



Jul. 2022

# Unidad 1: Conceptos de

Python



# Introducción

**Guido van Rossum** es el creador de Python

Actualmente ingeniero en Microsoft. ex-Dropbox, ex-Google, ...

Bautizó el lenguaje con el **nombre** de Monty Python

- 1989 durante las Navidades - lo creó por aburrimiento
- 1990: Lanzamiento
- 1995: "¿Python? ¿Qué es eso?"
- 1997: "¡Pero si nadie usa Python!"
- 1999: "¿Dónde puedo contratar programadores Python?"
- 2000: Python 2.0 - uno de los lenguajes más populares
- 2008: Python 3.0 - cambios significativos



# Introducción

1. Python es un lenguaje de programación de **propósito general**
  - interpretado,
  - interactivo,
  - orientado a objetos
  - con semántica dinámica.
2. Puede presumir de ser un lenguaje a la vez **fácil y potente**
3. Elegante **sintaxis y tipificación** dinámica
  - ideal para scripts y aplicaciones robustas
  - código sencillo de escribir y fácil de mantener
4. Con los **módulos y paquetes**
  - modularidad del programa y la reutilización del código
  - permite el desarrollo sea más rápido
5. El intérprete, la extensa biblioteca estándar y de terceros
  - Se **distribuyen libremente**
6. **Fácil de aprender** y de convertirse en experto
  - Aprender Python **no requiere ningún requisito** previo

# ¿Por qué Python es tan popular?

1. Funciona en **todas las plataformas**
  - Windows, GNU/Linux, macOS
2. Es un **lenguaje polivalente** puede aplicarse en:
  - Automatización
  - sitios web
  - aplicaciones web
  - interfaces gráficas de usuario
  - servidores de red
  - APIs de back-end
  - aplicaciones de escritorio
  - herramientas médicas
  - dispositivos IoT
  - analizar datos (big data)
  - cálculo científico,
  - aprendizaje automático
3. Las **empresas tecnológicas** adoran Python:  
Google, Youtube, Facebook, IBM,  
NASA, Dropbox, Yahoo, Mozilla,  
Quora, Instagram, Uber y Reddit, etc...

