Introducción a Python
ANTEL

Sobre el curso

- **Horario:** Lunes, Miércoles y Viernes de 14:00 a 16:00
- **Salón: Plataforma Futura**
- **Fechas Previstas:**

  3/08 al 16/09 (40 hs)
- **Instructores:**

  Rodrigo Pérez

  Gustavo Signorele
- **Asistencia: 80 %**
- **Material:** https://corp.antel.com.uy/wiki/x/eH7LMQ
- **Prueba final:** Trabajo práctico grupal

Cronograma

- Introducción: Historia y Filosofía.
- Strings y salida en consola
- Estructuras y elementos del lenguaje
- Condicionales y control de flujo.
- Bucles y sentencias de repetición.
- Módulos, paquetes y namespaces
- Funciones definidas por el usuario
- Listas y Diccionarios
- Métodos del objeto list y del objeto dict
- Introducción a la Orientación a Objetos
- Programación funcional
- Módulos de la librería estándar
- Prueba teórica y práctica.

Antes de comenzar

- Presentación de los docentes
- Presentación del grupo:
    - Cada uno debe presentarse
    - Lugar de trabajo
    - Ideas previas
    - Expectativas del curso
    - ¿Programa generalmente, en qué lenguaje?
- Verifiquemos que tengamos instalado Python
    - Versiones 3.x - 3.7.2
    - PyCharm

Introducción

- ¿Qué es Python?
  - Lenguaje de programación de alto nivel
- Interpretado.
  - Traduce el programa poco a poco, leyendo y ejecutando cada comando de archivos *.pyc
  - No depende de la plataforma: Flexible y portable
  - Genera archivos *.pyc para el intérprete

Características

- Tipado dinámico
  - No es necesario declarar los tipos de variable y pueden variar a lo largo del programa.
- Fuertemente tipado
  - No se permite tratar una variable como si fuera otro tipo de variable.
- Multi paradigama
  - Orientación a objetos, imperativa y funcional.

Características (2)

- **Modos de ejecución**

  Modo interactivo:

  Los lenguajes interpretados permiten este modo de ejecución. Damos órdenes al interprete y obtenemos una respuesta inmediata a cada una (¡probemos!).

  Usando módulos

  Un módulo es un archivo almacenado en disco con extensión .py (lo ejecutamos con python.exe nombre_modulo.py)

Modos de ejecución

- Ejemplo

  Un módulo es un archivo almacenado en disco con extensión .py

  En windows, lo ejecutamos así:

  D:\Python36\python.exe factorial.py

  En linux,

  Python3.6 factorial.py o sino

  chmod +x factorial.py

  ./factorial.py

Características (3)

- **Software Libre**

  Comunidad muy grande

  comp.lang.python (Google Group)

- **Gran variedad de funciones y librerías**
- El interprete de Python es un programa, ¿En qué lenguaje está escrito ese programa?

Formateo de Python

- A diferencia de otros lenguajes, Python, tiene una manera muy definida de escribirse.
- Por ejemplo, en C ó Java el inicio y el fin de los bloques de código está marcados por { }. En Python, hay que tabular
- El símbolo : indica continuación de sentencia en el renglón siguiente
- Python nos obliga a hacer legibles nuestros programas, no hay alternativa

