Programowanie I

Zajęcia nr 11

Rafał Masełek

19. maja 2022r.

Zadanie 0 – pola i metody statyczne

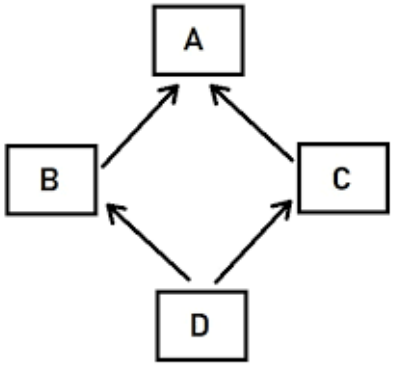
Napisz klasę MathTool, która posiada pola statyczne przechowujące wartości stałych matematycznych: π oraz e. Dodatkowo klasa ma metodę statyczną "factorial", która liczy silnię w sposób rekurencyjny. Przetestuj działanie klasy **bez tworzenia jej obiektów**.

Zadanie 1 – dziedziczenie podwójne a la diament

Napisz klasę Parent, która ma trzy pola "name", "surname" oraz "votes", przechowujące odpowiednio: imię, nazwisko oraz informację, czy dana osoba głosuje w wyborach (True/False). Pola powinny być ustawiane za pomocą konstruktora, gdzie domyślna wartość pola "votes" jest True. Dodatkowo klasa ma metodę "hello", która zwraca komunikat (napis) "Hi! I am a parent!".

Zdefiniuj dwie klasy dziedziczące po Parent: ChildA oraz ChildB. Klasy mają przeciążoną metodę "hello". Klasy różnią się tym, że ChildA głosuje, a ChildB nie. Mają zatem przeciwne wartości pola "votes".

Na koniec zdefiniuj klasę GrandChild dziedziczącą zarówno po ChildA jak i ChildB. Diagram dziedziczenia ma zatem strukturę "diamentu", jak przedstawiono schematycznie na rys. . Klasa GrandChild ma przeciążoną metodę "hello".



W głównej części programu stwórz obiekt klasy GrandChild podając imię i nazwisko, a następnie wypisz wynik działania metody "hello". Wypisz również wartość pola "votes".

Każda klasa posiada automatycznie definiowane pole "__mro__", które jest krotką zawierającą nazwy typów (klas). Kiedy chcemy dostać się do pola lub metody danego obiektu, to są one szukane w klasach zgodnie z kolejnością we wspomnianej krotce. Spróbuj zamienić kolejność dziedziczenia po klasach ChildA i ChildB klasy GrandChild. Co się zmieniło?

Zadanie 2 – Model Standardowy

Obowiązującym paradygmatem Fizyki Cząstek jest Model Standardowy – teoria, która w spójny sposób opisuje oddziaływania silne, słabe i elektromagnetyczne. Model zawiera następujące cząstki:

