Wizualizacja danych
semestr letni 2024

Dr Anna Muranova
UWM w Olsztynie

Wykład 1

- Wstęp do języka Python
- Wizualizacja danych w Pythonie
- Wizualizacja danych w Rze

Język Python

- Poprawna wymowa: pajton.
- Język Python stworzył we wczesnych latach 90. Guido van Rossum – jako następcę języka ABC.
- Nazwa języka pochodzi od serialu komediowego emitowanego w latach siedemdziesiątych przez BBC – „Monty Python's Flying Circus" (Latający cyrk Monty Pythona). Projektant, będąc fanem serialu i poszukując nazwy krótkiej, unikalnej i nieco tajemniczej, uznał tę za świetną.

Przełomowy rok – 2008

- Utworzenie drugiej gałęzi rozwoju 3.x. Początkowe obie gałęzie były rozwijane niezależnie, lecz wsparcie Pythona 2.x zostało zakończone w roku 2020.
- Python 2.x cały czas jest wykorzystywany w niektórych narzędziach, np. w ArcGis Desktop
  `https://support.esri.com/en/technicalarticle/000013224`

Oprogramowanie

Python w wersji 3.12.
`https://www.python.org/`

Środowisko **PyCharm**
`https://www.jetbrains.com/pycharm/`

Uruchomianie konsoli

Main menu –> Tools –> Python or Debug Console
Podstawowe operacje arytmetyczne:

```
+, -, *, /, //, %, **
>, <, <=, >=, !=, ==
```

Importowanie bibliotek

```
import math
math.e
```

Pierwszy program

```
print("Hello world!")
```

Jeszcze jeden program:

```
print('3 / 2 =', 3 / 2)
print('3 // 2 =', 3 // 2)
print('3 / 2.0 =', 3 / 2.0)
print('3 // 2.0 =', 3 // 2.0)
```

Styl PEP8

- wymowa: pi-i-pi-ejt
- standaryzacja kodu używana m.in. przy rozwijaniu nowych funkcjonalności
- używanie daje lepszą organizację i czytelność kod
- pełna wersja `https://www.python.org/dev/peps/pep-0008/`

Znaki odstępu

- we wcięciach stosujemy spacje (a nie tabulatory)
- każdy poziom wcięcia powinien składać się z 4 spacji
- wiersz powinien składać się z maksymalnie 79 znaków

Puste linie:

- dwie linie między funkcjami najwyższego poziomu i między klasami.
- pojedyncza linia między funkcjami w klasie

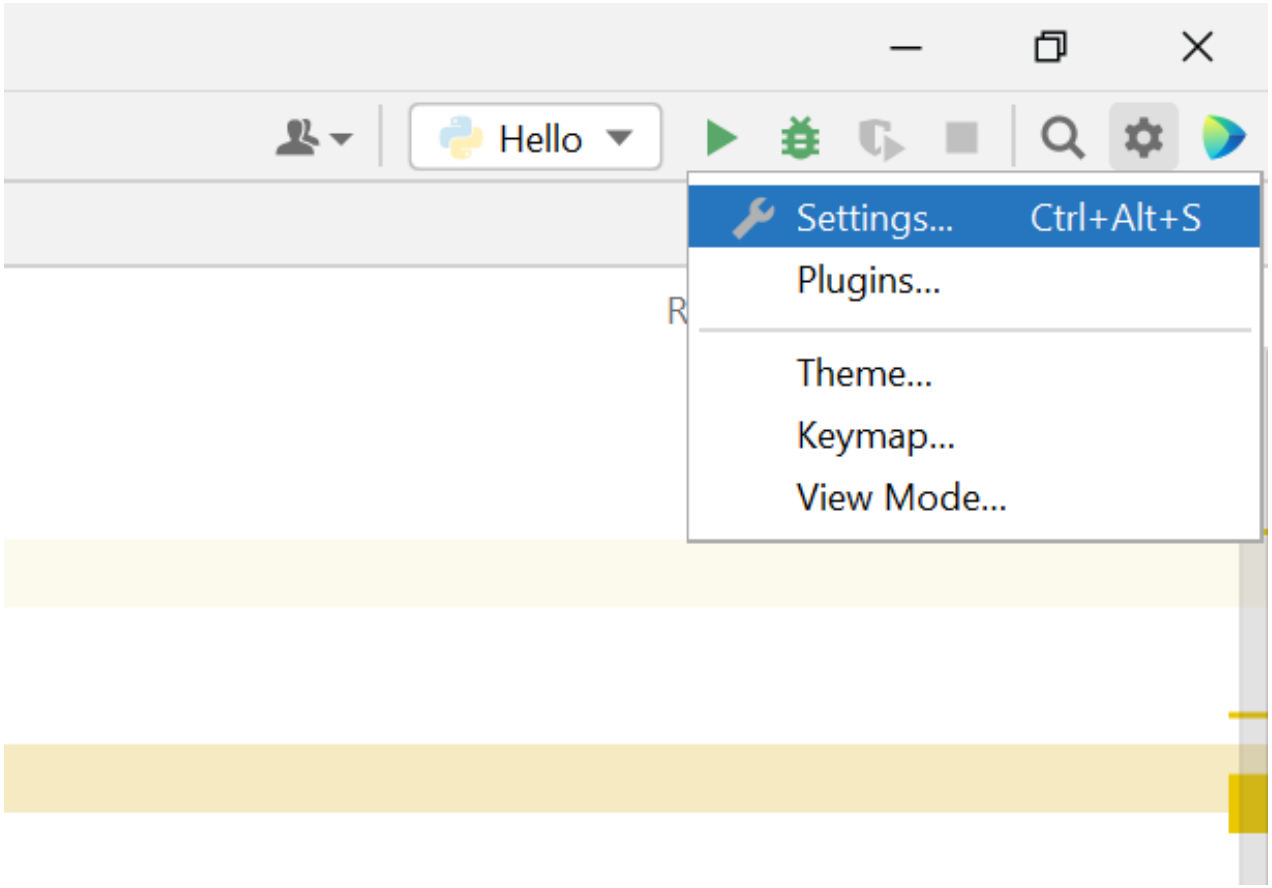
Kodowanie:

