PerfectPup🐾

Updated Description of Project

Project Description: PerfectPup is a website that helps individuals and families find the ideal dog breed that is best suited for their unique needs, lifestyle, and preferences. Whether you are looking for a loyal companion, a family-friendly pet, a guard dog, or a furry friend, PerfectPup will find the perfect match for you.

Project Goals: Selecting which dog breed to choose is a top question that all prospective dog owners should ask. Users can input information about their needs and preferences such as a dog's activity level, temperament, trainability, size, and grooming frequency. PerfectPup will then analyze the user's response and provide them with the exact match dog breeds whose characteristics best match their preferences. It will return a ranked ordering of the dog breeds based on how well they match, up to 10 dog breeds. As a stretch goal, for each dog breed suggested, it will also provide a detailed description of the breed, including its temperament, activity level, trainability, and other important information.

Broad Categories: Pets

Information Retrieval Aspect: We will be using a dataset that was extracted from the American Kennel Club website that contains information on the characteristics of 277 dog breeds. For each of the dog breeds, the dataset contains information on 20 features pertaining to the breed such as its description, temperament, popularity, height, weight, grooming frequency, etc.

Social Information Aspect: Dataset includes subjective "reviews" and characteristics of our puppers, including personality traits, popularity, etc.

With our TA, we discussed our new project proposal, pivoting away from trying to generate a travel itinerary, to instead generating the best dog breed for a prospective dog owner. This shift in project was due to feasibility as exploring the travel itinerary generator made us realize that we would need to utilize, parse, and extract information from over 5 datasets as we could not find a comprehensive dataset that encompasses all restaurants, tourist attractions, hotel stays, modes of transportation, and their corresponding prices/location for various travel destinations. Due to the time constraint of the class project, we shifted our focus on a project idea with a more niche topic so that we can narrow it down to one dataset. After brainstorming various potential projects, we all came into agreement of implementing the dog breed recommender project and we confirmed this with our TA who supported the idea. We also discussed how we'd include a social information aspect, how we should approach data exploration, and the literature review.

To evaluate our application and the accuracy of our recommendations, we can manually review the recommendations provided by PerfectPup and compare the characteristics of our recommendations to the user's stated preferences. In addition, another method that we could potentially use is to compare our recommendations with that of existing systems to evaluate the reliability of our application. We can also conduct user testing by letting a representative population use our application, and give feedback on whether or not the breeds output have traits that they were looking for.

Lit review: Currently, there exists many options for a prospective pet owner seeking to determine which dog breed they want, ranging from blogs with long descriptions of popular dog breeds, quizzes, social media posts. The options offered by many famous pet/dog organizations include long blogs with descriptions of popular dog breeds, and quizzes, such as Purina, IAMS, and even one offered by the organization that is providing our dataset, AKC. However, the quizzes do not allow for open ended responses, and do not account for descriptions of the dogs and weigh in personality preferences. This limits the possible characteristics the user may want for their furry companion. Our approach would instead be open ended, and allow for a much more specific and faster, personalized selection of dog breed.

Sketched examples and descriptions of idealized inputs and outputs

Overview

For our user inputs, we will first allow them a free form text input to describe their ideal dog with any characteristics they want. This query will take into consideration the temperament, description, and demeanor for each dog breed. Our next input will be how much time does the user have on average daily to dedicate to taking care of their pet? We will provide them with a slider where they can pick how many hours. This query will take into consideration the grooming and shedding frequency, energy level, and trainability. Our last input will be asking how big their space is, and will give them three options: Small, Medium, Big. This query will take into consideration height/weight, energy level of the breeds. For an optional advanced search/stretch goal, we will ask if the user has any preferences for Life expectancy, weight/height, Group (9 breed groups, allergens/shedding, good with children(this will mostly be a direct string-matching query and not referenced in our examples). Our output will be the top matching dog breeds and a short explanation of why these dog breeds were chosen.

