qué usar Python?



Recomendaciones para instalar el Pycharm

Es el ambiente de desarrollo - IDE, que se utilizará en el curso. Es decir el software oficial del curso.

Pycharm, lo provee la compañía JetBrains.

Para poder instalar el software en tu computador o laptop, primero se tiene que solicitar autorización o licencia del uso del software por un año, esto se realiza directamente en la pág de JetBrains.

Te invitamos a ver los videos tutoriales, para que según el sistema operativo de tu computador puedas elegir el video apropiado.

Es importante que notes, que también debes descargar el intérprete de Python y realizar el enlace con el Pycharm

Pycharm



COMPUTING-UTEC

Programación 1

Instalación de Python y PyCharm Professional



macOS

By Christian Ledgard



Windows

By Christian Ledgard



Linux

By Luis Ponce

Para acceder a los videos visita este link:

<https://sites.google.com/utec.edu.pe/cs-utec/home>

Se recomienda instalar la versión profesional y “no” la Community Edition

Recomendaciones para crear un proyecto

El Pycharm, permite crear proyectos utilizando dos opciones:

1. Un ambiente virtual
2. Utilizando el intérprete instalado en el disco de la computadora

Se recomienda crear proyecto utilizando el intérprete instalado en el disco de la computadora, debido a que se crean los archivos estrictamente necesarios y esto facilitará que se suba las respuesta cuando den prácticas calificadas.

Recomendaciones para crear un proyecto

Para crear el proyecto se recomienda:

1. Crear una carpeta
2. Crear el proyecto utilizando el intérprete instalado en el disco duro
3. Verificar la ruta donde se están grabando los archivos que forman parte del proyecto

Ingreso y Salida de Datos

`input` permite ingresar datos desde el teclado. En otras palabras, enviar información desde la consola hacia el programa.

main.py

```
1 nombre = input("Ingresa tu nombre:")  
2 print("Hola " + nombre)  
3  
4
```

Console

Shell

```
Ingresa tu nombre:Pedro  
Hola Pedro
```



Ingreso y Salida de Datos

input

- La instrucción `input` retorna una cadena.
- Se pueden realizar operaciones con la variable creada.
- Es posible convertir de cadena a entero utilizando el constructor `int`

main.py

```
1 n = input("Ingresa un numero:")
2 print(n * 5)
3
```

Console

Shell

```
Ingresa un numero:7
77777
❖
```

main.py

```
1 n = int(input("Ingresa un numero:"))
2 print(n * 5)
3
```

Console

Shell

```
Ingresa un numero:7
35
❖
```

Caracteres Especiales

Backslash

- Cuando se quiere utilizar comillas simples o dobles dentro de una cadena. También para utilizar otros caracteres, por ejemplo: backslash
- Cuando se quiere introducir fin de línea o tabulaciones dentro de una cadena.

```
1 s = 'Me gustan los helados D\'onofrio'
2 print(s)
3 s = "Lionel \"La Pulga\" Messi"
4 print(s)
```

Console

Shell

```
Me gustan los helados D'onofrio
Lionel "La Pulga" Messi
```

```
1 s = "Asi\nfunciona\nel\nfin\nde\nlinea"
2 print(s)
3 s = "Esto\tes\tuna\ttabulacion"
4 print(s)
```

Console

Shell

```
Me gustan los helados D'onofrio
Lionel "La Pulga" Messi
```

Tipos de Datos

- **Números enteros (int).**
- **Números de punto flotante (float).**
- **Números complejos (complex).**
- **Valores booleanos (bool).**
- **Cadenas de caracteres (str).**

Tipos de Datos

int

- Es de precisión ilimitada.
- Decimal, binario, octal, hexadecimal.

```
1  n = 45
2  print(n)
3  n = 0b10
4  print(n)
5  n = 0o10
6  print(n)
7  n = 0x10
8  print(n)
```

Console

Shell

```
45
2
8
16
```

Tipos de Datos

float

- Números literales, notación científica, constructor float.
- Decimal, binario, octal, hexadecimal.

```
1  n = 3.1415
2  print(n)
3  n = 15e4
4  print(n)
5  n = 1.616e-35
6  print(n)
7  n = float(7)
8  print(n)
9  n = float("3.1415")
10 print(n)
```

Console

Shell

```
3.1415
150000.0
1.616e-35
7.0
3.1415
```

Tipos de Datos

bool

- Representación lógica.
- Muy importante para las estructuras de control condicional y repetitiva.

```
1  b = True
2  print(b)
3  b = False
4  print(b)
5  b = bool(0)
6  print(b)
7  b = bool(4)
8  print(b)
9  b = bool(1.234)
10 print(b)
11 b = bool("")
12 print(b)
13 b = bool("test")
14 print(b)
```

```
True
False
False
True
True
False
True
>
```

