Curso Intensivo Python - Día 1

Gabriel Valenzuela

@elestudianteclases

Verano 2021

Agenda

- Los temas a ver el día de hoy comprenden:
  - Presentación e introducción al curso
  - Preparar el ambiente de trabajo
  - Entrada y salida por consola
  - Variables: Tipos y operaciones

Presentación e introducción al curso

Gabriel Valenzuela

- Estudiante avanzado de Ingeniería en Computación
- Desarrollador en C++
- Ayduante alumno en UNC (Análisis Matemático II)

Presentación e introducción al curso

Gabriel Valenzuela

- Estudiante avanzado de Ingeniería en Computación
- Desarrollador en C++
- Ayduante alumno en UNC (Análisis Matemático II)

¿Consultas?

- gabriel.valenzuela@mi.unc.edu.ar

Presentación e introducción al curso

Planificación

- La planificación del curso consiste en:

Fecha	Tema
06/03	Conceptos escenciales
13/03	Bucles y condicionales
20/03	Funciones, archivos y proyecto final

Presentación e introducción al curso

Planificación

- La planificación del curso consiste en:

Fecha	Tema
06/03	Conceptos escenciales
13/03	Bucles y condicionales
20/03	Funciones, archivos y proyecto final

¡Nota importante!

Fecha	Tema
06/03	Conceptos escenciales
13/03	Bucles y condicionales
20/03	Funciones, archivos y proyecto final

Python: Introducción - ¿Qué?¿Cuándo?¿Ah?

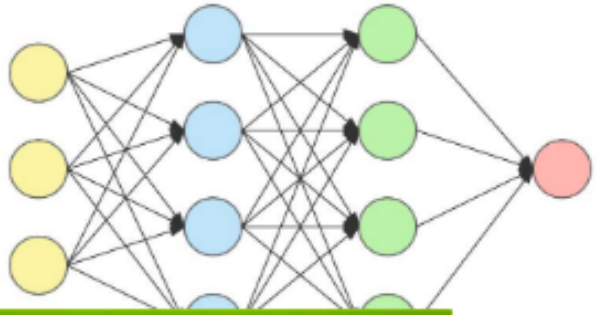
Origenes

- Python es un lenguaje de programación multiparadigma e interpretado que surgió a finales de los años '80 por Guido van Rossum
- Multiparadigma: En la programación un paradigma, al igual que en el lenguaje tradicional es una "guia" para crear código, los dos mas famosos son el orientado a objetos (OO) y funcional (FP) por sus siglas en inglés.
- Interpretado: Un lenguaje puede ser compilado (Ej: C++) o interpretado. Compilado significa que pasa por un proceso previo, la compilación, para poder ejecutar el programa. Interpretado, el intérprete va leyendo en tiempo de ejecución el programa.

$\longrightarrow$ ¿Cuál es la diferencia principal entre lenguaje compilado y ejecutado?