Examples

Example 1: (valid input and good output)

- User input:
    - Free Form Text: *I want a dog that is fearless and fun-loving that doesn't require much attention.*
    - Hours: *4*

    - Space: *Medium*
- System output: **Miniature Pinscher, Caucasian Shepherd Dog, Kuvasz**
- Output rating: **Good**
- Explanation: This output is good because it returns dog breeds that match those personality traits, but only those that are categorized as breeds that do not need much attention or exercise, and are of small or medium build. The miniature Pinscher fits this criteria perfectly because it only needs occasional or rare grooming, rarely sheds, is small, and doesn't need frequent exercise. The hours input is not small enough to where we'd need to necessarily narrow down the scope to dog breeds that only characterize as calm. An example of dog breed that almost but doesn't quite match would be the Bullmastiff, that while Bullmastiff personality traits closely match our personality query, and their medium build also satisfies the space query, they are highly energetic and require a lot of exercise which would not be a good fit since the user input only 4 hours. While the Weimaraner fits the personality traits almost perfectly (friendly and fearless), they are larger in size, and require lots of attention so our system would not output this breed.

Example 2: (valid input and bad output)
- User input:
    - Free From Text: *I want a small dog that is obedient*
    - Hours: *2 hours*
    - Space: *small*
- System output: **Perro de Presa Canario**
  Other valid outputs but don't fit because their energy level is too high: Miniature Schnauzer, Shetland Sheepdog, Weimaraner, Beauceron, English Springer Spaniel, Slovakian Wirehaired Pointer, Stabyhoun
- Output rating: **Bad**
- Explanation: This is a bad example because it only results in one dog, it would be good in our design to include some results that may have missed the query and why, as a result the user can see if they're flexible with one of their criteria, what the other options are

Example 3: (valid input and bad output)
- User input:
    - Free Form Text: *I want a social and small dog that I can hang out with at home.*
    - Hours: *6 hours*
    - Space: *small*
- System output: **Brussels Griffon**
- Output rating: **Bad**
- Discuss why these outputs are good examples are good and why the bad outputs are bad, how do they (not) satisfy the user's information need?

- Explanation: This output is a bad example because it returns only one dog breed that matches the user's query while there are also 2 other dog breeds (Havanese and Russian Tsvetnaya Bolonka) that also match the user's input really well. Outputting only one of the best matches gives the user less options than they really have.

Engineering Challenges:
Given the queries for personality, hours, and space, we will map these inputs to their corresponding related columns in our database. The free form text query will primarily take into account the temperament, description, and demeanor for each dog breed, but also any other characteristics that they input. We will use a similarity measurement such as cosine similarity with inverted index on our input and the data column that says "description" (a paragraph description about the dog (as a stretch goal, if we feel our results could be better we may web scrape descriptions about each dog breed to use with cosine similarity). We will then use the boolean AND search to connect the description results with the advanced search parameters. Next, for hours, we will use a mapping from the hours they input to a point range. The three factors - trainability, energy level, and shedding frequency will each have points relating to them and adding them up will match them with the number of hours owners have available. Finally, for the space we will take in the height and weight and energy level to match it with the appropriate space, doing the same with the point system. The challenge will be finetuning this algorithm and ensuring the boolean AND works for the entire query.

Weekly Schedule

Experience
Amy has full-stack experience with the MERN stack.
Louis has full-stack experience with the MERN stack, Postman, Flask, and SQLite.
Hannah has full stack experience with the MERN stack and Postman.
Daniel has full-stack experience with the MERN stack and Postman.
Connie has frontend experience and UI/UX design expertise to make cool designs and interactions

Schedule

Week	Amy	Louis	Dan	Hannah	Connie	Team Goal
3/19 - 3/25	• Wire input with the results	• Figure out processing for "temperament"	• Figure out processing for "description"	• Wire input with the results	• Implement input and output mvp	(1) Produce a basic prototype of the app with a focus on