Tipos de Datos

str

- Es inmutable (no puede ser modificado una vez que ha sido creado).
- Secuencia de caracteres.

```
1 s = "Estas 'comillas simples' dentro de comillas dobles"  
2 print(s)  
3 s = 'Estas "comillas dobles" dentro de comillas simples'  
4 print(s)  
5 s = """Tambien se puede usar triple "comilla doble" """  
6 print(s)
```

Console

Shell

```
Estas 'comillas simples' dentro de comillas dobles  
Estas "comillas dobles" dentro de comillas simples  
Tambien se puede usar triple "comilla doble"
```

Operadores Aritméticos

- Si ambos operandos son enteros (`int`), el resultado es entero.
- Si al menos un operando es decimal (`float`), el resultado es decimal.
- Suma, resta, multiplicación, división real, división entera, módulo y potencia.

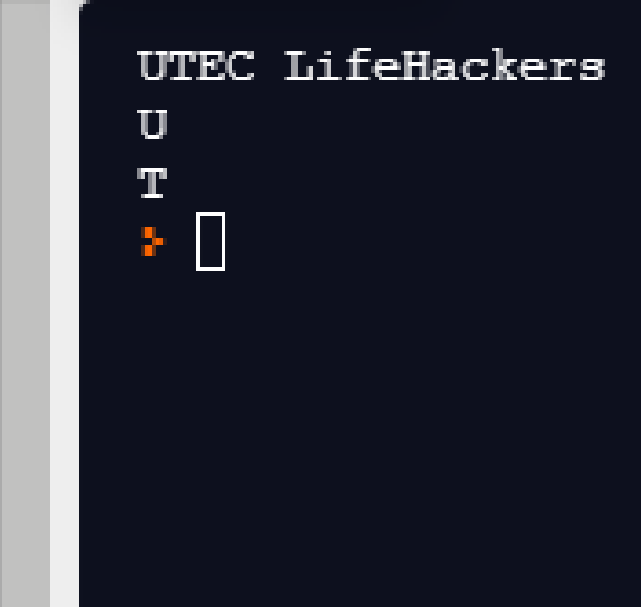
```
1 print("1: ", 5 + 2)
2 print("2: ", 5 + 7.5)
3 print("3: ", 5 - 2)
4 print("4: ", 5.0 - 2)
5 print("5: ", 5.2 * 10)
6 print("6: ", 5 * 2)
7 print("7: ", 5 / 2)
8 print("8: ", 5.0 / 2)
9 print("9: ", 5 // 2)
10 print("10: ", 5 % 2)
11 print("11: ", 5.0 % 2)
12 print("12: ", 2 ** 5)
13 print("13: ", 2.0 ** 5)
```

```
1: 7
2: 12.5
3: 3
4: 3.0
5: 52.0
6: 10
7: 2.5
8: 2.5
9: 2
10: 1
11: 1.0
12: 32
13: 32.0
>
```

Operaciones con Cadenas

- Unir dos cadenas (concatenar).
- Obtener un elemento de la cadena ([]).

```
1  a = "UTEC"
2  b = "Life"
3  c = "Hackers"
4  logo = a + " " + b + c
5  print(logo)
6  print(logo[0])
7  print(logo[1])
```



```
UTEC LifeHackers
U
T
> 
```


Importante: Identación

- En Python, la indentación (sangría, espacios en blanco al inicio de la línea) tiene un significado semántico. Por ello, es importante que, por ahora, todas las líneas que escribas empiecen desde la primera posición, es decir, que no estén indentadas.

Precedencia de Operadores

Prioridad	Operador
1	Potencia: **
2	Multiplicación, división, división entera, módulo: *, /, //, %
3	Suma, resta: +, -

```
1 print(3 * 5 + 1)
2 print(3 * (5 + 1))
```

```
16
18
```

Ejercicios

1. **Escribir un programa en Python que permita hallar el área y el volumen de una esfera.**
2. **Escribir un programa en Python que permita ingresar un monto en soles y muestre su equivalente en dólares y euros.**
3. **Escribir un programa que solicite al usuario su nombre y edad. Posteriormente, deberá indicarle en qué año cumplirá 100 años.**
4. **Escribir un programa que reciba una cantidad de minutos y calcule su equivalente en segundos.**
5. **Escriba un programa que reciba un número representando una cantidad de segundos. Posteriormente, el programa deberá calcular cuantos minutos y segundos hay en ese tiempo. Por ejemplo: 150 segundos -> 2 minutos 30 segundos.**

Cierre

En esta sesión aprendiste:

- **Crear un proyecto**
- **Los tipos de datos str, int, float y bool.**
- **Identación y sintaxis básica de Python.**
- **Realizar operaciones con diferentes tipos de datos.**

Gracias

Nos vemos en la siguiente
clase!

