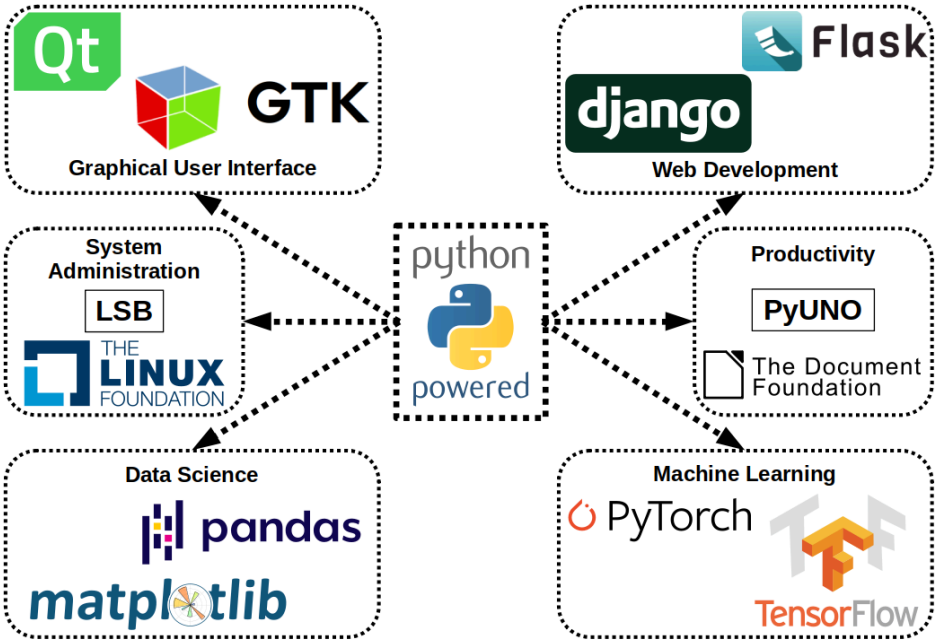
Język programowania wysokiego poziomu 1 - Wprowadzenie do języka Python

Przed przystąpieniem do wykonania ćwiczenia należy zapoznać się ze środowiskiem google colab, na którym mogą Państwo w dowolnej chwili korzystać z platformy z interpreterem Pythona.

Python - język programowania wysokiego poziomu, ogólnego przeznaczenia, cechujący się rozbudowanym pakietem bibliotek.



Rys. 1. Wybrane pakiety Pythona i ich zastosowania
Źródło: wikipedia

Wybrane, ważne cechy:

- dynamiczne typowanie
- automatyczne zarządzanie pamięcią
- wspieranie wielu paradygmatów (w tym obiektowego)
- czytelność
- **indentacja - wymuszenie formatowania kodu**

Formatowanie wyjścia:

```
print("Wartosc zmiennej: %s" % zmienna)
print("Wartosc zmiennej:", zmienna)
print(f"Wartosc zmiennej:{zmienna}")
print("Wartosc zmiennej:{}".format(zmienna))
```

Instrukcje warunkowe:

```
if warunek1:
  pass
elif warunek2:
  pass
else:
  pass
```

Funkcje:
W Pythonie nie ma konieczności wskazywania typów zwracanych ani typów argumentów. Ta sama funkcja może zwracać różne typy. Możliwe jest na przykład:

```
def function_name(arg1, arg2):
  if arg1 == 0:
    return
  if arg2 == 1:
    return 1
  return False
```

Występują również funkcje anonimowe (znane jako lambda functions). Są to (a przynajmniej powinny takie być) krótkie funkcje bez nazwy stosowane pomocniczo przy innych operacjach. Funkcja anonimowa może zostać przypisana do zmiennej.
Przykładowa funkcja lambda:

```
lambda x: x % 2 == 0
```

zwraca True jeżeli x jest liczbą parzystą

Pętle:
Pętla for ma postać:

```
for element in nazwa_kolekcji:
  …some code…
```

w wyniku takiej postaci uzyskujemy dostęp do elementu kolekcji
lub

```
for index, element in enumerate(nazwa_kolekcji):
  …some code…
```

dostęp do elementu i jego indeksu. Jeśli chcemy w pętli wygenerować kolejne liczby, to służy do tego funkcja range, która przyjmuje początek, koniec i krok. Jednak podanie

samego końca spowoduje wygenerowanie sekwencji o początku 0 i kroku 1. Przykład użycia:

```
for i in range(10):
  …some code…
```

