Laboratorium– Python – PPY – PJATK 2023

Zajęcia 4

29.03.2023

1 Wprowadzenie do laboratorium

1.1 Linie

Jaka jest różnica między linią **fizyczną** a **logiczną**?

```
x = 1  # pierwsza linia logiczna i fizyczna
if x > 0:  # druga linia logiczna i fizyczna
    print("x jest większe od zera")  # trzecia linia logiczna i fizyczna
# czwarta linia fizyczna
```

- Niezwykle istotny jest PEP 8 – zbiór zasad (`https://peps.python.org/pep-0008/`).

1.2 Dobre praktyki (wcięcia) – przykład:

```
# Correct:

# Aligned with opening delimiter.
foo = long_function_name(var_one, var_two,
                         var_three, var_four)

# Add 4 spaces (an extra level of indentation) to distinguish arguments from the rest.
def long_function_name(
        var_one, var_two, var_three,
        var_four):
    print(var_one)

# Hanging indents should add a level.
foo = long_function_name(
    var_one, var_two,
    var_three, var_four)
```

Wcięcia prawidłowe – **4** spacje! – brak spacji – błąd!

1.3 Złe praktyki (wcięcia) – przykład:

```
# Wrong:

# Arguments on first line forbidden when not using vertical alignment.
foo = long_function_name(var_one, var_two,
    var_three, var_four)

# Further indentation required as indentation is not distinguishable.
def long_function_name(
    var_one, var_two, var_three,
    var_four):
    print(var_one)
```

2 Wprowadzenie do instrukcji

Poznane do tej pory instrukcje sterujące

- instrukcja przypisania, np. *x = 1*;
- instrukcja wywołania funckji, np. *print(x)*;
- wyświetlanie wartości, np. *»>1*;
- usuwanie referencji, np. *del x["Stanowisko"]*;

3 Intstrukcje sterujące

Podczas zajęć omówimy następujące instrukcje sterujące

- instrukcje *if/elif/else*;
- pętle *for/else*;
- pętle *while/else*;
- instrukcje *pass, continue, break.*

3.1 Instrukcje *if/elif/else*

Ogólny format instrukcji warunkowych/sterujących:

```
if wyrażenie -> bool:
    zagnieżdżony blok instrukcji sterującej
elif wyrażenie2 -> bool:
    zagnieżdżony blok kodu
elif wyrażenieN -> bool:
    zagnieżdżony blok kodu
```

```
...
else:
    blok kodu wykonywany wtedy, gdy wczesniejsze wyrażenia zwrócą False
```

Przykład działania instrukcji sterującej:

```
print("Pierwsza część kodu")
x, y, z = 9, 2, 3
if x < y and z == 3:
    print("Druga część kodu")
elif x > z:
    print("Blok kodu zagnieżdżony w ELIF")
else:
    print("Dodatkowa część kodu")
print("Trzecia część kodu")
```

Instrukcja if w Java:

```
if ( x > y) {
    System.out.println(x);
} else {
    System.out.println(y);
}
```

Instrukcja if w Python:

```
if x > y:
    print(x)
else:
    print(y)
```

W pythonie występuje dwukropek po zakończeniu wiersza nagłówkowego instrukcji.

3.2 Pętla *for*

Ogólny format pętli 'for':

```
for(dla) elementu in(w) obiekcie_iterowalnym:
    wykonaj zawarte tutaj(w bloku) instrukcje
else(w przeciwnym razie):
    wykonaj zawarte tutaj instrukcje,
    gdy pętla zakończy swoje działanie
```

Przykłady pętli 'for':

```
for litera in "Max Wolf":  # iterowanie po łańcuchu znaków
    print(litera)

for liczba in [1, 2, 3]:  # iterowanie po liście
    print(liczba)
```

```
for litera in ('a', 'b', 'c'):  # iterowanie po krotce
    print(litera)

for (a, b) in [('a', 1), ('b', 2)]:  # iterowanie po liście krotek
    print(a, b)

for key in {"Key": 1}:  # iterowanie po kluczach słownika
    print(key)

for (key, value) in {"Key": 1}.items():  # iterowanie po parach klucz-wartość słownika
    print(key, value)

for i in range(8):  # przykład pętli zagnieżdżonej
    for j in range(7):
        print(i, j)

for i in "Powiedziałem":  # przykład wykorzystania pętli for z klauzulą else
    print(i, end="")
else:
    print(".")
```

3.2.1 Przykłady – "for":

1. Przykład oddzielania wyrazów w słowie za pomocą przecinka:

```
wyraz = "Warszawa"
for litera in wyraz:
    print(litera, end=", ")
```