	• Determine how data will be parsed; generate good_types; how good input will be differentiated from bad input • Complete write-up	• Implement information retrieval - space and time calculations + queries- techniques to generate desired output • Complete write-up	• Implement information retrieval - boolean AND - techniques to generate desired output • Complete write-up	• Implement information retrieval - advanced search queries - techniques to generate desired output • Complete slides	• Finetune cosine similarity and figure out how similarity percent calculated • Complete slides	information retrieval techniques and data manipulation (2) Will have a very basic UI for input (3) Will produce an output that the team can analyze (4) Complete the write-up and slides for the PO3 milestone
3/26 - 3/31	• Based on TA feedback, further develop backend • Work on fine-tuning relevant query results and how this translate to our dataset, iterate on the description personality cosine matching	• Based on TA feedback, further develop backend • Work on fine-tuning relevant query results and how this translate to our dataset, iterate on the description personality cosine matching	• Based on TA feedback, further develop backend • Improve the current information retrieval system, explore other means of processing data that goes beyond the basic method done for the first prototype	• Based on TA feedback, further develop backend • Improve the current information retrieval system, explore other means of processing data that goes beyond the basic method done for the first prototype	• Based on TA feedback, further develop frontend • Utilize libraries like Material UI to make React website more dynamic and presentable	(1) After presenting to TA, address blockages and feedback to improve the PO2 prototype (2) Present project in class
4/1 - 4/9	Spring break	Spring break	Spring break	Spring break	Spring break	Get well deserved rest!
4/10 - 4/16	• Finalize input text processing • Address any	• Finalize feature mapping for all breeds • Address any	• Finalize input text processing • Address any	• Finalize feature mapping for all breeds	• Implement the results page that provides the users more	(1) Meet with TA to discuss plans/blockages for second prototype

	backend blockages after discussing with TA	backend blockages after discussing with TA	backend blockages after discussing with TA	• Address any backend blockages after discussing with TA	information about the dog breeds they were matched with • Explore card designs and implementations	(2) Implement a functional version of the app (3) Have a cleaner app that looks close to final submission
4/17 - 4/23	• Finalize IR system and fully optimize its performance to account for all edge cases	• Finalize IR system and fully optimize its performance to account for all edge cases	• Wire backend and frontend, help with the backend processing for backend dataviz	• Wire backend and frontend, help with the backend processing for backend dataviz	• Finalize frontend - implement dataviz stretch goal to provide transparency on results	(1) Based on second prototype feedback, implement necessary changes towards submitting the final app
4/24 - 5/1	• Finalize app, contribute where help is needed • Write report	• Finalize app, contribute where help is needed • Write report	• Finalize app, contribute where help is needed • Write report	• Finalize app, contribute where help is needed • Create slides	• Finalize app, contribute where help is needed • Create slides	(1) Complete work needed to submit the final app
5/2 - Final Deadline	• Complete write-up	• Complete write-up	• Complete write-up	• Complete write-up	• Complete write-up	(1) Submit written report

PerfectPup, CS 4300 (Language and Information)

Dog Breed Data Set Exploration

Since we are creating a information retrieval system that will allow users to query dog characteristics, personalities, specificities, etc., and return a list of the most relevant dog breeds, we are choosing to use data that was harvested from the American Kennel Club website. A registry of purebred dog pedigrees (which happens to promote/sanction the Westminster Kennel Club Dog Show, one of, if not, the most famous and oldest dog shows!), that describes various characteristics and data for 277 different dog breeds. There are roughly 340 to 360 recognized dog breeds in total, so this collection of 277 breeds represents, on the lower bound ~77%, and on the upperbound ~61%, of all dog breeds, so there is a healthy representation. Because of this, we do not plan on acquiring more data, as missing breeds could potentially be ones that are not universally recognized or rare/niche. This dataset includes information such as description, height, weight, personality, popularity, and a lot more, which will allow users to query on a rich and expressive set of parameters they could be looking for in a dog breed. We will be exploring this data by finding the distribution and analyzing some of the statistics for a variety of these characteristics. At this time we do not plan to use a supervised ML approach. All credit goes to Telmo Filho, of the University of Bristol, for scraping this data off the AKC website using BeautifulSoup.