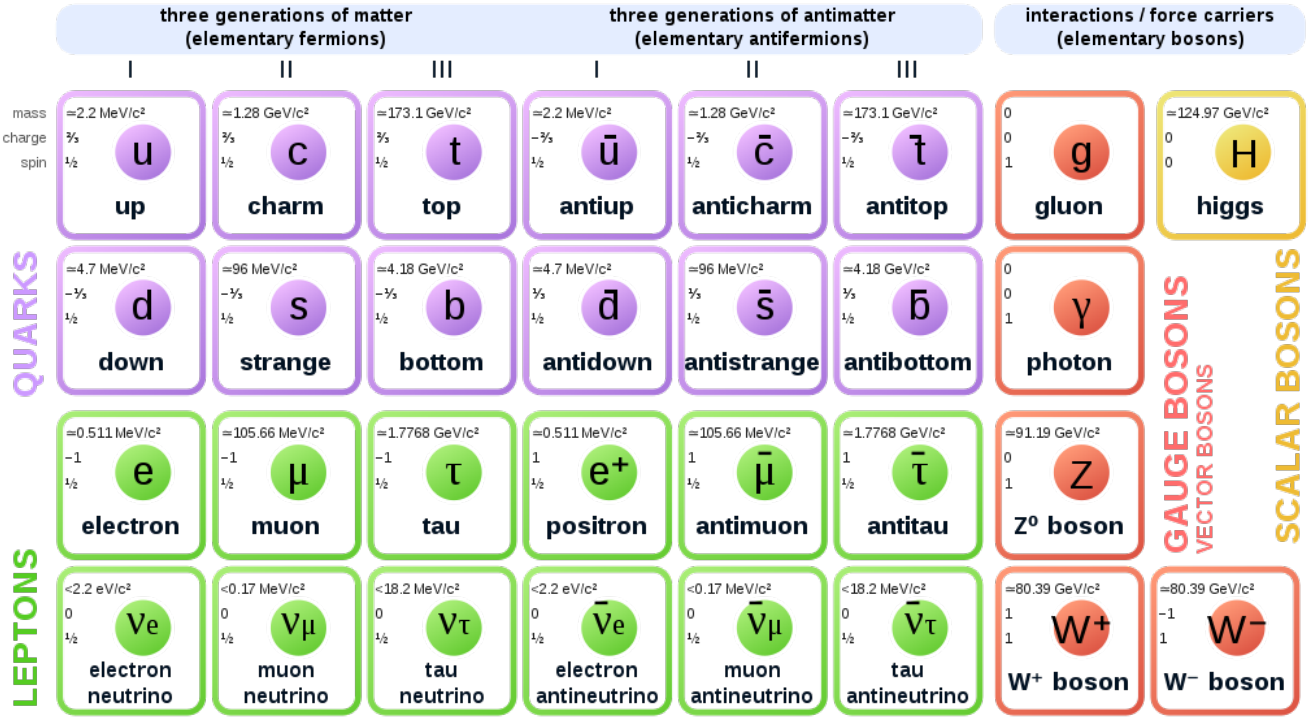
- 6 kwarków
  - górny/up (u)
  - dolny/down (d)
  - powabny/charm (c)
  - dziwny/strange (s)
  - szczytowy/top (t)
  - denny/bottom (b)
- 3 leptony naładowane elektrycznie
  - elektron/electron (e)
  - mion/muon (μ)
  - taon/tau (τ)
- 3 nienaładowane elektrycznie leptony – neutrina, po jednym dla elektronu, mionu i taonu
- 4 bozony wektorowe o spinie 1
  - foton/photon (γ)
  - bozon Z
  - bozon W^+ (i jego antycząstka W^-)
  - gluon
- bozon Higgsa – cząstka skalarna o spinie 0

Wszystkie cząstki wraz z ich **antycząstkami** przedstawiono na rys. .

Zaimplementuj strukturę Modelu Standardowego cząstek elementarnych za pomocą klas i dziedziczenia w Pythonie. Zacznij od stworzenia klasy Particle, która będzie klasą bazową dla innych. Klasa particle powinna mieć następujące pola ustawiane za pomocą argumentów konstruktora:

- full_name – pełna nazwa cząstki
- name – krótka nazwa, np. "e-"
- mass – masa cząstki, domyslnie 0
- charge – ładunek elektryczny, domyślnie 0
- color – True/False w zależności czy cząstka oddziałuje silnie (kwarki i gluony) czy nie (pozostałe cząstki), domyślnie False

Dodatkowo, klasa Particle powinna mieć pola baryon_number oraz lepton_number ustawiane na 0. **Uwaga: Ponieważ właściwości stworzonych cząstek nie powinny być zmieniane w programie, dlatego korzystamy z konwencji nazewniczej, w której każde nazwa każdego pola zaczyna się od znaku podkreślenia "_".** Klasa Particle powinna mieć przeciążone metody __str__ i __repr__, pierwsza z nich zwracać pełną nazwę cząstki. Dodatkowo klasa Particle powinna zawierać metodę "get_antiparticle", która zwraca obiekt Particle (lub pochodny), ale dla antycząstki. Antycząstka ma odpowiednią nazwę (długą i krótką, przypomnij sobie zadanie z kodami PDG), taką samą masę i parametr "color" jak cząstka, ale przeciwny ładunek elektryczny i liczby leptonową i barionową. Uwaga: ta metoda powinna działać również dla klas dziedziczących po klasie Particle, czyli zwracany obiekt musi być odpowiedniego typu. Zamiast przeciążać, użyj tutaj " self.__class__ " aby uzyskać odpowiedni typ. Klasa Particle powinna również posiadać dwie metody typu "get", które będą zwracać liczbę barionową i leptonową.