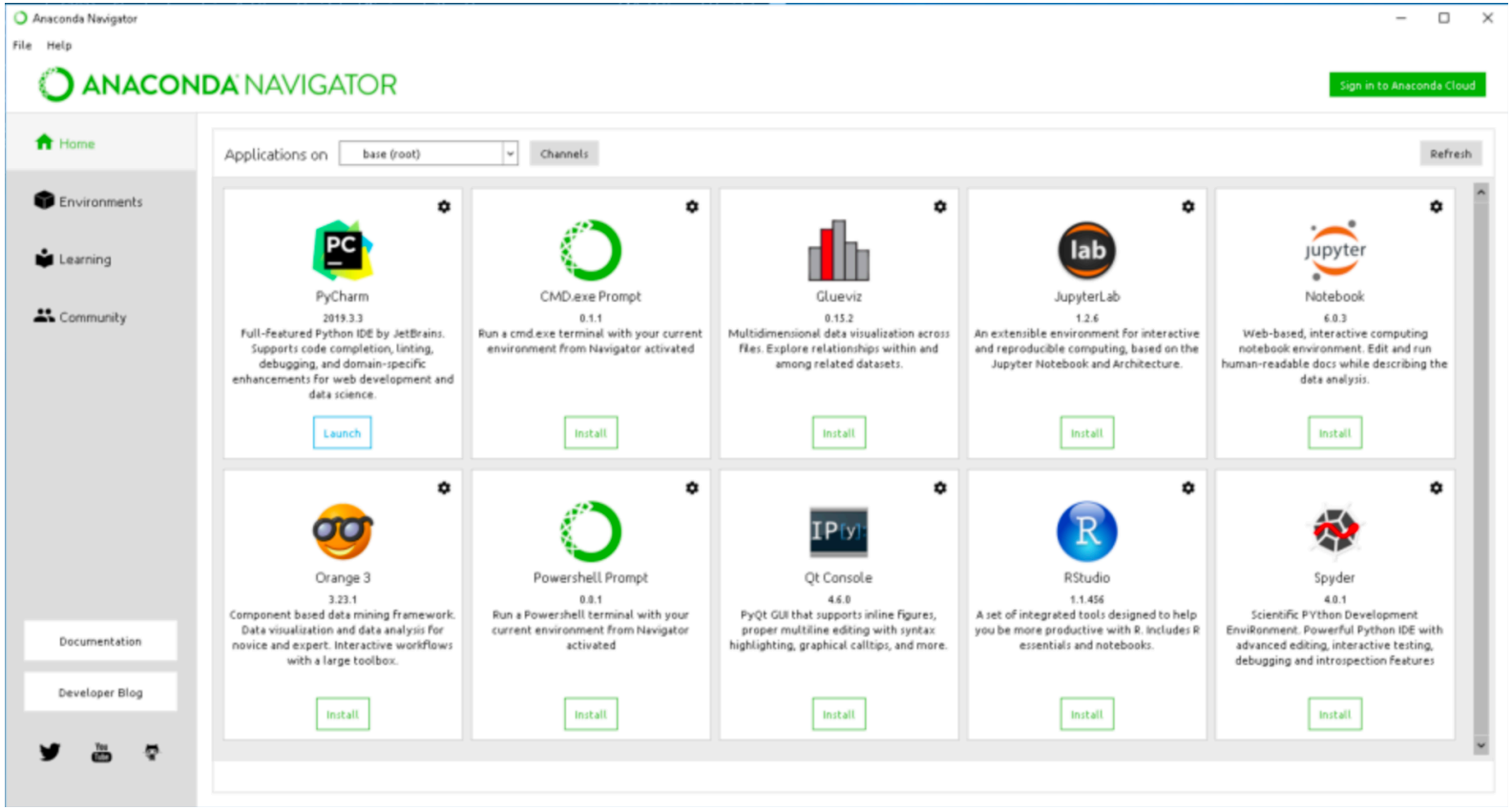
Python: Introducción - ¿Qué?¿Cuándo?¿Ah?

Aplicaciones



Herramientas

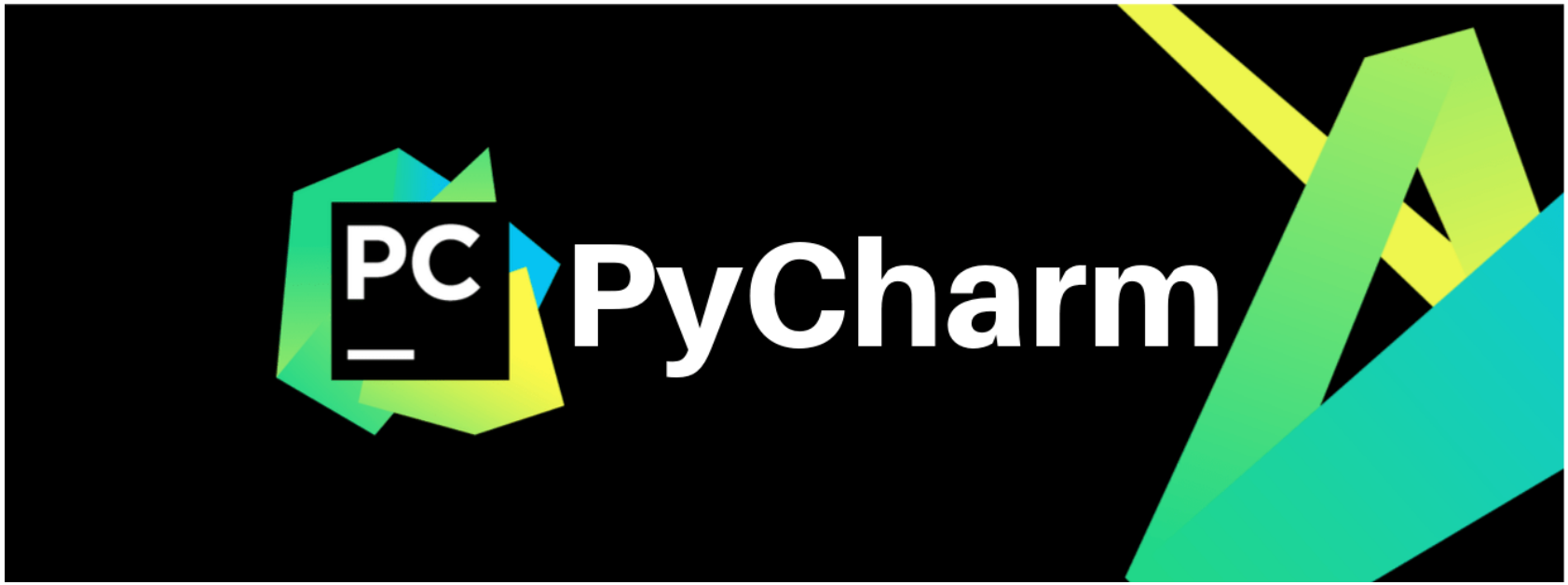
¿Qué vamos a necesitar?