Imports and Initial Observations

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt

dog_data = pd.read_csv("akc-data-latest.csv")

# Data Cleaning
dog_data['popularity'] = dog_data['popularity'].fillna(1000)
dog_data['popularity'] = dog_data['popularity'].replace("nf", 1000)
```

Our data is in the .csv format, so let's see how it looks like as a "spreadsheet". Note, the column "Unnamed: 0" is the name of our dog breeds.

In [2]:
```
dog_data.head()
```

Out[2]:

	Unnamed: 0	description	temperament	popularity	min_height	max_height	min_weight	max_weight	min_expectancy	max_expectancy	...	grooming_frequency_value	grooming_frequency_category	shedding_value	shedding_category	energy_level_value	energy_level_categ
0	Affenpinscher	The Affen's worn look has been described as unkempt.	Confident, Curious, Funloving, Fearless	149	22.86	29.21	3.175147	4.535924	12.0	15.0	...	0.6	2-3 Times a Week Brushing	0.6	Seasonal	0.6	Regular Exe
1	Afghan Hound	The Afghan Hound is an ancient breed of sighthound.	Dignified, Profoundly Loyal, Aristocratic	113	63.50	68.58	22.679619	27.215542	12.0	15.0	...	0.8	Daily Brushing	0.2	Infrequent	0.8	Exer
2	Airedale Terrier	The Airedale Terrier is the largest of all ter...	Friendly, Clever, Courageous	60	58.42	58.42	22.679619	31.751466	11.0	14.0	...	0.6	2-3 Times a Week Brushing	0.4	Occasional	0.6	Regular Exe
3	Akita	Akitas are burly, substantial, boned spitz-type dogs. ...	Courageous, Confident, Profoundly Loyal	47	66.96	71.12	31.751466	58.967008	10.0	13.0	...	0.8	Daily Brushing	0.6	Seasonal	0.8	Exer
4	Alaskan Malamute	The Alaskan Malamute stands 23 to 25 inches at...	Affectionate, Loyal, Playful	58	58.42	63.50	34.019428	38.555351	10.0	14.0	...	0.6	2-3 Times a Week Brushing	0.6	Seasonal	0.8	Exer

5 rows × 21 columns

From this, we can see that the data we have to play with is the dog breed name itself, a description, temperament / personality traits, popularity, height, weight, life expectancy, grooomy frequency, shedding frequency, energy levels, trainability, and demeanor. This gives us quite a rich data set that includes almost every factor that one would consider when trying to choose a dog breed, giving us a lot of power how we analyze the users query when determining our most relevant results.

Let's confirm that there are 277 breeds.

In [3]:
```
print("There are " + str(len(dog_data["Unnamed: 0"])) + " different dog breeds in this data set.")
```

```
There are 277 different dog breeds in this data set.
```

Let's take a look at the top 10 most popular breeds!

In [4]:
```
breed_popularity = []
for idx in range(277):
    breed_popularity.append((float(dog_data["popularity"][idx]),dog_data["Unnamed: 0"][idx]))

breed_popularity = sorted(breed_popularity, key= lambda bd: bd[0])

for idx in range(12):
    print(breed_popularity[idx][1] + " " + str(breed_popularity[idx][0]))
```

```
Labrador Retriever 1.0
German Shepherd Dog 2.0
Golden Retriever 3.0
French Bulldog 4.0
Beagle 5.0
Beagle 6.0
Poodle (Miniature) 7.0
Poodle (Standard) 7.0
Poodle (Toy) 7.0
Rottweiler 8.0
German Shorthaired Pointer 9.0
Yorkshire Terrier 10.0
```

Unfortunately, while the AKC has information on many dog breeds, they only recognize 200 dog breeds (as purebred) as of 2023, with only 195 recognized in 2020 when this data was harvested. For some reason, (what I'm assuming to be scraping errors), some recognized dogs in 2020 have a popularity value of "nf", which means they also cannot be ranked. As a result, many dog breeds cannot be assigned a popularity score. It may be possible to reconstruct this list in our information retrieval system, but this is not necessary to be more up to date and accurate, however. From this list we can see that many of the breeds that are universally recognizable, such as Poodles, Golden Retrievers, and Bulldogs, are present on this list! Popularity is an interesting social measure, and this top ten seems to line up with what I would expect with purebred dogs.

For fun, let's take a look at the top 10 least popular breeds as well.

In [5]:
```
for idx in range(185,195):
    print(breed_popularity[idx][1] + " " + str(breed_popularity[idx][0]))
```

