Akademia Pythona

III Instrukcje i składnia

KN Pythona - Kurs Pythona

KN Pythona wita na kursie Pythona.

Plan:

- Pętle while i for
- Iteracje i składanie list - część 1.
- Wprowadzenie do dokumentacji

Pętle while

Pętla while:

- Najbardziej uniwersalna instrukcja pętli w Pythonie
- Wykonuje zagnieżdżony blok kodu dopóki warunek w nagłówku jest prawdą

Pętla while - składnia

```
while <test>:
    <instrukcje1>
else:
    <instrukcje2> # wykonane jeśli pętli nie zakończyło br
```

Pętla while

```
a = 0
while a < 10:
    print(a)
    a += 1
# 0 1 2 3 4 5 6 7 8 9
```

Instrukcje break i continue

```
while <test1>:
    <instrukcje1>
    if <test2>:
        break # wyjdź z pętli
    if <test3>:
        continue # przejdź bezpośrednio do nagłówka pętli
else:
    <instrukcje2>
```

Część else pętli while

while/else:

- Unikalna dla Pythona instrukcja
- Udostępnia jawną składnię służącą do zapisu często występującego scenariusza

Część else pętli while

```
found = False
while x and not found:
    if match(x[0]):
        print("ni")
        found = True
    else:
        x = x[1:]
if not found:
    print('not found')
```

Część else pętli while

```
while x:
    if match(x[0]):
        print('ni')
        break
    x = x[1:]
else:
    print('not found')
```

Pętle for

Pętle for w Pythonie służą do iteracji po obiektach iterowalnych (np. listy, zakresy, sekwencje, itd. . .).

Pętle for - ogólny format

```
for <cel> in <obiekt>: # przypisanie kolejnych elementów z
    <instrukcje1>
    if <test1>:
        break
    if <test2>:
        continue
else:
    <instrukcje2>
```

Pętle for - przykłady

```
for part in ['KN', 'Pythona']:
    print(part)

for x in range(100):
    print(x ** 2)
```

Pętle for - przykłady

```
for (a, b) in [(1, 2), (3, 4), (5, 6)]:
    print(a, b)

d = {'a': 1, 'b': 2, 'c': 3}
for key, value in d.items():
    print(key, value, d[key], value == d[key])
```

Pętle for - przykłady

```
for line in open('textfile.txt'):
    print(line)

for c in 'kn pythona':
    print(c, ord(c))
```

Pętle for - zakresy

Instrukcja range(, <stop + 1>,)

Przechodzenie równoległe - zip

```
l1 = [1, 2, 3, 4]
l2 = [5, 6, 7, 8]
for (x, y) in zip(l1, l2):
    print(x, y)
```

Przechodzenie równoległe - map

```
s = 'kn pythona'
list(map(ord, s))
```

Tworzenie słowników - zip

```
keys = ['a', 'b', 'c']
values = [1, 2, 3]
d = dict(zip(keys, values))
```

Iteracja po sekwencji z pozycją - enumerate

```
for index, item in enumerate(sequence):
    print(index, item)
```

Protokół iteracyjny Pythona

Protokół iteracyjny:

- iterator <=> obiekt zwracany przez funkcję iter()
- obiekt iterowany <=> obiekt obsługujący metodę next(I) #**next**()
- na koniec iteracji obiekt iterowany zwraca wyjątek StopIteration

Protokół iteracyjny Pythona

```
l = [1, 2, 3]
i = iter(l)
i.next() # 1
i.next() # 2
i.next() # 3
i.next() # StopIteration
```

Protokół iteracyjny Pythona

```
l = [1, 2, 3]
iter(l) is l # False
f = open('text.txt')
iter(f) is f # True
```

Protokół iteracyjny Pythona - ogólna postać

```
i = iter(obj)
while True:
    try:
        x = next(i)
    except StopIteration:
        break
    <instrukcje>

for x in obj:
    <instrukcje>
```

Protokół iteracyjny Pythona

```
from random import randrange

class RandomSequence:
    def __next__(self):
        x = randrange(12, 100)
        if x > 98:
            raise StopIteration
        else:
            return x
    def __iter__(self):
        return self
```

Listy składane

Ang. List comprehension

```
l = [x ** 2 for x in range(100)]
l = []
for x in range(100):
    l.append(x ** 2)
```

Rozszerzona składnia list składanych

```
even_squares = [x ** 2 for x in range(100) if not x % 2]

double_iter = [x * y for x in range(10) for y in range(10)]
```

Narzędzia iteracyjne

```
sum(seq)
any(seq)
all(seq)
max(seq)
min(seq)
list(seq)
tuple(seq)
"string".join(seq)
a, *b = seq
```

Zbiory i słowniki składane

```
s = {x * y for x in range(10) for y in range(10)}

numbered = {x: y for x, y in enumerate(range(100))}
```

Iteratory map(), zip(), filter()

```
list(map(abs, (-1, 0, 1)))

list(zip(l1, l2))

list(filter(bool, [1, 0, -1])) # [1, -1]
```

Dokumentacja Pythona

Forma	Rola
Komentarz ze znakiem #	Dokumentacja w pliku
Funkcja dir	Lista atrybutów w obiekcie
Łańcuchy znaków dokumentacji	Dokumentacjaw pliku
doc	dołączana do obiektów
PyDoc funkcja help	Interaktywna pomoc dla obiektów
PyDoc raporty HTML	Dokumentacja w przeglądarce

Dokumentacja funkcji(metod)

```
def function(arg1, args):
    """
    Make things.

    Args:
        arg1 (type) arg1 description.
        arg2 (type) arg2 description.

    Returns:
        return value
    """
    <instrukcje>
```