# Demanda de desarrolladores de Python

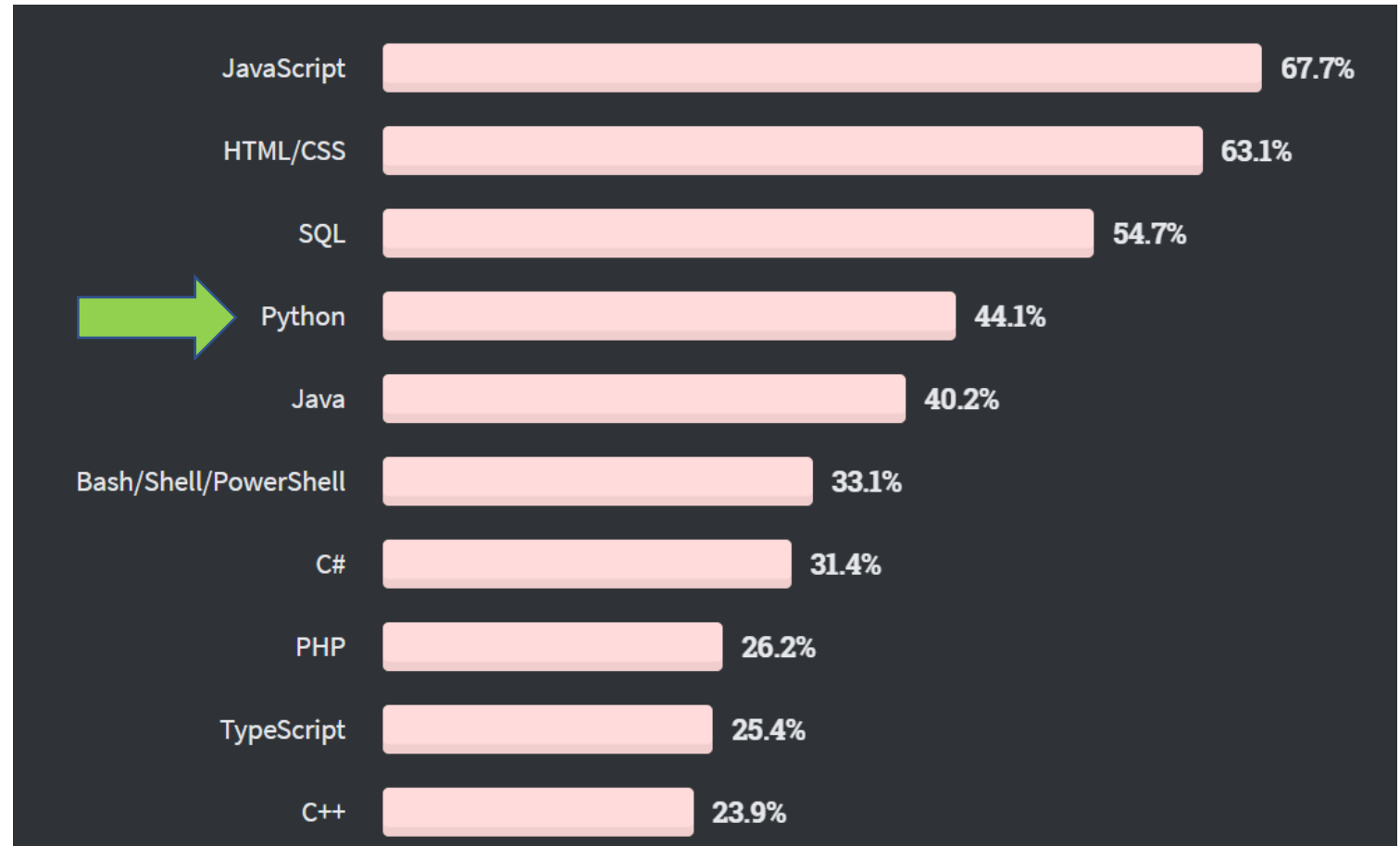
## Tendencias (2021):

- Python es el segundo **lenguaje** de programación más **popular**
  - Tiene una curva de aprendizaje pronunciada
- Se utiliza para **construir soluciones digitales** basadas en:
  - análisis de datos (DA)
  - aprendizaje automático (ML) y
  - inteligencia artificial (AI)
- Creciente popularidad de las tecnologías de IA, ML y DA
  - Continua demanda de programadores
  - Es una habilidad bien pagada