```
Cirneco dell'Etna 183.0
Finnish Spitz 184.0
Gesky Terrier 185.0
Pharaoh Hound 186.0
Bergamasco Sheepdog 187.0
English Foxhound 188.0
Harrier 188.0
Chinook 190.0
Norwegian Lundehund 191.0
Sloughi 192.0
```

Personally, I have never heard of these breeds, so this is probably accurate.

With that out of the way, let's start to do some real analysis of this data.

Distributions of a Variety of Characteristics

Something important that we should do is analyze how well our data is distributed across values. A weak distribution would imply a large similarity across all our breeds. In this case, it would not be a very interesting or useful data set, and therefore project, if all the dog breeds we were dealing with were practically the same. This will give us a good idea of how many different possibilities and combinations of traits there are and how expressive this data set can be against a wide range of queries.

Demeanor Analysis

From our data set, demeanor is defined as "A categorization of reaction to strangers and other pets", which seems like a pretty useful metric to include. Some people have children, like to take their dog out and around, or have frequent guests, so knowing how a dog breed typical reacts around others is an important characteristic. Let's see what different kinds of demeanors AKC thinks dogs can have.

In [6]:
```
demeanors = list(set(list(dog_data["demeanor_category"])))
print(demeanors)
```

```
[nan, 'Friendly', 'Outgoing', 'Alert/Responsive', 'Reserved with Strangers', 'Aloof/Wary']
```

It looks like dogs can be "Alot/Wary", "Friendly", "Alert/Responsive", "Reserved with Strangers", and "Outgoing" based on some kind of a 1-5 scale. Considering this is a hard characteristic to objectively measure, the AKC does a good job creating 5 different distinctions such that one can easily see the difference between an "Outgoing" dog breed and one that is "Reserved with Strangers". Unfortunately, once again, it seems like due to data collection issues, some dog breeds were not assigned a demeanor category, which may or may not pose an issue in the final product. Let's see how many dog breeds this will be an issue for.

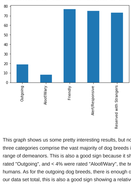
In [7]:
```
nan_counts = ["nan" for demeanor in list(dog_data["demeanor_category"]) if pd.isna(demeanor)]
print("There are " + str(len(nan_counts)) + " dog breed with NaN values for their demeanor category.")
```

```
There are 25 dog breed with NaN values for their demeanor category.
```

For most dogs, this missing value will not be an issue. For the 25 breeds who are missing it, many, if not all of, the values in their other columns of the csv are not empty, which will still give us plenty of characteristics that we can query against, even if it is incomplete. Let's analyze the distribution now.

In [8]:
```
demeanor_counts = [demeanor for demeanor in list(dog_data["demeanor_category"]) if demeanor != "nan"]
pd.Series(demeanor_counts).value_counts(sort=False).plot(kind='bar')
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffc83208b50>
```



This graph shows us some pretty interesting results, but none unexpected. Among the demeanor categories "Friendly", "Alert/Responsive", and "Reserved with Strangers", there was a roughly equal frequency of them at around ~70-80 dog breeds in each category. These three categories comprise the vast majority of dog breeds in our data set. Although our graph is un-ordered, these three categories also represent the three less extreme demeanor categories on both ends, so it makes sense that most dog breeds would fall in the milder range of demeanors. This is also a good sign because it shows that our data is distributed relatively well across the mean, allowing for many options among the most common / less extreme demeanors, which most people are probably seeking. About 7% of dogs were rated "Outgoing", and ~4% were rated "Alot/Wary", the two extremes of our demeanor categories. Intuitively, this makes sense, as dogs are considered "man's best friend", so it would be odd if there was a significant amount of them that did not play nice with other humans. As for the outgoing dog breeds, there is enough of them that provide a good range of options while maintaing their own category of hyper-friendliness for those who seek a dog breed of that kind. Since these two extremes represent < 10% of the dog breeds in our data set total, this is also a good sign showing a relatively even distribution.

Let's now try to analyze the distribution of some characteristics that are not as discrete or subjective as demeanor...

Weight Analysis

Our data set contains the `max_weight` and `min_weight` in kilograms for our set of 277 dog breeds. Like demeanor, this is also a very important attribute to consider because of the many deciding factors there are when choosing a dog breed, size is definitely a deal breaker. Weight is a generally good metric for dog size as larger dogs will simply weigh more, and the ranges of dog weights is very wide. Let's see what the range is in our data set.

In [9]:
```
min_weights = list(dog_data["min_weight"])
min_weights = [weight for weight in min_weights if weight != 0.0]
min_weights = [weight for weight in min_weights if pd.notna(weight)]
max_weights = list(dog_data["max_weight"])
max_weights = [weight for weight in max_weights if pd.notna(weight)]

