لنشغل البرنامج:

|  |
| --- |
| >> python pizza.py |

وسنحصل على المخرج التالي:

|  |
| --- |
| Output  Pizza created: artichoke ($15) Made1 artichoke pizza(s)  Ate1 pizza(s)  Pizza created: margherita ($12) Made2 margherita pizza(s)  Ate1 pizza(s) |

|  |  |  |
| --- | --- | --- |
| برؤية أن البرنامج يعمل، ولكننا نستطيع أن نستخدم وحدة التسجيل لذات الغرض | print | تسمح لنا تعليمات |

بدال.

|  |  |  |
| --- | --- | --- |
| import | printمن الشيفرة البرمجية، ونستورد الوحدة باستخدام األمر | لنقم بإزالة تعليمات |

logging:

|  |
| --- |
| import logging  classPizza():   def \_\_init\_\_(self, name, value): self.name = name   self.value = value  ... |

|  |  |  |
| --- | --- | --- |
| ، وهو مستوى فوق مستوى | المستوى التلقائي للتسجيل في وحدة التسجيل هو مستوى التحذير (WARNING) | |
| بما أننا سنستخدم الوحدة بغرض التنقيح في هذا المثال، سنحتاج الى تعديل إعدادات | | التنقيح (DEBUG). |

|  |  |  |
| --- | --- | --- |
| بحيث تعود معلومات التنقيح لنا من خالل لوحة | logging.DEBUG | التسجيل لتصبح بمستوى التنقيح |

التحكم. ونقوم بإعداد ذلك بإضافة ما يلي بعد تعليماتة االستيراد:

|  |
| --- |
| import logging  logging.basicConfig(level=logging.DEBUG) classPizza():  ... |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10.قيمته | يشير لقيد رقمي | هذا المستوى المتمثل بlogging.DEBUG | | | | |
| print | | | | | سنستبدل اآلن جميع تعليمات | |
| ثابت(logging.DEBUG | | | ، | logging.debug()بتعليمات | | |
| logging.debug()دالة). نستطيع أن نمرر لهذه الدالة نفس المدخالت النصية لتعليماتة | | | | | | بينما |

print كما هو موجود باألسفل:

|  |
| --- |
| import logging  logging.basicConfig(level=logging.DEBUG)  classPizza():   def \_\_init\_\_(self, name, price):   self.name = name   self.price = price   logging.debug("Pizza created: {} (${})".format(self.name,self.price))  def make(self, quantity=1):  logging.debug("Made {} {}  pizza(s)".format(quantity,self.name))  def eat(self, quantity=1): |

|  |
| --- |
| logging.debug("Ate {}  pizza(s)".format(quantity,self.name))  pizza\_01 =Pizza("artichoke",15) pizza\_01.make()  pizza\_01.eat()  pizza\_02 =Pizza("margherita",12) pizza\_02.make(2)  pizza\_02.eat() |

|  |  |
| --- | --- |
| لهذا الحد، نستطيع تشغيل البرنامج عبر تنفيذ األمر | python pizza.py وسنحصل على المخرج |

التالي

|  |
| --- |
| Output  DEBUG:root:Pizza created: artichoke ($15) DEBUG:root:Made1 artichoke pizza(s)  DEBUG:root:Ate1 pizza(s)  DEBUG:root:Pizza created: margherita ($12) DEBUG:root:Made2 margherita pizza(s)  DEBUG:root:Ate1 pizza(s) |

|  |  |  |
| --- | --- | --- |
| الحظ أن مستوى التسجيل في المخرج السابق هو | DEBUGباإلضافة لكلمة root والتي تشير لمستوى | |
| المُسجل (logger) الذي يتم استخدامه. يعني ما سبق أن وحدة التسجيل logging | | من الممكن أن يتم |

استخدامها إلعداد أكثر من مُسجل بأسماء مختلفة.

فمثال، نستطيع إنشاء مسجلين باسمين مختلفين ومخرجات مختلفة كما هو موضح باألسفل

|  |
| --- |
| logger1 = logging.getLogger("module\_1") logger2 = logging.getLogger("module\_2")  logger1.debug("Module 1 debugger") logger2.debug("Module 2 debugger")  Output  DEBUG:module\_1:Module1debugger DEBUG:module\_2:Module2debugger |

|  |  |  |
| --- | --- | --- |
| ،لطباعة الرسائل على وحدة التحكم | بعد أن أصبحت لدينا المعرفة الالزمة لكيفية استخدام الوحدة logging | |
| الرسائل إلى ملف خارجي. | | دعونا نكمل شرح الوحدة ونتعرف على كيفية استخدام الوحدة في طباعة |

**التسجيل في ملف**

الغرض األساسي للتسجيل هو حفظ البيانات في ملف وليس إظهار معلومات التسجيل على وحدة التحكم. يتيح

لك التسجيل في ملف حفظ بيانات التسجيل مع مرور الوقت واستخدامها في عملية التحليل والمتابعة ولتحديد