Disponible aquí

Herramientas

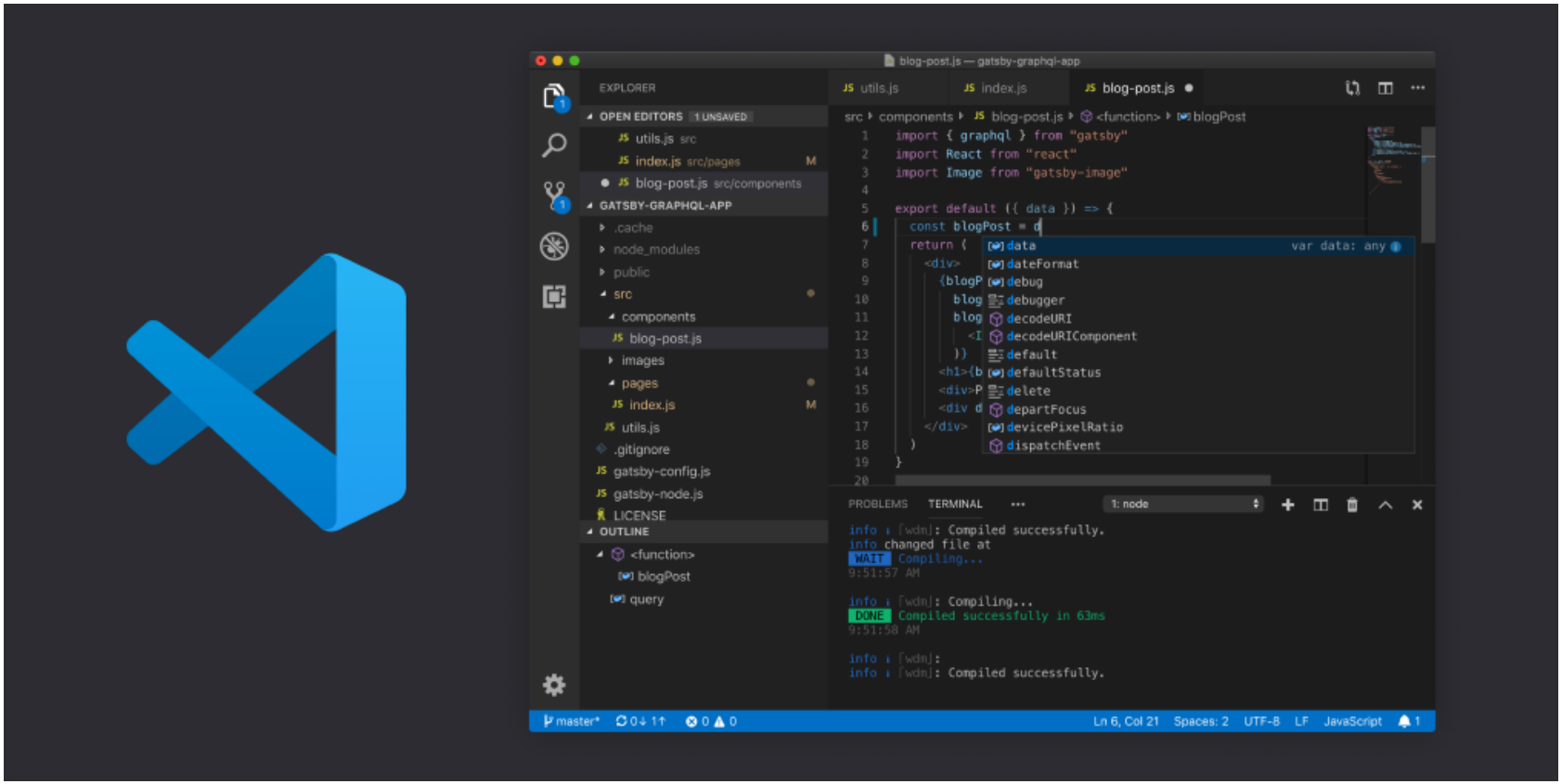
¿Qué vamos a necesitar?