print("The lightest dog in our data set can have a minimum weight of " + str(min(min_weights)) + " kilograms and the heaviest dog in our data set can have a maximum weight of " + str(max(max_weights)) + "kilograms.")
```

```
The lightest dog in our data set can have a minimum weight of 1.3607711 and the heaviest dog in our data set can have a maximum weight of 68.0388555
```

As we can see, dog weights vary extremely, with our lightest small dog breed weighing ~1.36 kg and our heaviest heavy dog breed weighing ~68 kg. Again, due to data collection issues some dogs either have no weights or a weight of 0. Let's see how many are impacted.

In [10]:
```
err_min_weights = list(dog_data["min_weight"])
err_min_weights = [weight for weight in err_min_weights if float(weight) == 0.0 or pd.isna(weight)]

print("There are " + str(len(err_min_weights)) + " dog breeds with problematic weight values." )
```

```
There are 3 dog breeds with problematic weight values.
```

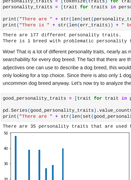
A very insignificant number of dog breeds have problematic entries for their weights, so in our final product, this will largely be a non-issue. These dogs have other data for the other characteristics properly filled in that will allow them to be queried. It would also not be much work to manually find and fill in the minimum and maximum weight for these dog breeds. Let's now analyze the distribution our dog weights. In order to do so, we will find the rough average weight of our dog breed and group them into "Small", "Medium", "Large", and "Giant" according to some rough consensus online about dog sizes.

In [11]:
```
average_weight = [w1+w2/2 for w1,w2 in zip(min_weights,max_weights)]

size_counts = []
for idx in range(len(average_weight)):
    if average_weight[idx] <= 10:
        size_counts.append("Small")
    elif average_weight[idx] > 10 and average_weight[idx] <= 26:
        size_counts.append("Medium")
    elif average_weight[idx] > 26 and average_weight[idx] <= 45:
        size_counts.append("Large")
    else:
        size_counts.append("Giant")

pd.Series(size_counts).value_counts(sort=False).plot(kind='bar')
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c7195b790>
```



Once again, this graph shows some really promising results for our distribution! Our extreme values, "Small" and "Giant" are slightly less frequent than our more middle values "Medium" and "Large", but overall, every size category is well represented, with the smallest being "Giant" with still at least 40 dog breeds represented. As the largest category, "Medium" represents slightly under 100 dog breeds, and slightly out dominates the "Large" category, which makes sense because despite the majority of dog breeds will be medium sized, i.e. right in the middle. Overall, there is a large and distributed variety of dog breed sizes that our information retrieval system has to choose from, providing a wide variety of dogs to further choose from according to the users query!

Finally, let's one more interesting one for fun.

Temperament / Personality Analysis

Dogs are like humans in that they each have their own unique personality, which is what makes them so lovable! They are like tiny human comparisons that can't speak, but are extremely loyal. While all dogs are different, the AKC has provided a temperament category, which is a general descriptor for this dog breed's personality. This is an important metric to include because people will generally use similar adjectives that the AKC uses to describe dogs in their queries describing the dog breed they want. We imagine this column may do a lot of heavy lifting when it comes to information retrieval as a result, so it would be good to analyze it now.

In [12]:
```
temperaments = list(dog_data["temperament"])
err_traits = [temperament for temperament in temperaments if pd.isna(temperament)]
temperaments = [temperament for temperament in temperaments if pd.notna(temperament)]


def tokenize(str):
    return str.split(",")

personality_traits = [tokenize(traits) for traits in temperaments]
personality_traits = [trait for traits in personality_traits for trait in traits]

print("There are " + str(len(set(personality_traits))) + " different personality traits.")
print("There is " + str(len(err_traits)) + " breed with problematic personality trait entries." )
```