<https://www.daxx.com/blog/development-trends/python-developer-salary-usa>

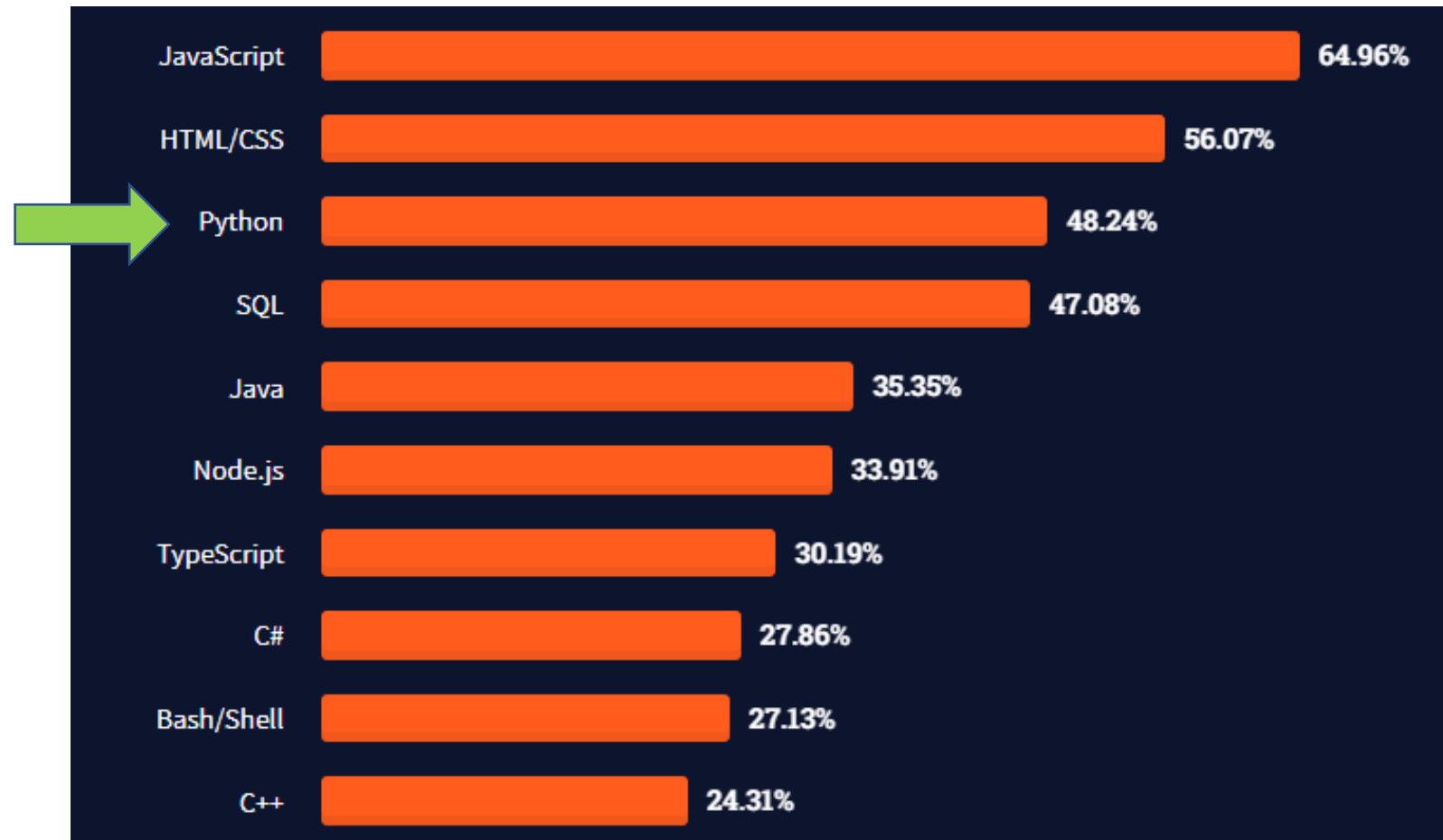


# Top 10 Tecnologías más populares en el mundo (2020 - 57378 encuestas)



<https://insights.stackoverflow.com/survey/2020#job-priorities>

# Top 10 Tecnologías más populares en el mundo (2021 - 83052 encuestas)



<https://insights.stackoverflow.com/survey/2021#job-priorities>



# ¿Qué hace un desarrollador de Python?

- Los lenguajes de programación son la forma en que los programadores expresan y comunican sus **ideas**, y la audiencia de esas ideas son otros programadores (NO los ordenadores)
- Una idea debe expresarse como un programa informático utilizando un lenguaje de programación
- Es responsable de
  - **escribir código** reutilizable y eficiente
  - desarrollar la **lógica** de la aplicación **del lado del servidor**
  - **desplegar** elementos **de cara al usuario**

# Bibliotecas y paquetes de Python

- Alojadas en el repositorio de software (PyPI)  
“Índice de Paquetes de Python”



Hay más de  
383.812 proyectos  
603,038 usuarios  
y sigue creciendo ...