Disponible aquí

Herramientas

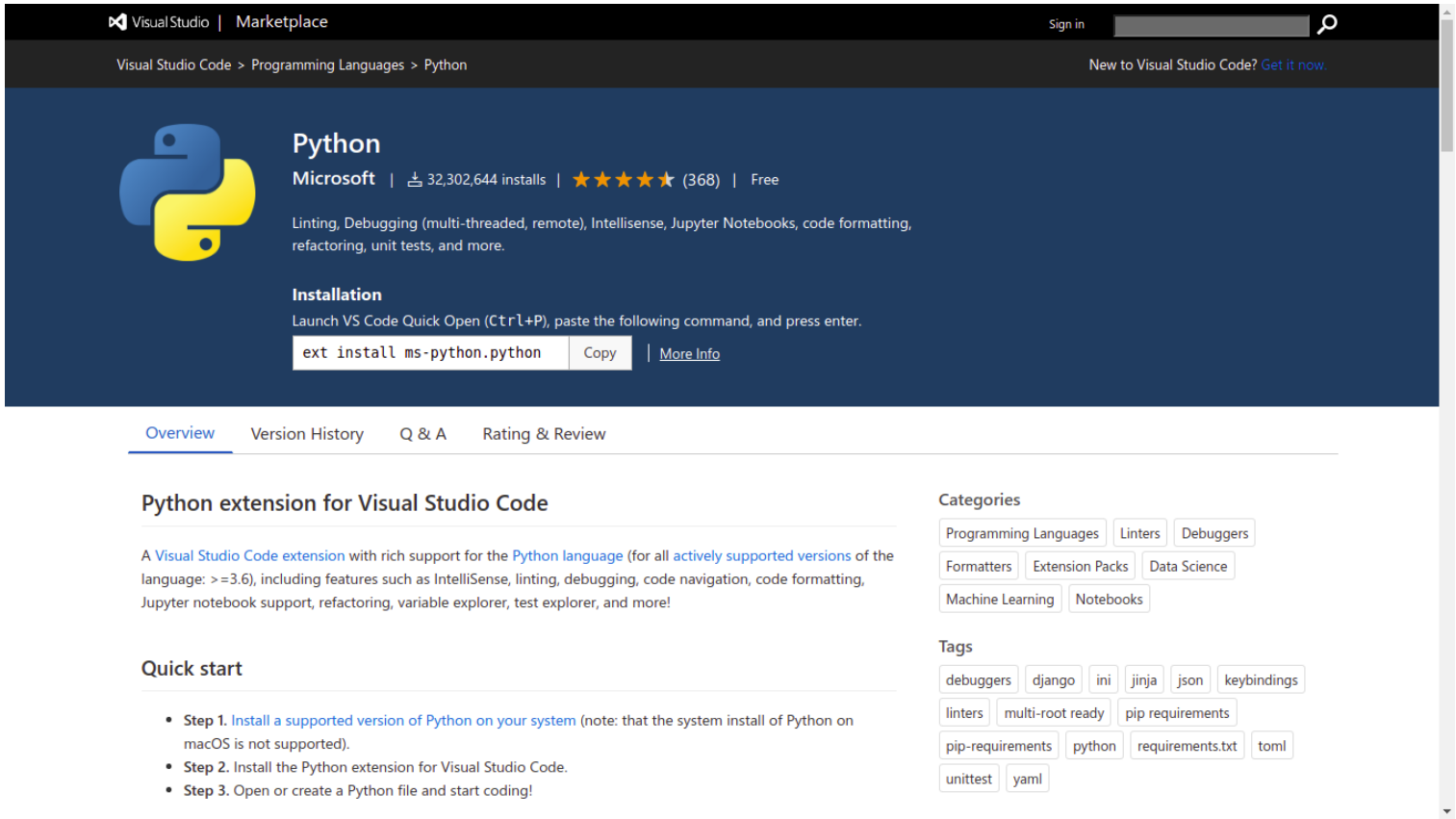
¿Qué vamos a necesitar?