2. Przykład ciągu liczbowego:

```
suma = 0
for liczba in range(1, 25):
    print(suma, end=" | ")
    suma += liczba # operator inkrementacji: suma = suma + liczba
print("Suma:", suma)
```

- Alternatywa do powyższego kodu z wykorzystaniem "else":

```
 suma = 0
for liczba in range(1, 25):
    print(suma, end=" | ")
    suma += liczba
else:
print("Suma:", suma)
```

3. Przykład z użyciem listy krotek:

```
lista_krotek = [('a', '10'), ('b', '20'), ('c', '30')]
for (litera, liczba) in lista_krotek: # () zwiększają przejrzystość kodu
    print(litera, liczba)
```

3. Przykład z użyciem słownika:

```
cialo_niebieskie = dict([
    ("Typ ciała niebieskiego", "satelita"),
    ("masa [kg]", 7.347_673e22),
    ("Średnica [km]", 3_474),
    ("Odległość od środka do środka ziemi [km]", 384_399)
])
for (klucz, wartosc) in cialo_niebieskie.items():
    print(klucz + ":", wartosc)
```

4. Tabliczka mnożenia, z zagnieżdzeniem pętli:

```
for i in range(1, 11):
    for j in range(1, 11):

        liczba = str(i*j)
        dlugosc_liczby = len(liczba)

        if dlugosc_liczby == 1:
            liczba = " " + liczba
        elif dlugosc_liczby == 2:
            liczba = " " + liczba

        print(liczba, end=" | ")
    print()
```

3.3 Pętla *while*

Ogólny format pętli 'while':

```
OGÓLNY FORMAT PĘTLI WHILE
while(dopóki) warunek(jest prawdziwy):
    wykonuj zawarte tutaj instrukcje
else(w przeciwnym razie):
    wykonaj zawarte tutaj instrukcje,
    gdy pętla zakończy swoje działanie
```

Przykłady pętli 'while':

- Przykład pętli teoretycznie nieskończonej:

```
while True:
    print("To jest pętla nieskończona (teoretycznie).")
```

Pętlę 'while' teoretycznie nieskończoną można zakończyć przy użyciu CTRL + C (w wierszu poleceń) lub przerywając sesję (PyCharm) przyciskiem STOP.

- Przykład typowej pętli 'while':

```
x = 0
y = 10
while x < y:
    x += 1
    print(x)
```

- Przykład typowej pętli 'while' z (opcjonalną) klauzolą 'else':

```
x = set([1, 2, 3, 4, 5])
while x:
    print(x.pop())
else:
    print("Zbiór x jest już pusty.")
```

- Przykład zagnieżdzonej pętli 'while':

```
x = 5
y = 5
while x:
    while y:
        print("W pętli drugiego poziomu.")
        y -= 1
    print("W pętli pierwszego poziomu.")
    x -= 1
```

3.3.1 Przykłady – "while":

Staramy się nie zagnieżdzać więcej niż **trzech** poziomów pętli.

1. Wyświetlanie wszystkich liczb parzystych, malejąco:

```
x = 30
while x:
    if x % 2 == 0:
        print(x, end=" | ")
    x -= 1
    if x == 0:
        print(x, end=" | ")
```

3.4 Podsumowanie pętli

Pętle **"for"** najlepiej sprawdzają się do iterowania po elementach kolekcji, a pętle **"while"** w pozostałych sytuacjach.

4 Pozostałe instrukcje

- Instrukcja "pass":

```
if True:
    pass

for i in range(10):
    pass
```

- Przykład:

```
print("Przed pętlą.")

for i in range(10):
    # komentarz testowy
    pass

print("Za pętlą.")
```

- Instrukcja "continue":

```
for i in range(10):
    if i % 2 == 0:
        print(i)
        continue
    print(":")
```

- Przykład (sprawdzanie czy dana liczba jest parzysta):

```
x = 10
while x:
    x -= 1
    if x % 2 == 0:
        continue
    print(x)
else:
    print("Koniec programu!")
```

- Instrukcja "break":

```
for i in range(10):
    if i < 5:
        print("$$$")
    else:
        break
```

- Przykład (zliczanie znaków z przerwaniem):

```
znaki = []
print("Liczenie podanych znaków:")
while True:
    dane_wejsciowe = input("Dane wejściowe:")
    if dane_wejsciowe == "red pill":
        break
    else:
        znaki += list(dane_wejsciowe)

print("Liczba wystąpień:", end=" ")
for element in sorted(set(znaki)):
    print(element, "=", znaki.count(element), end=" | ")
```

- Przerywanie:

```
for i in range(10):
    if i != 5:
        print(i)
    else:
        break
else:
    print("W klauzuli 'else'.")
```