Paradigmas

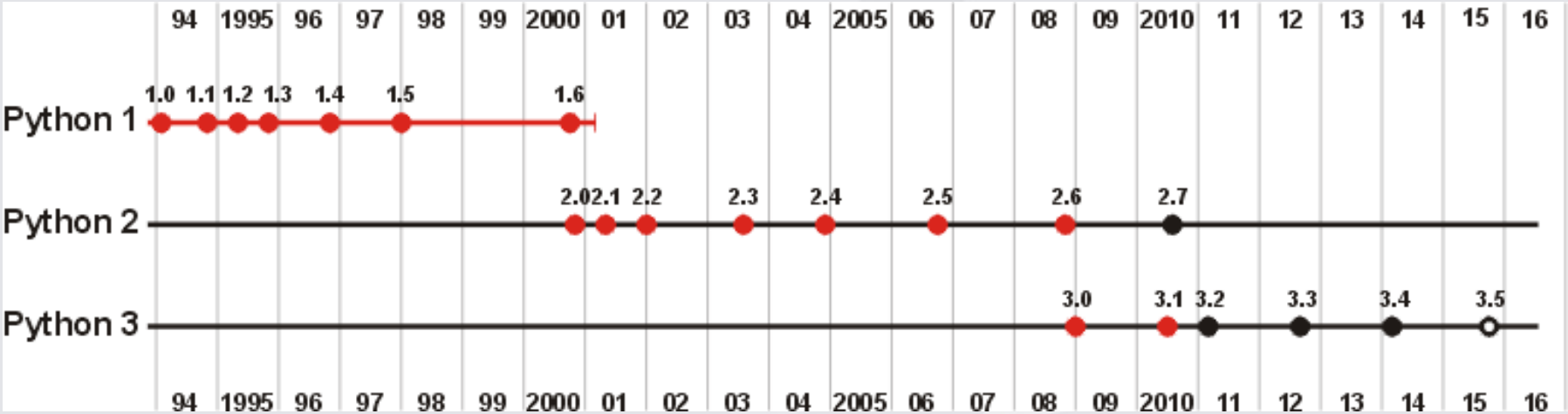
- Programación Orientada a objetos
  - Clases
  - Objetos
  - Propiedades
  - Herencia, Herencia múltiple
- Programación Funcional
  - Funciones de orden superior
  - Map, filter, reduce

Historia

- Creado por **Guido van Rossum** en el Centro de Matemáticas e Informática de Holanda.
- Apareció en 1991 la versión 0.9.0 con herencia, excepciones, funciones, str, list y dict
- En 1994 se liberó la versión 1.0.0 que Incluyó herramientas de programación funcional.

Historia (2)

- Las versiones de Python se identifican por tres números X.Y.Z
  - X – Grandes cambios en la versión.
  - Y – Cambios importantes
  - Z – Correcciones generales pequeñas y fallos de seguridad



2.7 vs 3.8

Python 2.7	Python 3.8
Versión 2.0 liberada en el 2000	Versión 3.0 se libero en 2008
Versión 2.7 liberada en 2010	Versión 3.8 liberada en 2018
Versión por defecto en Linux y Macs	
Aún existen un algunas dependencias que la utilizan	Aún se están migrando a la versión nueva
Librerías que soportan solo 2.7	Aún se están migrando todas las librerias.
Extensa documentación y ejemplos disponibles	Variada documentación
print es una sentencia	print es una función.
Problemas con caracteres Unicode	Por defecto maneja caracteres Unicode
División entre enteros devuelve estrictamente un entero	Puede o no devolver un entero.
Problemas con raw_input e input	Problemas solucionados