Disponible aquí

Herramientas

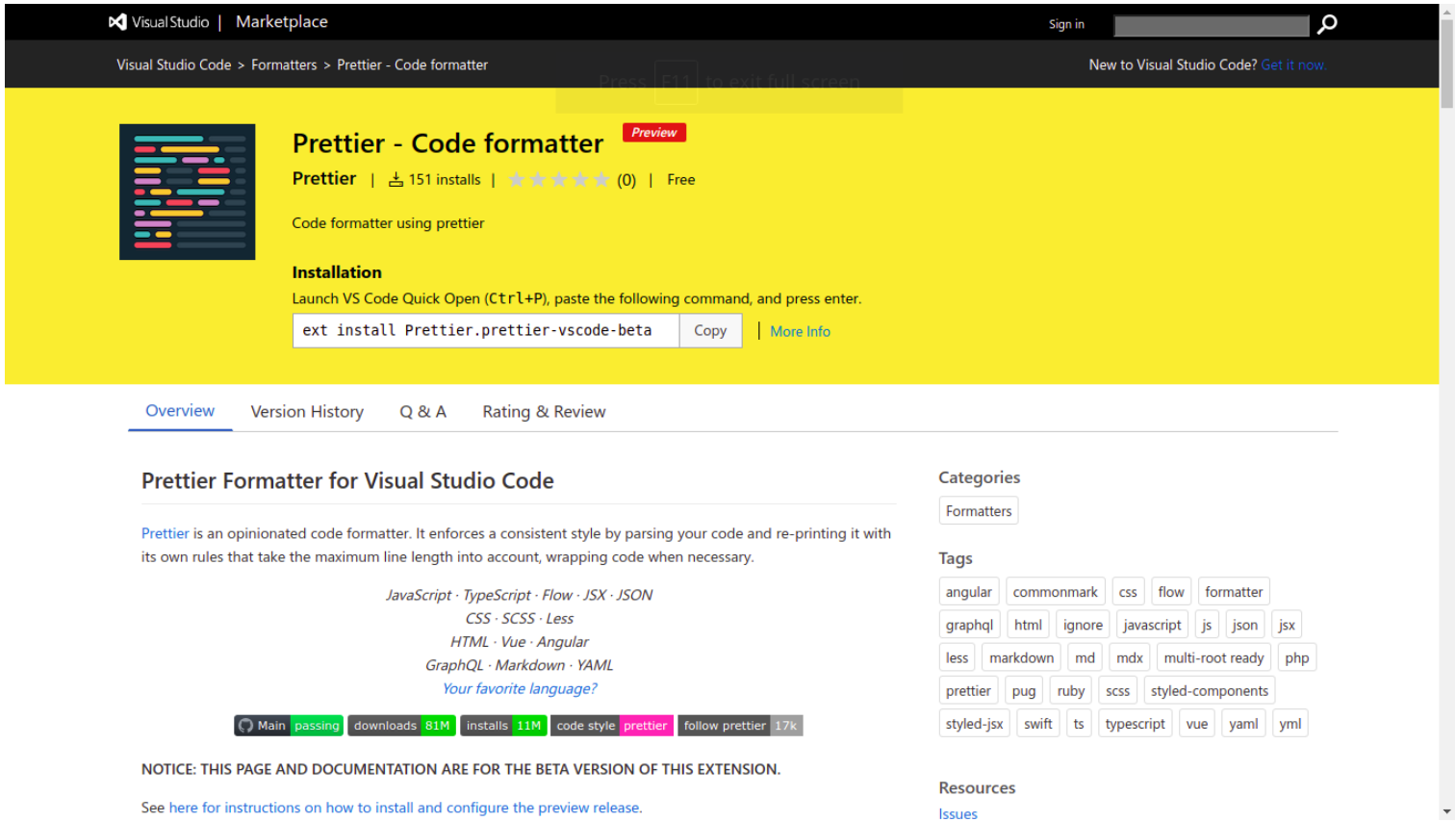
¿Qué podemos añadir?



Disponible aquí

Herramientas

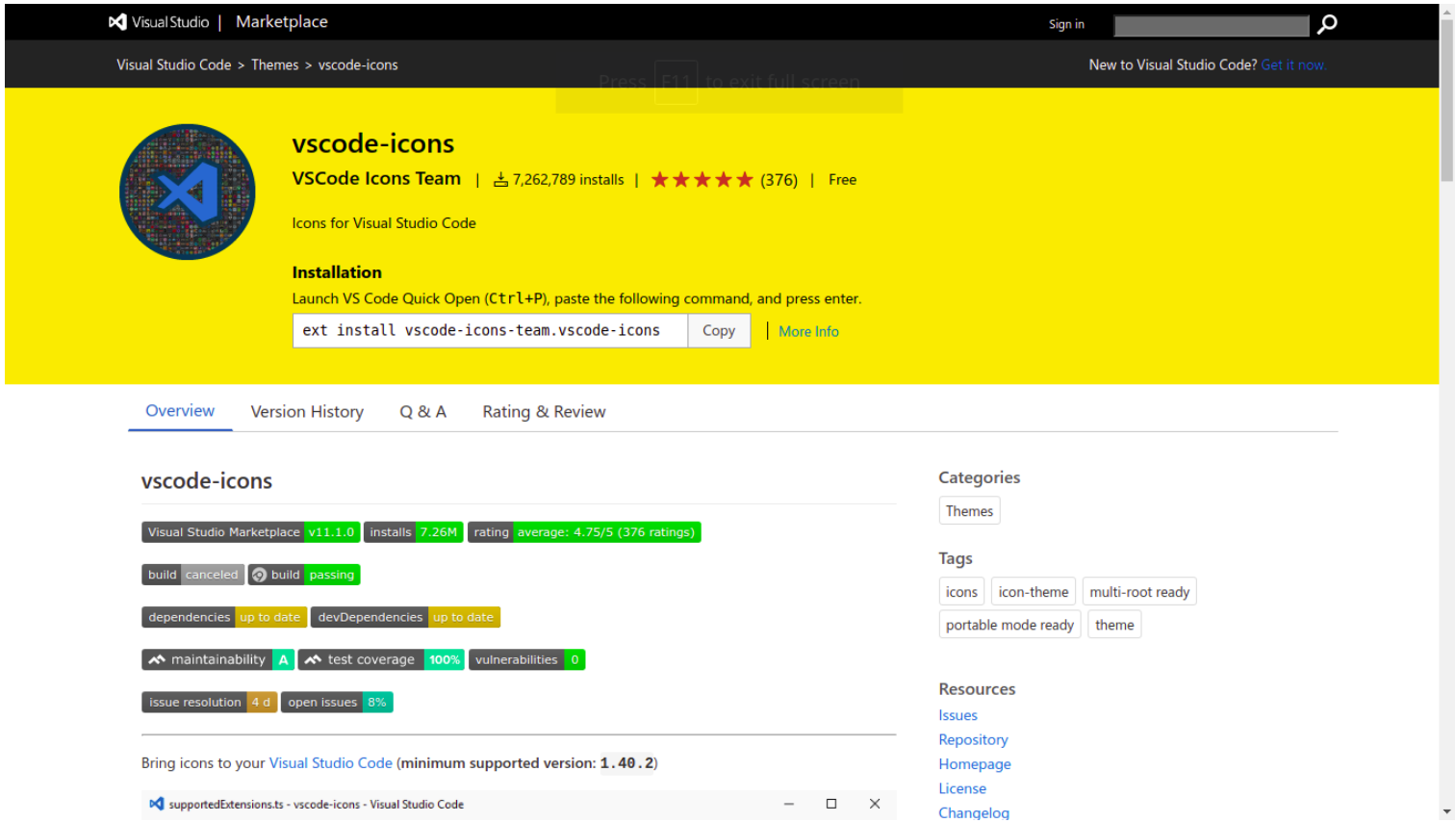
¿Qué podemos añadir?