Pętla while wykonuje się dopóki warunek jest prawdziwy

```
while warunek:
  …some code…
```

podobnie jak w innych językach występują instrukcje break oraz continue.

Funkcje map, filter i reduce
To gotowe funkcje ułatwiające wiele zadań.

Ciekawe zastosowania dla lambdy obejmują jej wykorzystanie w innych funkcjach, np:
map(operacja, kolekcja) - zastosuje operację do każdego elementu kolekcji
filter(kryterium, kolekcja) - podobnie jak map zastosuje kryterium do każdego elementu kolekcji, ale wynik będzie zawierał tylko te elementy, dla których kryterium zwróci True
Te funkcje zwracają jednak obiekt własnej klasy.

Map:
Umożliwia zastosowanie funkcji do elementów kolekcji. Przykładowo

```
items = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x**2, items))
```

jest równoważne do:

```
items = [1, 2, 3, 4, 5]
squared = []
for i in items:
    squared.append(i**2)
```

Filter:
Funkcja ta służy do zachowania tylko tych elementów, dla których funkcja podana jako parametr zwróci True. Przykład:

```
number_list = range(-5, 5)
positive = list(filter(lambda x: x > 0, number_list))
print(positive)
# Output: [1,2,3,4]
```

Istnieje także funkcja reduce (jej zastosowanie wymaga jej zaimportowania). Służy ona do wykonywania obliczeń na kolekcji i zwracania wyniku.

Struktury danych:

```
lista = [1,2,[10,20],"napis"]
```

możemy elementy dodawać, usuwać, zmieniać, np:

- del lista - usunie całą listę,
- lista.append(...) spowoduje dodanie elementu na koniec listy
- lista.remove(...) - usunie element o podanej nazwie,
- lista.pop(...) usunie element o podanym indeksie
- lista.insert(index, element) doda element na wskazaną pozycję

Operatory + i * mają zdefiniowane działanie w kontekście list. +, tak jak w przypadku string, to konkatenacja, czyli połączenie dwóch list w jedną.

```
tupla = (1,2,3)
```

Znana również jako krotka - po utworzeniu tupli nie możemy modyfikować jej zawartości (dodawać, usuwać, zmieniać elementów), jednak możemy ją wyczyścić usuwając wszystkie elementy
set i frozen set - unikalne wartości, pierwszy modyfikowalny, drugi nie :)

słownik (mapa, hashmapa, dictionary) forma **klucz : wartość**. Tworzenie:

```
slownik = {klucz : wartosc}
```

Uwaga! kluczami mogą być jedynie hashowalne elementy **(nie listy, słowniki, sety** - chyba, że frozen sety :) **…).** A żeby hashowanie miało w tym przypadku sens musi się odbywać na obiektach niemodyfikowalnych (*immutable)*.

Slicing
Używany do ekstrakcji wybranych obiektów z list, stringów lib tupli.

```
a[start:stop]  # obiekty od start do stop-1
a[start:] # obiekty od start do końca
a[:stop] # obiekty od początku do stop-1
a[:] # kopia wszystkich elementów
a[:-3] # 3 elementy od końca
```

można też skorzystać ze schematu `[start:stop:step]`, np:

```
a[::-1] # obiekty w odwróconej kolejności
a[::2] # co drugi obiekt
```