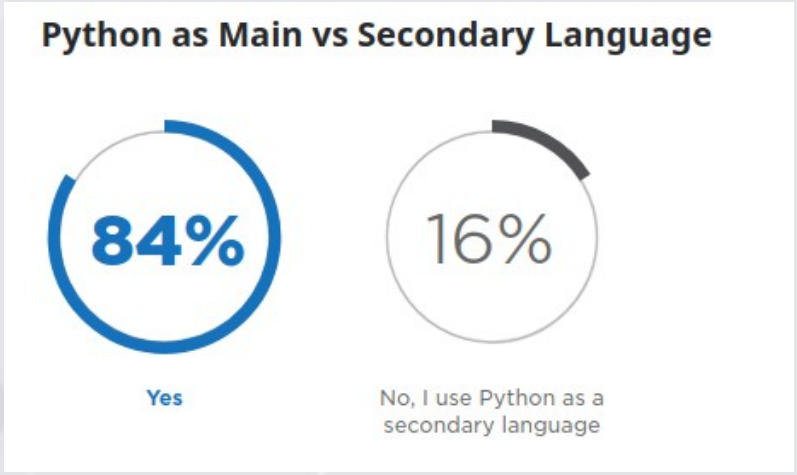
Python Survey

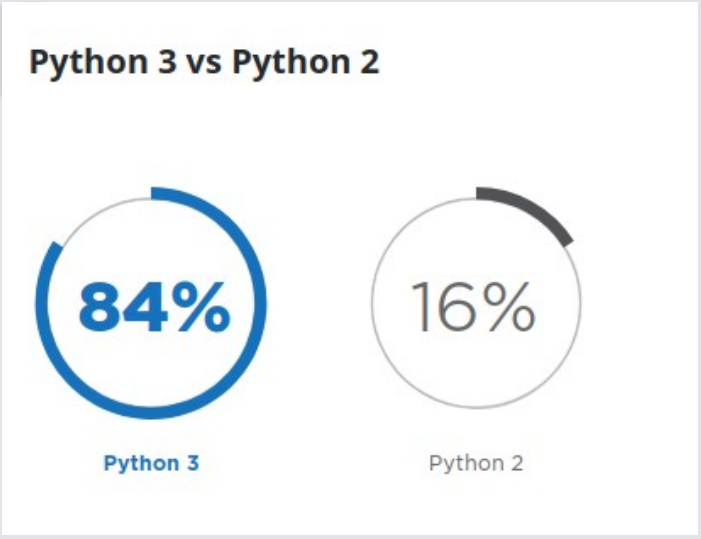
- Encuesta realizada por la Python Software Foundation
- Python 2 vs Python 3

  El uso principal de Python 3 está creciendo rápidamente.

  El uso de Python 2 está disminuyendo, ya que no se desarrolla activamente, no obtiene nuevas funciones y su mantenimiento se detuvo en comienzo del 2020.