Disponible aquí

Herramientas

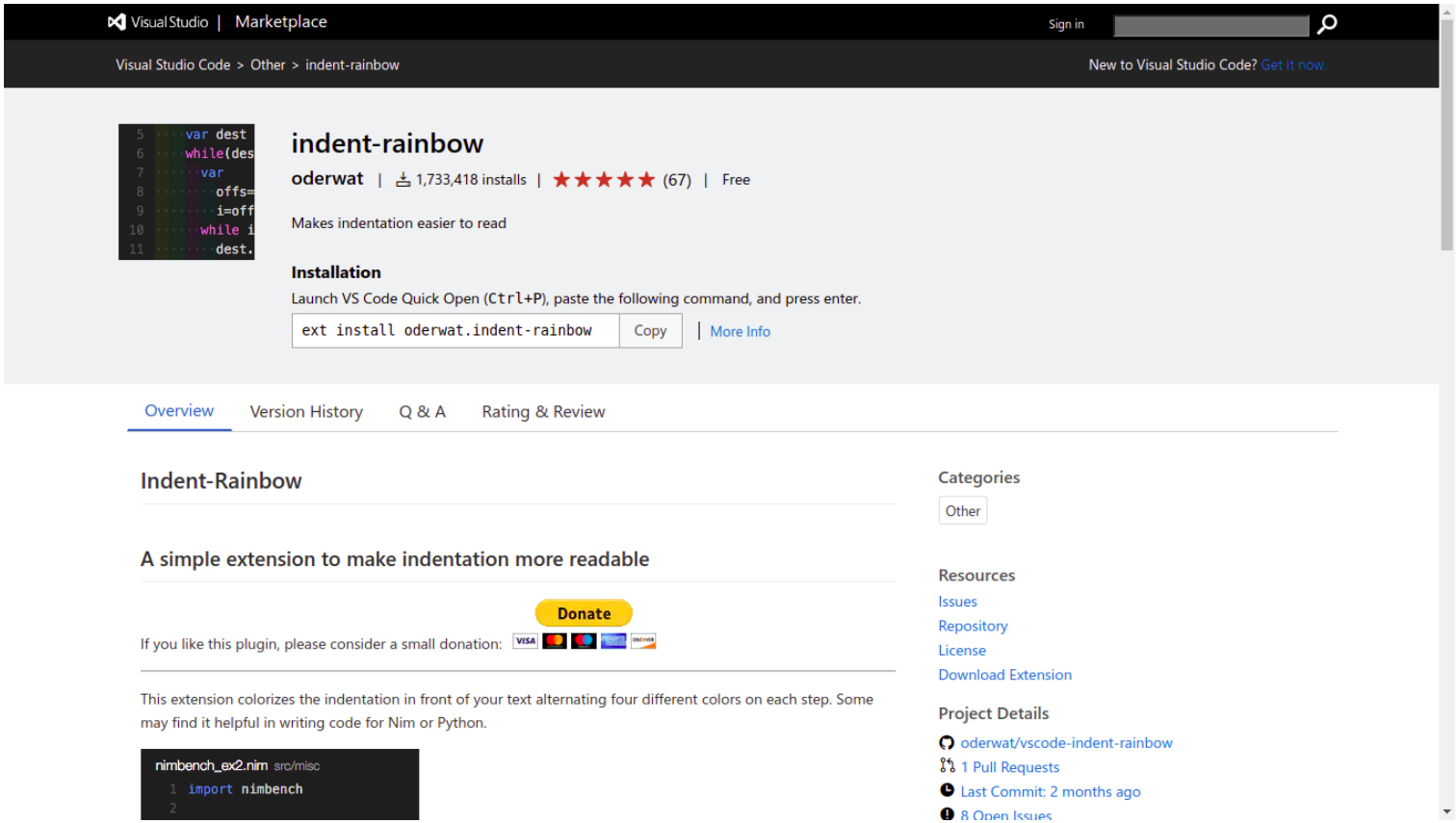
¿Qué podemos añadir?