# Diferentes versiones de Python

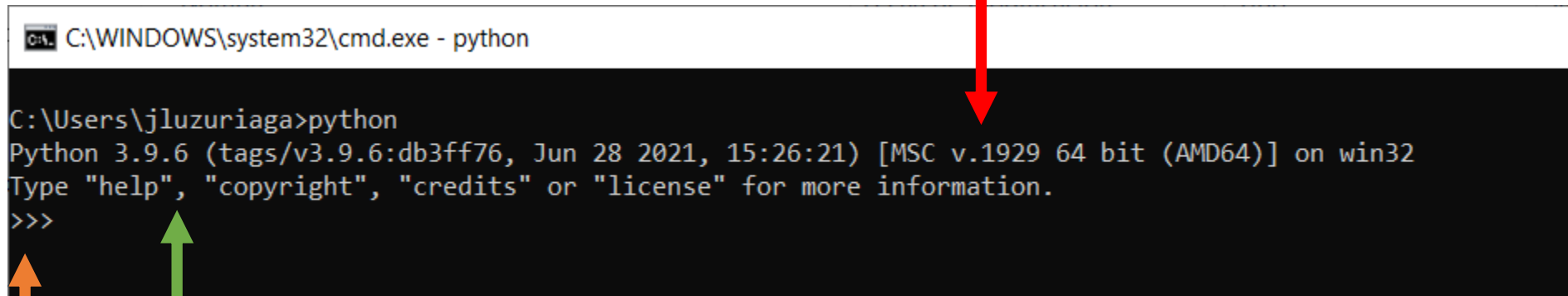
- Python 2 y Python 3 son las dos versiones populares
  - Ambas son muy similares
  - Hay algunas diferencias sintácticas que pueden traer problemas
- Gran parte del código escrito para Python 3 se ejecuta en Python 2 y viceversa
- **DEPRECATION**: *Python 2.7 reached the end of its life on [January 1st, 2020](#).*
  - Python 3 es el único que está en desarrollo activo (en la actualidad)
    - sólo se desarrolla para Python 3.
    - La mayoría de las librerías de terceros más utilizadas han sido portadas a Python 3
- Usaremos Python 3.8 para todos los ejemplos del curso



# Comprobar la instalación de Python

versión principal y la fecha de lanzamiento

el sistema en el que se está ejecutando la shell interactiva



```
C:\WINDOWS\system32\cmd.exe - python

C:\Users\jluzuriaga>python
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

ejemplos de comandos que podemos escribir

el prompt >>> para introducir un comando

# Intérprete de Python

- Python cuenta con un IDLE (*Entorno integrado de desarrollo y aprendizaje*)
  - una herramienta de desarrollo ideal para iniciarse en Python
  - Permite probar programas fácilmente sin tiempo de espera
  - Se puede editar y crear fácilmente archivos con extensión `.py`.
- IDLE primero lee la línea de un programa,
  - evalúa sus instrucciones,
  - las imprime en la pantalla
  - y hace un bucle en la línea siguiente.
- IDLE proporciona `tracebacks` para mencionar errores

# ¿Cómo utilizar el IDLE de Python?

- Como cualquier otro shell que interactúa con el sistema operativo (Bash o CMD)
  - además interactúa con el intérprete de Python con instrucciones Python
- Navegación en el shell
  - **Alt+P** para recuperar una instrucción escrita previamente
  - **Alt+N** para recuperar la siguiente instrucción
- Gestión y ejecución de los archivos de script
  - **Ctrl+N** nuevo
  - **Ctrl+O** abrir
- Escribir y formatear código

# Python Básico



# Objetivos de aprendizaje

- Utilizar Python IDLE (shell interactivo)
- Escribir y ejecutar scripts sencillos de Python
- Escribir y ejecutar scripts dinámicos
- Utilizar variables y describir los diferentes tipos de valores que se pueden asignar
- Obtener la entrada del usuario desde el teclado
- Explicar la importancia de los comentarios y la indentación

# E1: Trabajar con el intérprete de Python (*shell*)

- Imprimir un mensaje
- Devuelve cualquier cosa que escribas
- Hacer operaciones aritméticas
- Salir