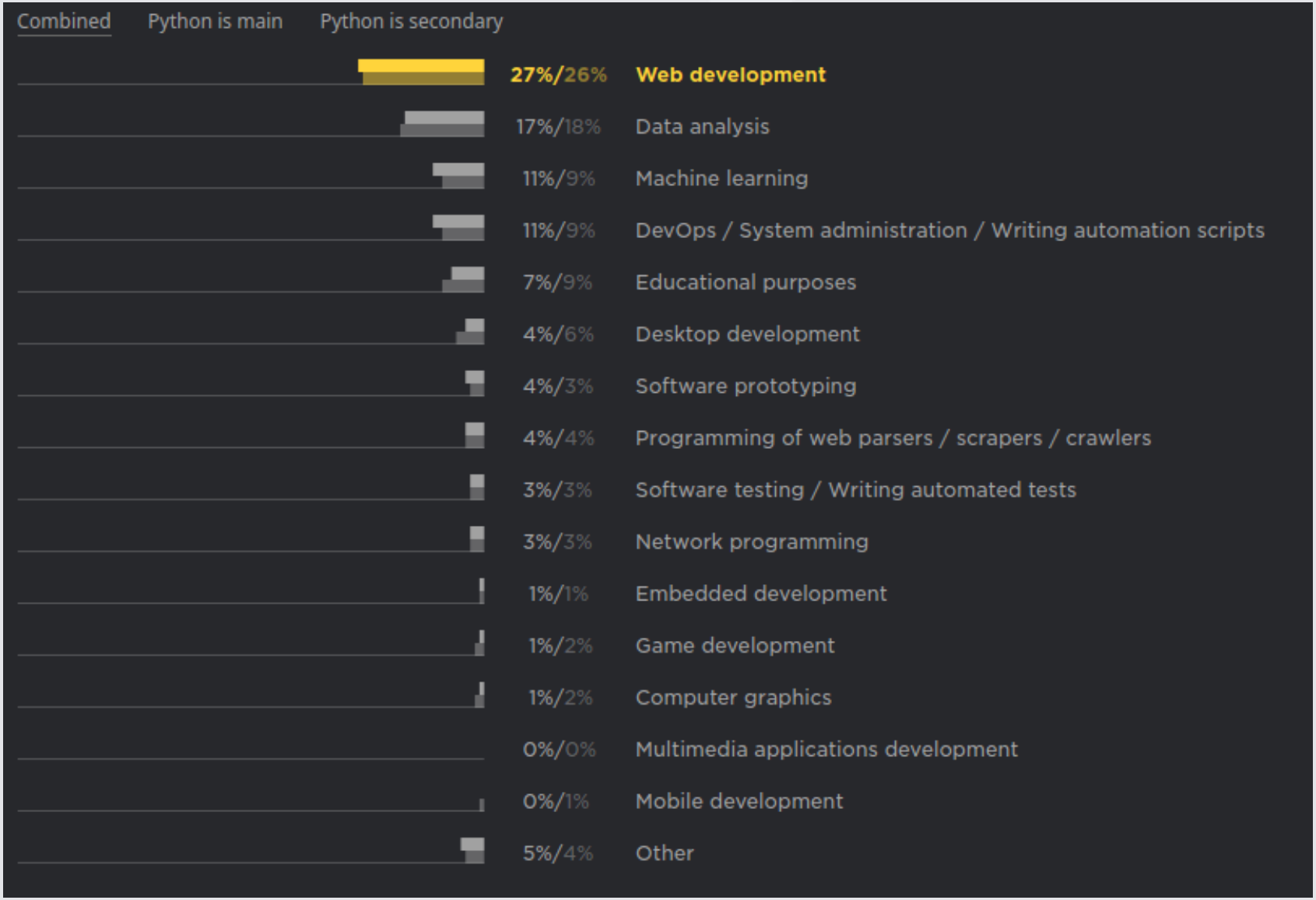
  https://pythonclock.org/





https://www.jetbrains.com/research/python-developers-survey-2018/

¿Para qué se utiliza Python?



¿Dónde se utiliza Python?

- Existen muchas aplicaciones en donde podemos utilizar Python.
- Más populares

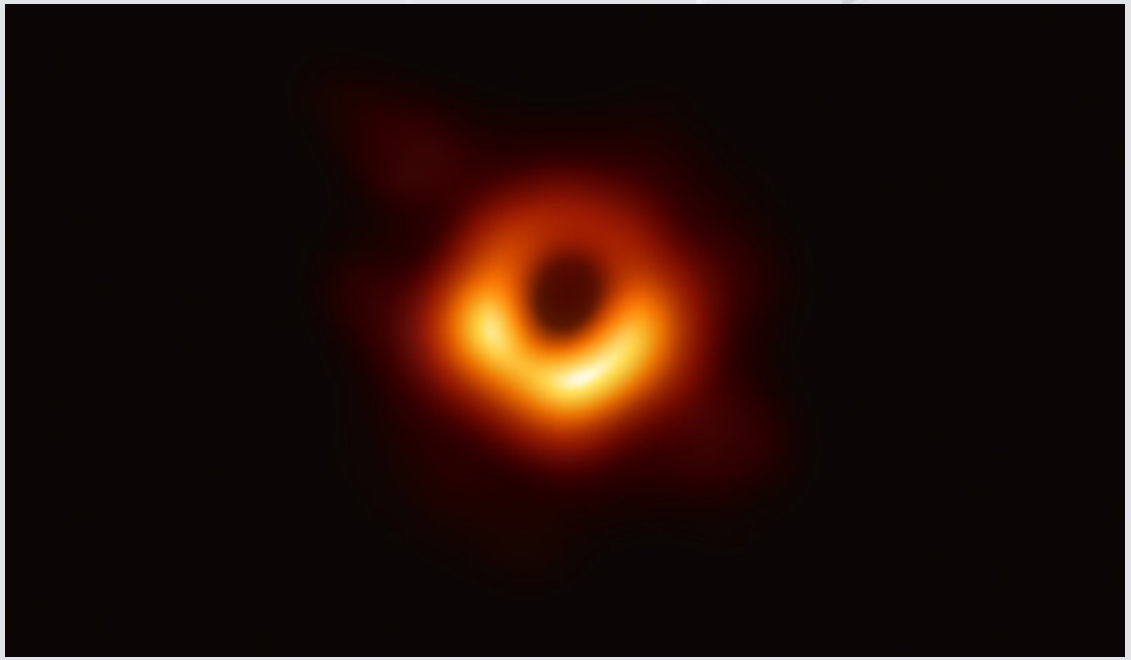
1) Desarrollo Web
   - Flask
   - Django