Disponible aquí

Herramientas

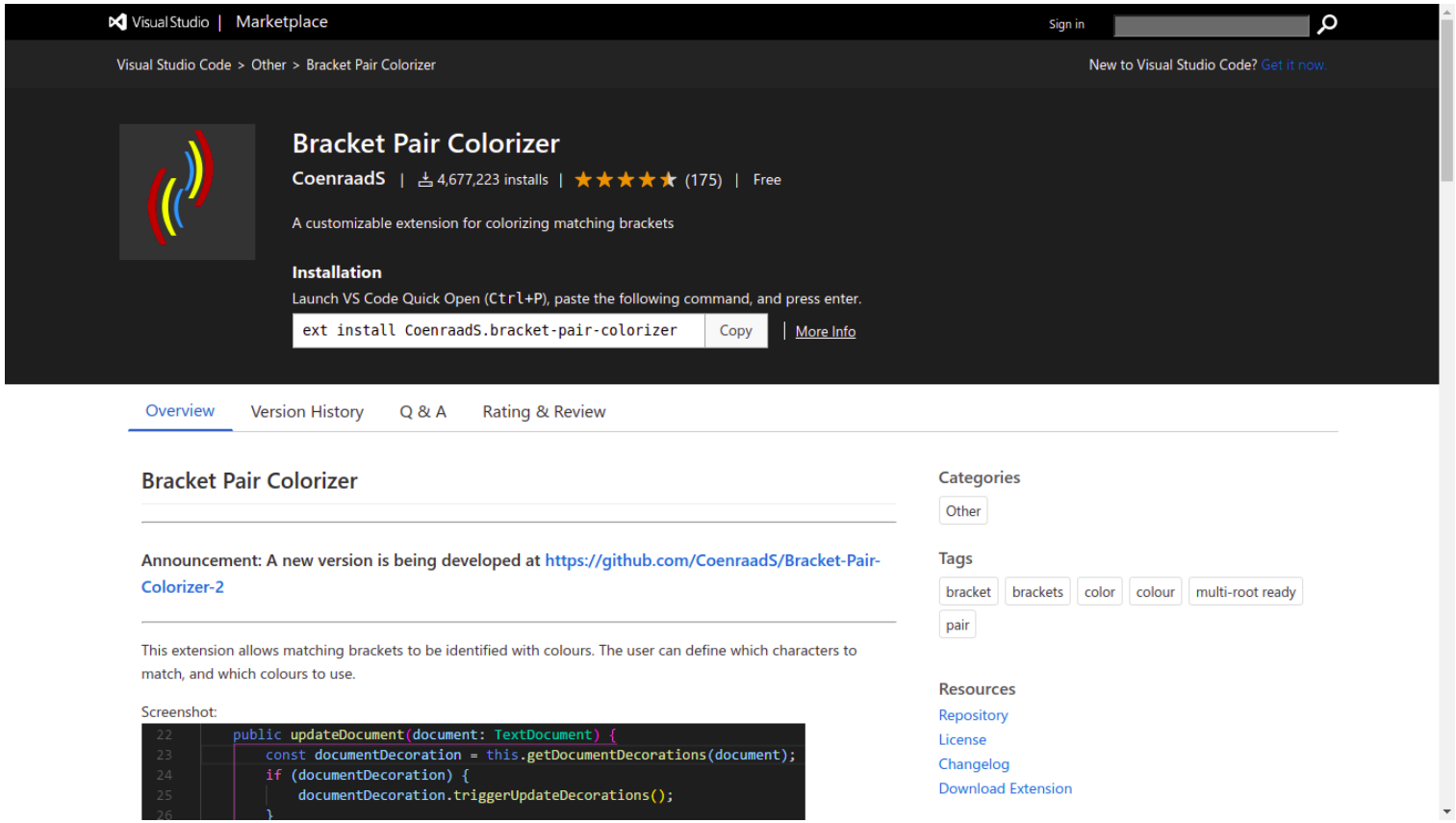
¿Qué podemos añadir?