```
>>> print("¡Hola Mundo!")
```

```
>>> "Eco"  
"Eco"  
>>> 1234  
1234
```

```
>>> 9 + 3  
12  
>>> 100 * 5  
20  
>>> -6 + 5 * 3  
9
```

```
>>> exit()  
>>> quit()
```

# Actividad 1: Trabajar con el intérprete

1. Imprime el mensaje "Happy birthday"
2. Ejecuta la operación  $17 + 35 * 2$
3. Imprime los números del 1 al 9 en una sola fila (sin bucles aún)
4. Salir del interprete

# Actividad 1: Trabajar con el intérprete

1. Imprime el mensaje "Happy birthday"
2. Ejecuta la operación  $17 + 35 * 2$
3. Imprime los números del 1 al 9 en una sola fila (sin bucles aún)
4. Salir del interprete

Resultado de la ejecución las instrucciones

```
>>> ?  
Happy birthday  
>>> ?  
87  
>>> ?  
1 2 3 4 5 6 7 8 9  
>>> ?
```

# Actividad 1: Trabajar con el intérprete

1. Imprime el mensaje "Happy birthday"
2. Ejecuta la operación  $17 + 35 * 2$
3. Imprime los números del 1 al 9 en una sola fila (sin bucles aún)
4. Salir del interprete

Resultado de la ejecución las instrucciones

```
>>> print("happy birthday")
Happy birthday
>>> 17+35*2
87
>>> print("1 2 3 4 5 6 7 8 9")
1 2 3 4 5 6 7 8 9
>>> exit()
```

# Escribir y ejecutar scripts

- Shell
  - Probar una hipótesis rápida
  - Comprobar si un método específico existe para algún tipo de datos.
  - No se puede escribir un programa completo
- Script (modulo)
  - Cualquier cosa que se puede ejecutar en shell se puede ejecutar en un script
  - Un archivo que contiene instrucciones de Python
  - Debe tener la extensión de archivo **.py**

## E2: Crear un Script

1. Abra su editor de texto.
2. Crea un archivo llamado `test1.py` e inserte el comando para imprimir “hola mundo”:
3. Guárdalo.
4. Abre tu terminal (cambia al directorio del archivo)
5. Ejecuta:  
`python test1.py.`



# Ejecución de un archivo que contiene comandos no válidos

- La introducción de instrucciones no válidas provoca un error
- En una salida que se llama trazas ([stack trace](#))
- Nos dice cosas útiles como
  - **dónde** ocurrió el error
  - qué **tipo de error**
  - qué otras llamadas se dispararon en el camino
- Las trazas deben leerse de [abajo a arriba](#)
- Identifica qué está causando el error para actuar en consecuencia

# E3: Pasar argumentos de usuario a los scripts

- Scripts dinámicos
  - Hacer que el usuario proporcione argumentos al llamar al script

1. Crea un nuevo archivo llamado `test2.py`

2. Luego, añade el siguiente código:

```
import sys  
sys.argv[1]
```

3. Guarda y ejecuta el script pasándole un argumento  
`python test2.py argumento1`

# Actividad 2: Ejecutar scripts simples

- Crear un script para generar de tarjetas de presentación
- Se deben pasar dos parámetros con el script (nombre y apellido)

1. Crear un script llamado `tarjeta.py`
2. Utilizar la sentencia para imprimir  
el Nombre y el Apellido  
20 guiones bajos (como bordes superior e inferior)
3. Ejecute el script pasando dos argumentos

# Actividad 2: Ejecutar scripts simples

- Crear un script para generar de tarjetas de presentación
- Se deben pasar dos parámetros con el script (nombre y apellido)

1. Crear un script llamado `tarjeta.py`
2. Utilizar la sentencia para imprimir el Nombre y el Apellido

20 guiones bajos (como bordes superior, izquierdo e inferior)

3. Ejecute el script pasando dos argumentos

