Python code to create a knowledge base using proportional logic and show that the given query entails the knowledge base or not.

```
def pl_true(sentence, model):
if isinstance(sentence, bool):
return sentence
elif sentence in model:
return model[sentence]
elif isinstance(sentence, tuple):
operator, *args = sentence
if operator == 'NOT':
return not pl_true(args[0], model)
elif operator == 'AND':
return all(pl_true(arg, model) for arg in args)
elif operator == 'OR':
return any(pl_true(arg, model) for arg in args)
elif operator == 'IMPLIES':
antecedent, consequent = args
return not pl_true(antecedent, model) or pl_true(consequent, model)
elif operator == 'IFF':
left, right = args
return pl_true(left, model) == pl_true(right, model)
return False

def tt_entails(kb, alpha):
symbols = get_symbols(kb, alpha)
```

```
return tt_check_all(kb, alpha, symbols, {})


def tt_check_all(kb, alpha, symbols, model):
if not symbols:
if pl_true(kb, model):
return pl_true(alpha, model)
else:
return True
else:
p, rest = symbols[0], symbols[1:]
model_true = model.copy()
model_true[p] = True
model_false = model.copy()
model_false[p] = False
return tt_check_all(kb, alpha, rest, model_true) and tt_check_all(kb, alpha,
rest, model_false)


def get_symbols(*sentences):
symbols = set()
for sentence in sentences:
collect_symbols(sentence, symbols)
return list(symbols)
```

```
def collect_symbols(sentence, symbols):
if isinstance(sentence, str) and sentence.isalpha():
symbols.add(sentence)
elif isinstance(sentence, tuple):
for arg in sentence[1:]:
collect_symbols(arg, symbols)

# User input for knowledge base and query
print("Enter the knowledge base (KB) as a tuple expression (e.g., ('AND', ('IMPLIES', 'P', 'Q'), 'P')):")
kb = eval(input("KB: "))

print("Enter the query (alpha) as a tuple expression or a symbol (e.g., 'Q'):")
alpha = eval(input("alpha: "))

# Check if KB entails alpha
result = tt_entails(kb, alpha)
print("Does KB entail alpha?", result)
```

Output:

Enter the knowledge base (KB) as a tuple expression (e.g., ('AND', ('IMPLIES', 'P', 'Q'), 'P')):

KB: (('A','OR','C'),'AND',('B','OR','NOT','C'))

Enter the query (alpha) as a tuple expression or a symbol (e.g., 'Q'):

alpha: ('A','OR','B')

Does KB entail alpha? True