```
There are 177 different personality traits.
There is 1 breed with problematic personality trait entries.
```

Wow! That is a lot of different personality traits, nearly as many as there are dogs. Thankfully, this gives us **many** different personality traits to query against, and from the data set we can see that most dogs have at least 3 personality traits, allowing for powerful searchability for every dog breed. The fact that there are that many unique personality traits means that most dog breeds have their own unique set personality traits, and there is not much overlap in dog breed personalities. Since there are only so many different adjectives one can use to describe a dog breed, this would suggest that many queries will have a uniquely top result that is very relevant, which for our purposes is a good thing, since most people are not looking to adopt multiple dog breeds at one time. They are really only looking for a top choice. Since there is also only 1 dog breed with a problematic personality trait entry, this will also be a non-problem. Unfortunately, this dog breed may not show up in many queries, however, after further inspection it seems that it is a rather niche / uncommon dog breed anyway. Let's now try to analyze the distribution of these personality traits.

In [13]:
```
good_personality_traits = [trait for trait in personality_traits if personality_traits.count(trait) > 6]

pd.Series(good_personality_traits).value_counts(sort=False).plot(kind='bar')
print("There are " + str(len(set(good_personality_traits))) + " personality traits that are used for at least 7 different dog breeds.")
```

```
There are 35 personality traits that are used for at least 7 different dog breeds.
```


From our graph we can see that there are still a significant, although much lesser, amount of personality traits that are used to describe at least 7 different dog breeds, 35 to be exact. This number was chosen arbitrarily so that all words could fit on the graph, since the vast majority of words were only used a few times, if not once. This further supports the ideas previously mentioned, which is that the vast majority of dog breeds are being unique described by specific personality traits, giving us powerful and expressive searching ability. This graph shows that many breeds do have some personality traits in common with each however, which is also a good thing since we want our user to have a variety among choices from a certain personality type to further narrow their query. Some notably common personality traits include "Friendly", occurring in nearly 50 different breeds, "Confident", "Affectionate", "Loyal", "Alert", "Intelligent", which are quite generic. Seeing as how most breeds have more than 3 personality traits, these are most likely additional traits that may not be as unique, but will help us when trying to identify dogs in vague or broad queries. Among the other "common" traits, there is a pretty even distribution. Overall, the temperament / personality traits data proves to be very insightful and potentially useful as it provides us to allow users to query dog breeds with both extreme narrowness and broad generality which should return accurate and relevant results.

Conclusion

As we've seen in this data set exploration, the AKC dog breed data set is very feasible to use in an information retrieval system. This data set contains information about nearly 80% of all recognized dog breeds, including the most popular and some of the more uncommon ones, thus it casts a wide net in the context of our information need. At 277 entries, it should not pose any technical problems in terms of search speed or memory usage. With a wide variety of both numeric and textual information about every dog breed, this should allow us to build a powerful and expressive system, since we have so many variables to compare queries to. Along with a general description of each dog breed, information retrieval using classic methods such as cosine and jaccard similarity will be very feasible. An inquiry into 3 of the 20 possible variables has shown us that our data is generally very well distributed across a respectable number of different buckets / bins. This suggests to us that queries on both ends of specificity, narrow and broad, should have definitively relevant set of results that will match our query. While there are definitely some issues with the data, such as missing popularity rankings and NaN values, many of these issues will probably not have a large enough impact on our results, if not easily rectifiable. The extensiveness of our data will simply allow us to consider other variables and ignore NaN ones, plus, finding a ranking of nearly 300 dog breeds would nigh impossible, so 193 is still an impressive and usable amount. Overall, this data set is very well suited for the task.