```
*-----  
| Primer nombre: jorge  
| Segundo nombre: Luzuriaga  
*-----
```

```
python tarjeta.py jorge Luzuriaga
```

# Sintaxis de Python

- Lo que se necesita para escribir un programa
- Cómo se estructuran las expresiones del lenguaje
- Variables
  - Referencias a valores en memoria con diferentes tipos de datos
  - Usadas para almacenar cualquier tipo de datos
  - El valor y el tipo de una variable pueden cambiar durante el tiempo de ejecución
- Valores
  - Se pueden asignar a las variables
  - **Tipos:** cadenas, enteros, decimales (flotantes), booleanos, y estructura de datos
  - Comprobación del tipo de un valor

```
>>> type(valor)
```

# Tipos de valores

- Valores numéricos - Enteros
  - Cualquier número positivo, negativo, incluyendo el 0
  - Nunca puede ser una fracción, un decimal, o un porcentaje

```
>>> type(3)  
<class 'int'>
```

- Cadena
  - Secuencia de caracteres que se coloca entre dos comillas (simples o dobles)

```
>>> type("Primera Semana de Julio")  
<clase 'str'>
```

# Conversión de tipos

- Si
  - una cadena tiene un entero dentro

```
>>> int(valor_cadena)
```

- un entero se quiere poner en una cadena

```
>>> str(valor_entero)
```



# Uso de variables

- Utilizamos las variables cuando en nuestro código tenemos un valor que queremos utilizar varias veces

```
>>> numero = 7
>>> numero * 5
>>> numero + 2
>>> numero / 3,5
>>> numero - numero
```

El valor de numero no cambiará a menos que lo reasignemos

```
>>> print(numero)
7
>>> numero = 22
>>> print(numero)
22
```

```
>>> x = numero + 1
>>> x
23
```

```
>>> mensaje = "I love Python"
>>> mensaje + "!" * 3
'I love Python!!!'
```

# Asignación múltiple

- Asignar múltiples variables en una sentencia

```
>>> a, b, c = 1, 2, 3
>>> print(a)
1
>>> print(b)
2
>>> print(c)
3
```

# Actividad 3: Uso de variables y sentencias de asignación

- Escribir un **script para calcular la velocidad** en
  - kilómetros por hora
  - millas por hora y
  - metros por segundoutilizando
  - la distancia (kilómetros) y el tiempo (horas)
- Se sabe que:
  - La fórmula para calcular la velocidad es distancia/tiempo
  - Para convertir
    - kilómetros en millas, divide los kilómetros por 1,6
    - los kilómetros en metros, multiplica los kilómetros por 1.000
    - las horas en segundos, multiplique las horas por 3.600
- Probar asignando:
  - `distancia_en_km = 150` y `tiempo_en_horas = 2`

$$\bar{v} = \frac{d}{t}$$

# Actividad 3: Uso de variables y sentencias de asignación

- Resultado de la ejecución del script

La velocidad en kilómetros por hora es: 75.0

La velocidad en millas por hora es: 46.875

La velocidad en metros por segundo es: 20.833333333333332

# Reglas para nombrar variables

1. Puede tener letras, símbolos y dígitos
2. No puede comenzar con un dígito

**7x** es un nombre de variable **inválido**

3. Debe evitarse los espacios
4. No se puede utilizar palabras clave de Python

*import, if, for, lambda, ...*

Ninguna del listado que se muestra en :

```
help()
help> keywords

ctrl+c
```

# Convenciones de nomenclatura de Python

- **Nombres de variables compuestos** **deben** escribirse en notación snake\_case

```
>>> primera_letra = "a"      # Snake  
>>> primeraLetra = "a"      # Camel  
>>> PrimeraLetra = "a"      # Pascal
```

- **Nombres de las constantes** **deben** escribirse en mayúsculas para denotar que sus valores no deben cambiar