Zdefiniuj klasy Fermion, Scalar i Vector, które dziedziczą po particle. Klasy te mają dodatkowe pole "spin", ustawione odpowiednio na 0.5, 0.0, 1.0. Zadbaj o to, żeby wywołanie __repr__ dla każdej z klas pochodnych zwracało odpowiedni napis.

Zdefiniuj klasy Lepton i Quark. Pierwsza z nich ma liczbę leptonową ustawioną na 1. Klasa Quark ma z kolei ustawioną liczbę barionową na 1/3. Zadbaj o to, żeby wywołanie __repr__ dla każdej z klas pochodnych zwracało odpowiedni napis.
Korzystając z rys. stwórz obiekty reprezentujące wszystkie cząstki Modelu Standardowego: 6 kwarków, 6 leptonów, 4 bozony cechowania i bozon Higgsa. Następnie, wykorzystując metodę "get_antiparticle", stwórz antycząstki i umieść je razem z cząstkami w jednej liście. Dla każdej cząstki/antyczastki wypisz w czytelny sposób: jej pełną nazwę, liczbę leptonową i barionową, napis zwracany przez __repr__.
Dodatkowo, zaimplementuj klasę "Leptoquark", dziedziczącą po klasach Lepton i Quark. Leptokwarki to hipotetyczne cząstki, które posiadają zarówno niezerową liczbę leptonową jak i barionową. Stwórz instancję tej klasy i wypisz jej szczegóły tak samo jak dla cząstek z Modelu Standardowego. Dodatkowo wypisz wartość parametru color.

Dla chętnych:
Supersymetria to jedna z najpopularniejszych teorii fizyki spoza Modelu Standardowego. Supersymetria zakłada istnienie symetrii pomiędzy bozonami i fermionami, która w niskich energiach zostaje spontanicznie złamana. Najprostszą teorią supersymetryczną jest Minimal Supersymmetric Standard Model (MSSM), w którym każda cząstka z SM ma swojego "superpartnera", czyli nową cząstkę, której spin różni się o 1/2. Np. gluon, który jest bozonem o spinie 1, ma superpartnera "gluino", który jest fermionem o spinie 1/2, itd. Korzystając ze stworzonych klas, a w razie konieczności pisząc nowe, zaimplementuj cząstki supersymetryczne z MSSM:

- 6 skwarków, po jednym dla każdego kwarku (skalarne cząstki naładowane, oddziałujące silnie)
- selektron, smion, staon (skalarne cząstki naładowane)
- 3 sneutrina (neutralne elektrycznie skalary)

- gluino, wino, bino[1] (fermiony, gluino ma ładunek kolorowy, są trzy wina: $W^0, W^\pm$ i jedno bino: B^0).
- 2 higgsy h_u i h_d zamiast jednego (neutralne skalary)
- 2 higgsina H_u i H_d (fermiony, ładunki $+1$ i -1 odpowiednio)

Po zaimplementowaniu, wypisz własności cząstek podobnie jak to zrobiłeś dla Modelu Standardowego.

Zainteresowanych Supersymetrią odsyłam do `https://arxiv.org/pdf/hep-ph/9709356.pdf`

[1] W Modelu Standardowym bozony $W^\pm$, Z i foton są rezultatem mieszania się dwóch pól: W i B. W MSSM superpartnerzy tych pól również się mieszają, tworząc różne obserwowalne cząstki (and. mass eigenstates). Dla uproszczenia skupmy się w zadaniu na polach, a nie stanach masowych.