2) Data Science
   - Machine Learning
   - Deep learning
   - Análisis de datos
   - Visualización de datos

3) Scripting
   - Automatización de tareas

¿Dónde se utiliza Python? (II)

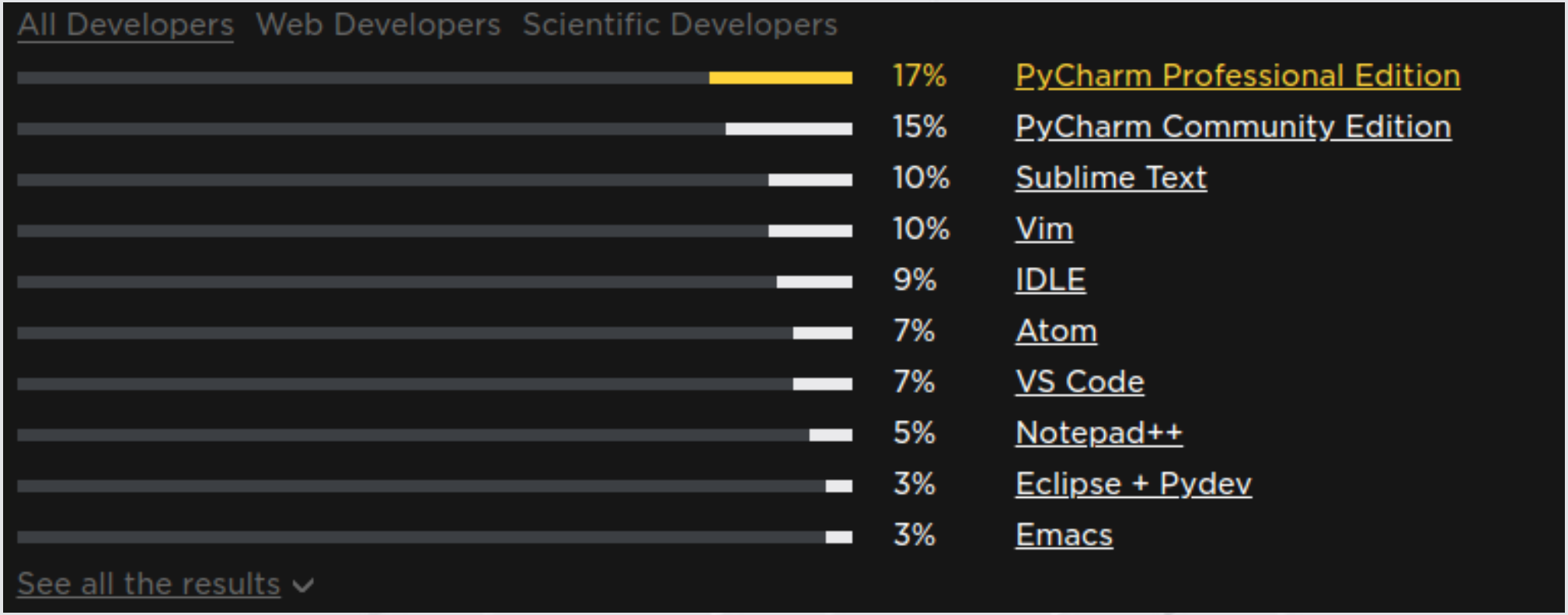
- Primera foto de un agujero negro en la galaxia M87 en la constelación de Virgo.
- Katie Bouman, Ingeniera de 29 años creó el algoritmo para ver el agujero negro.
- Desarrollo hecho en Python https://github.com/achael/eht-imaging