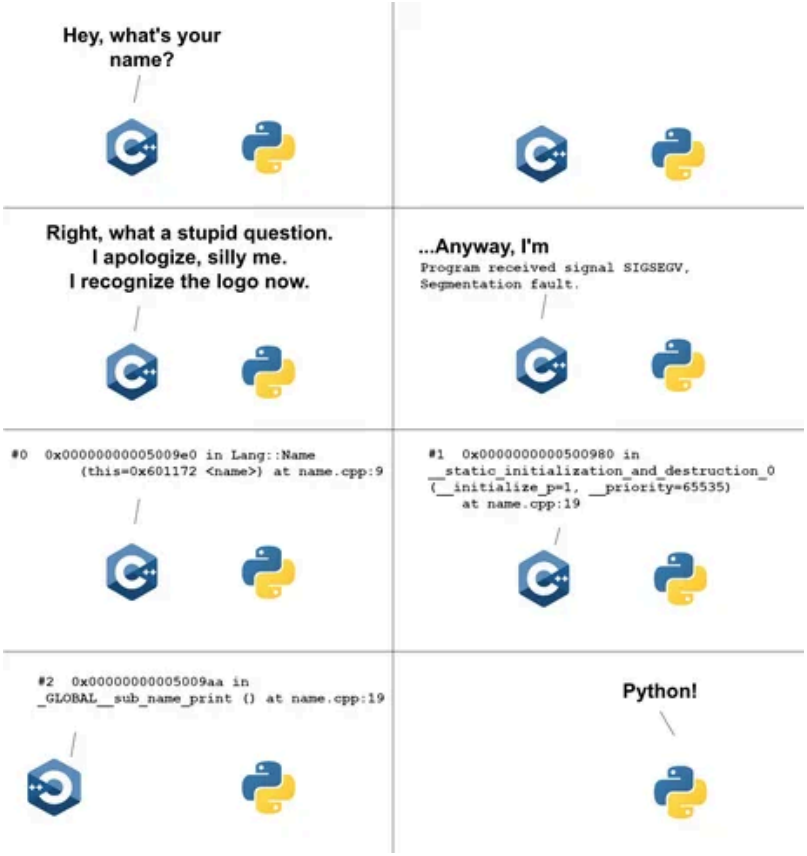
Należy zachować ostrożność korzystając z tego rozwiązania - np. w przypadku wyboru a[:-2], gdy a zawiera tylko jeden element otrzymamy pustą listę, a nie błąd. Chcąc przypisać wynik slicingu do zmiennej otrzymamy kopię, ale możemy też zamienić wybrane elementy (np. lista[0:2] = [0,0,0] spowoduje podmianę pierwszych trzech elementów w liście na zera).

W Pythonie zwykle łatwo można printować struktury danych w konsoli (nie wymaga to przeciążania operatorów jak przykładowo w języku C++). Na przykład wywołanie funkcji print z argumentem typu lista intów spowoduje wydrukowanie wszystkich jej elementów. podobnie dla słowników, struktur zagnieżdżonych, innych typów.

Wspaniałą cechą Pythona jest dostępność wielu przydatnych funkcji w obrębie typów danych. Przykładowe funkcje: keys(), sum(), sort().

Zadanie:
Mając pięć dodatnich liczb całkowitych, znajdź wartości minimalne i maksymalne, które można obliczyć, sumując dokładnie cztery z pięciu liczb całkowitych. Następnie wydrukuj odpowiednie wartości minimalne i maksymalne w formacie “Sum1: <WARTOSC>, sum2:<WARTOSC>”. Jeżeli suma sum1 i sum2 jest liczbą parzystą, to usuń wszystkie parzyste liczby z początkowej listy (wykorzystaj funkcję filter, utwórz własną funkcję lambda), w przeciwnym razie usuń wszystkie liczby nieparzyste. Na koniec wydrukuj listę.

Przykład:
Dany jest ciąg liczb 1,3,2,7,5
Suma czterech najmniejszych liczb: 1+2+3+5 = 11
Suma czterech największych liczb: 7+5+3+2=17
Funkcja ma za zadanie pokazać 11 17
11+17 = 28
28 jest liczbą parzystą, więc w konsoli powinno ukazać się 1,3,7,5



źródło reddit