```
>>> NUMERO_DE_PLANETAS = 8
```

- Evitar nombres de variables de un solo carácter que pueden confundirse ejemplo las mayúsculas I y O con 1 y 0

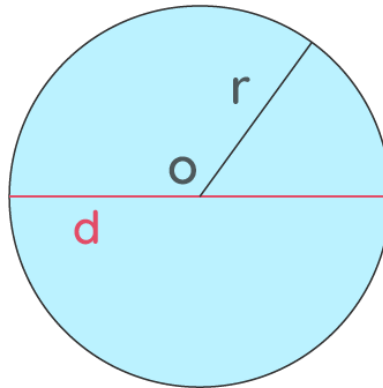
# Actividad 4: Asignación de variables

- Escribir un script que calcule el área y la circunferencia de un círculo con un radio de 7.
  - declara el valor constante,  $\pi$  (PI) = 3,14159
- Se sabe las fórmulas para calcular

$$\text{Area } A = \pi r^2$$

$$\text{Circumference } C = 2\pi r$$

$$\text{Diameter } d = 2r$$



- Resultado de la ejecución del script

Area: 153.93791

Circunferencia: 43.98226



# Entrada de usuario - función input()

- Para obtener la entrada del teclado del usuario desde el CLI
  - Se detiene la ejecución del programa hasta que el usuario después de completar su entrada presiona la tecla *Enter*
- No hay ninguna señal que nos permita saber cuándo escribir
  - Se puede pasar un aviso a la función de entrada

```
>>> mensaje = input("Introduce un texto: ")
```
- Los valores devueltos por la función de entrada son siempre cadenas

# Comentarios

- Son una parte fundamental en la programación
- Hacen que el código sea más fácil de entender para los humanos
- Puede decirnos
  - por qué se tomaron ciertas decisiones,
  - algunas mejoras que se pueden hacer en el futuro, y
  - explicar la lógica del negocio
- Hay tres formas de escribir comentarios
  - cadenas de documentación (docstrings) (envueltas entre comillas triples `'''`)
  - comentarios en línea (comienza con un signo `#`, misma línea)
  - comentarios en bloque (comienza con un signo `#`, mismo nivel de sangría)

# Indentación (Sangrado)

- Un bloque es un grupo de sentencias que deben ejecutarse juntas
- Los distintos lenguajes representan los bloques de forma diferente

```
if (true) {  
    // ejecuta este bloque de sentencias  
} else {  
    // ejecuta otro bloque de sentencias  
}
```

- En Python los bloques están indentados **en lugar de utilizar llaves** y pueden estar anidados dentro de otros bloques

# Actividad 4: Corregir las sangrías en un bloque de código

- Arregla el fragmento de código en el archivo [indentacionincorrecta.py](#) para tener la sangría apropiada en cada bloque.
- Una vez arreglado, ejecutar el script en la terminal
- Verificar que obtienes la siguiente salida:

```
1  
2  
3  
4  
5
```

# Actividad 5: Implementar la entrada del usuario y los comentarios en un script

- Escriba un script que tome de la entrada del usuario un número e imprima su tabla de multiplicación del 1 al 10

1. Añadir un docstrip explicando lo que hace el script
2. Asignar a variable llamada número la entrada del usuario
3. Convertir la entrada a un número entero
4. Imprimir 25 guiones bajos como superior e inferior de la tabla
5. Imprimir la multiplicación de ese número por cada número del 1 al 10

- Resultado de la ejecución del script

Generar una tabla de multiplicar para: 6

---

```
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
6 * 10 = 60
```

---

# Resumen

- Existen dos formas de ejecutar instrucciones de Python.
  - ejecutar comandos a través del shell interactivo de Python
  - ejecutando scripts guardados
- Hemos cubierto
  - la sintaxis de Python
  - la asignación de variables
  - diferentes tipos de valores
  - la importancia de las palabras reservadas
  - la función de entrada
  - las diferentes formas de escribir comentarios en el código
  - la importancia de la indentación