Frameworks

- Django: Framework para desarrollo web del estilo Modelo–vista–controlador
- Flask: Framework para crear aplicaciones web.
- PyGame: Biblioteca que permite la creación de vídeo juegos.
- Scrapy: Web Scrawler framework
- Numpy: Agrega una biblioteca matemáticas de mayor nivel.
- SciPy: Módulos para optimización, álgebra lineal, integración, interpolación, etc.
- SimPy: Simulación de eventos discretos

Entornos de desarrollo



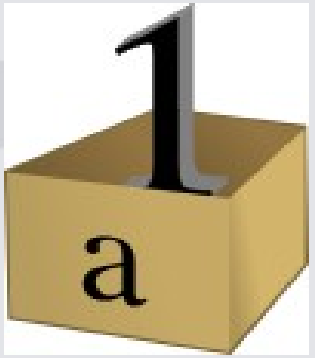
https://www.jetbrains.com/research/python-developers-survey-2017/

Variables

- Otros lenguajes tienen "variables"
- En muchos otros lenguajes, al asignar un valor a una variable se coloca el valor en una especie de caja.

  int a = 1;

- La caja "a" ahora contiene el entero 1.



Variables (2)

- Al asignar otro valor a la misma variable lo que ocurre es que el valor que contiene la caja se sustituye por el nuevo

  a = 2;

- Ahora "a" contiene el entero 2.

Variables (3)

- En Python no hay que declarar las variables ni su tipo.
- Sin embargo, antes de usar una variables es necesario inicializarla
- Al inicializar una variable que no existía, Python deduce su tipo y guarda espacio en memoria para almacenar su valor.
- **Identificadores validos**
  - Debe comenzar con una letra o un guión bajo (_)
  - Luego puede seguir con letras, números y guiones bajos

Variables (4)

- En otros lenguajes al asignar una variable a otra se realiza una copia del valor y se coloca en la nueva caja:

  int b = a;

- "b" es una segunda caja, con una copia del entero 2. La caja "a" tiene una copia totalmente independiente.

Variables (5)

- Python tiene "nombres"
- En Python, un "nombre" o "identificador" es algo parecido a una etiqueta de un paquete de correos adjuntada a un objeto.
- *a = 1*

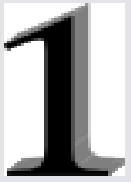
- En este caso, el objeto entero 1 tiene una etiqueta con texto "a".

Variables (6)

- Si reasignamos "a", lo que hacemos es colocar la etiqueta a otro objeto:

  a = 2

  Ahora el nombre "a" está asignado al objeto entero 2.

- El objeto entero original 1 ya no tiene la etiqueta "a". Puede que siga existiendo, pero ya no podemos acceder a través del nombre "a". (Cuando un objeto no tiene más referencias o etiquetas asociadas, se elimina de memoria >> Garbage collector)

Variables (7)

- Si asignamos un nombre a otro, lo único que hacemos es adjuntar otra etiqueta de nombre a un objeto existente:

  b = a

- El nombre "b" es sólo una segunda etiqueta añadida al mismo objeto que "a".
- En Python normalmente usamos el término "variables" pero en realidad nos estamos refiriendo a "nombres". Las "variables" son etiquetas con nombres para los valores, no cajas con nombres.

Variables (8)

- Identificadores validos:
  - mivariable
  - mi_variable
  - var12t
  - Variable
- Python distingue entre mayúsculas y minúsculas
  - mi_variable != Mi_variable
- Por convención, no usaremos identificadores que empiezan con mayúscula y para separar las palabras usaremos el guión bajo.

Variables (8)

- *minusculas_con_guiones* para funciones, métodos, variables, atributos
- TODO_MAYUSCULAS para las constantes
- PalabrasEnMayusculas para las clases
- camelCase sólo si es necesario adaptarse a convenciones que ya se utilizaban en el código

Palabras reservadas

- Python reserva 33 palabras para describir la estructura del programa, y no permite que se usen como identificadores.
- Estas no se pueden utilizar como un identificador de una variable.
- help() y luego *keywords*

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