- dla Pythona 3 sugerowane i domyślne to UTF-8.

Stringi:

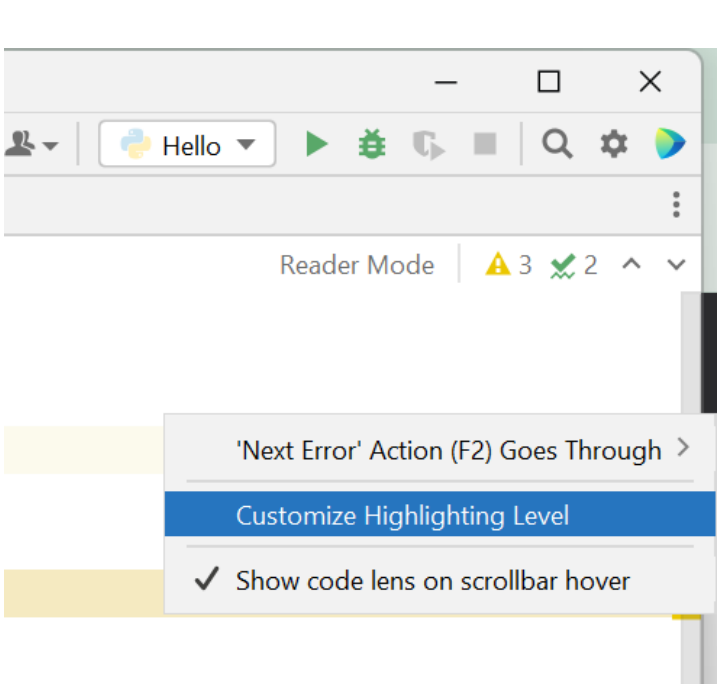
- można używać pojedynczych apostrofów jak i podwójnych cudzysłowów
- ważne, aby stosować wybraną notację konsekwentnie
- jedyny wyjątek to gdy wewnątrz `stringu` chcemy użyć cudzysłów np. `print('Oglądam film "Player One"')`

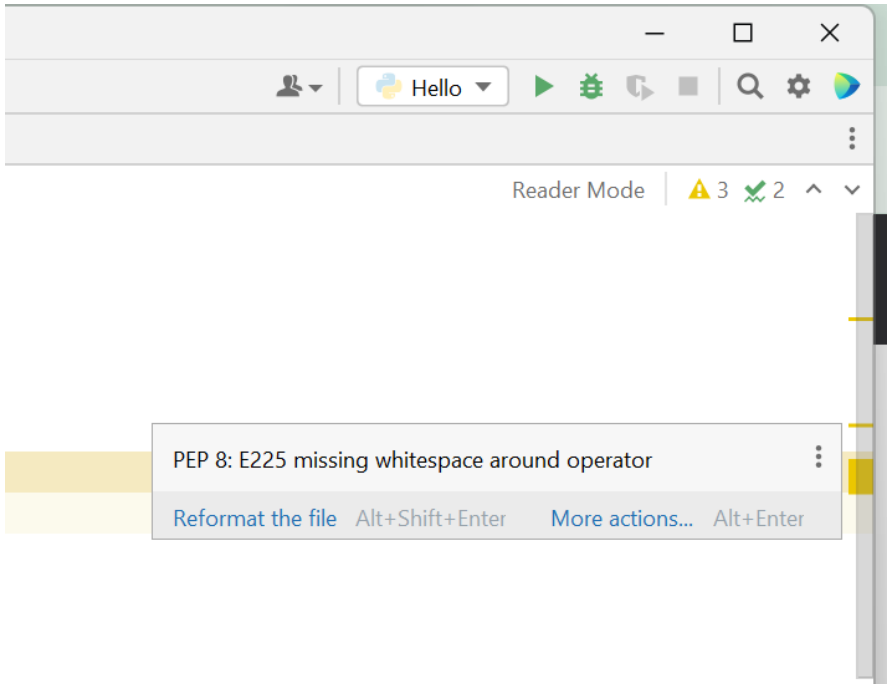
Jak sprawdzić, by kod spełniał standard PEP8?