Disponible aquí

Herramientas

¿Qué podemos añadir?



Disponible aquí

Complejo vs Complicado

El Zen de Python dice:

Simple es mejor que complejo.
Compejo es mejor que complicado.

- Mientras más *complicado* es algo, más difícil de entender
- Mientras más *complejo* es algo, más partes tiene

- Los problemas complicados tienden a no tener soluciones simples
- Pero es posible **modularizar** la situación para tener muchos componentes simples

- Por ejemplo, el uso de funciones y en formas más avanzadas, clases y objetos have que el código sea más *complejo*, pero es todavía fácil de entender

Salida por consola

- Es clásico en la programación hacer *¡Hola Mundo!*
- La operación es sencilla, pero importante ya que representa una interacción con el usuario
- Para desplegar información sin la necesidad de una GUI (Interfaz Gráfica de Usuario)

La función print()

- Se compone de la siguiente forma: **print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)**
- *objects puede ser una cadena, número, lista, diccionario, etc.
- El argumento **sep** indica el caracter que separa las palabras
- El argumento **end** indica el caracter que finaliza las frases.
- Los restantes argumentos vienen por defecto y no los necesitamos

¡Hola mundo!

```
1 #Imprime por consola ¡Hola mundo!
2 #print("¡Hola mundo!")
```

Salida por consola avanzado

- Se compone de la siguiente forma: **format(value[,format_spec])**
- La función format permite convertir un valor a una expresión "formateada", que es controlado por el format_spec.
- La interpretación del format_spec dependerá el tipo de valor de argumento, sin embargo, **existe** una sintaxis estándar de formateo que es el mas comúnmente usado: Disponible aquí

Ejemplo

```
1 #Nuevo fomato de sintaxis que soporta nuevas y diferentes opciones
2 print('{0}, {1}, {2}'.format('a','b','c'))
3 #Se puede acceder por nombre de las variables
4 print('Coordendas: {lat}, {long}'.format(lat='38.23N',long='-114.81O'))
5 #Salida en diferentes representaciones numéricas
6 print('int: {0:d}, hex: {0:x}, oct {0:o}, bin{0:b}'.format(16))
7 #Despliega la cantidad de cifras decimales
8 print('Valor decimal: {:.2%}'.format(1/3))
```

Entrada por consola

- La instrucción para entrada por la consola por parte de usuario consiste en la función **input()**
- **¡Ojo!** input devuelve siempre una cadena, por lo que para tomar un dato entero o flotante se "castea" con las funciones **int()** o **float()**

Ejemplo

```
1 #Toma una cadena o string de entrada
2 cadena = input('Despliego información al usuario: ')
3 #Toma un entero o int de entrada
4 entero = int(input()) #Aqui no se despliega nada
5 #Toma un flotante o float de entrada
6 flotante = float(input('Esto es opcional \n'))
```

Variables

- En Python una variable es un un nombre que se usa para hacer referencia a un valor.
- El nombre de la variable **no debe** ser una palabra reservada de Python.
- La sintaxis de las variables es:
  - nombreVariable = valorVariable
- Python es un lenguaje con tipado dinámico, esto es, las variables pueden cambiar de tipo.
- El *tipo* de la variable es una categoría del valor: **entero,flotante,cadena,lista,diccionario,objeto en general**
- El tipo de una variable limita las operaciones que se pueden llevar a cabo con dichas variables.

Ejemplo

```
1  #Algunas variables
2  variableA = 10
3  variableB = 7.3
4  variableC = 'Cadena'
5  variableD = [1,2,3]
6  variableE = (1,2,3)
7  variableF = {'A':1,'B':2,'C':3}
8  #Imprimos los valores
9  print(variableA)
10 print(variableB)
11 print(variableC)
12 print(variableD)
13 print(variableE)
14 print(variableF)
15 #Verificamos los tipos
16 print(type(variableA))
17 print(type(variableB))
18 print(type(variableC))
19 print(type(variableD))
20 print(type(variableE))
21 print(type(variableF))
```

Operaciones básicas

- Python
  tiene numerosas operaciones que pueden llevar a cabo cálculos matemáticos.

Símbolo	Operación	Descripción
+	Adición	Suma dos números
-	Substracción	Resta dos números
*	Producto	Multiplica dos números
/	División	División convencional, el resultado es un flotante
//	División entera	El resultado de la división es un número entero
%	Modulo	Devuelve el residuo
**	Potencia	Eleva un número a un exponente

Hands On!

¿Preguntas?