Settings –> Editor –> Inspections –> Python –> PEP8 . . .

Dodać podpowiedzi:





Przykład

```
var1 = 2
var2 = 3
var1 += 2
var3 = var1
var1 *= 3
var4 = var1/var2

var2 +=2


print('var2 = ', var2)
#print('var2 = ', var2+=2)
```

oznacza komentarz

Funkcje w Pythonie

Funkcje są definiowane z użyciem słowa kluczowego def, po którym umieszcza się nazwę funkcji, a potem nawiasy. Jeżeli funkcja nie wymaga informacji z zewnątrz nawiasy pozostawiamy puste.

```
def przywitanie():
    print ("Pozdrowienia z mojej funckji!")

def przywitanie_imienne(imie, zyczenia):
    print ("Witaj" + imie + ". Zycze Tobie " + zyczenia)

przywitanie() # Wypisze "Pozdrowienia z mojej funckji!"
przywitanie_imienne("Jacek", "zdrowia") # Wypisze immienne zyczenia
```

Funkcje w Pythonie

Aby przerwać działanie funkcji i zwrócić wartość, musisz użyć słowa `return`, a za nim umieścić zwracaną wartość. Jeżeli pominiesz wartość, to funkcja tylko zakończy swoje działanie i nic nie zwróci. Przykład:

```
def dzielenie(dzielna, dzielnik):
    if(dzielnik == 0):
        return # zakoncz funkcje nic nie zwracajac
    else:
        return dzielna / dzielnik

print (dzielenie(5, 0))
print ("############################")
print (dzielenie(10, 2))
```

if...

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

Bez wcięć będzie błąd kompilacji

```
a = 33
b = 200
if b > a:
print("b is greater than a")
```

if... else...

```
a = 200
b = 33
if b > a:
  mmin  = a
else:
  mmin = b
print(mmin)
```

if... else...

```
def mmin (a, b):
    if b > a:
        return a
    return b

print(mmin(330,2))
print(mmin(2,330))
```

elif

```
a = 200
if a > 0:
  print("a is positive")
elif a < 0:
  print("a is negative")
else:
  print("a is 0")
```

Można używać bez `else`:

```
a = 200
if a > 0:
  print("a is positive")
elif a < 0:
  print("a is negative")
```

Przykład

```
day = int(input("Enter a number from 1 to 7: "))

if day == 1:
    print(day, "is Sunday")
elif day == 2:
    print(day, "is Monday")
elif day == 3:
    print(day, "is Tuesday")
elif day == 4:
    print(day, "is Wednesday")
elif day == 5:
    print(day, "is Thursday")
elif day == 6:
    print(day, "is Friday")
elif day == 7:
    print(day, "is Saturday")
else:
    print("Wrong input! Please enter a number from 1 to 7.")
```

Przykład

```
def week(day):
    if day == 1:
        print(day, "is Sunday")
    elif day == 2:
        print(day, "is Monday")
    elif day == 3:
        print(day, "is Tuesday")
    elif day == 4:
        print(day, "is Wednesday")
    elif day == 5:
        print(day, "is Thursday")
    elif day == 6:
        print(day, "is Friday")
    elif day == 7:
        print(day, "is Saturday")
    else:
        print("Wrong input! Please enter a number from 1 to 7.")

week(3)
```

```
def week(day):
    if day == 1:
        return "Sunday"
    elif day == 2:
        return "Monday"
    elif day == 3:
        return "Tuesday"
    elif day == 4:
        return "Wednesday"
    elif day == 5:
        return "Thursday"
    elif day == 6:
        return "Friday"
    elif day == 7:
        return "Saturday"
    else:
        return 0

print(week(3))
print(week(8))
```

Krotka pisownia warunku

```
if a > b: print("a is greater than b")
```

```
a = 2
b = 330
print("A") if a > b else print("B")
```

```
a = 330
b = 330
print("A") if a > b else print("=") if a == b else print("B")
```

... and ...

```
a = 5
if a > 0 and a < 100:
    print("True")
else:
    print("False")
```

```
a = 5
if 0 < a < 100:
    print("True")
else:
    print("False")
```

... or ...

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

... not ...

```
a = 33
b = 200
if not a > b:
  print("a is NOT greater than b")
```

Włożone if ...

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

pass

Jeżeli w `if` nic nie wykonujemy to musimy tam wpisać pass

```
a = 200
b = 20

if b > a:
  pass
else:
    print("Hello")
```