Predavanje04 - Variable Scope

March 24, 2022

1 Variable scope

Spremenljivke se razlikujejo tudi po tem koliko dolgo obstajajo (variable lifetime) in od kje lahko dostopamo do njih (variable scope).

Spremenljivka definirana znotraj funkcije (kot parameter ali navadno) obstaja samo znotraj funkcije.

Ko se izvajanje funkcije konča, spremenljivka neha obstajati.

[1]:
```
def funkcija(spr1):
    spr2 = 10
    print(f"Spr1: {spr1}")
    print(f"Spr2: {spr2}")


funkcija(5)
print(f"Spr1: {spr1}")
print(f"Spr2: {spr2}")
```

```
Spr1: 5
Spr2: 10
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\Users\FAKULT~1\AppData\Local\Temp/ipykernel_17268/2660599479.py in <module>
      6
      7 funkcija(5)
----> 8 print(f"Spr1: {spr1}")
      9 print(f"Spr2: {spr2}")

NameError: name 'spr1' is not defined
```

Spremenljivka definirana znotraj naše glavne kode (zunaj naših funkcij) je **globalna spremenljivka** in je dostopna skozi našo celotno kodo.

[73]:
```
spr1 = 5
print(f"Spr1: {spr1}")
```

```
if spr1 == 5:
    spr2 = 10
print(f"Spremenljivka2: {spr2}")
print()

def funkcija():
    spr3 = 200
    print(f"Spr1: {spr1}")
    print(f"Spr2: {spr2}")
    print(f"Spr3: {spr3}")

funkcija()
print()

print(f"Spr1: {spr1}")
print(f"Spr2: {spr2}")
```

```
Spr1: 5
Spremenljivka2: 10

Spr1: 5
Spr2: 10
Spr3: 200

Spr1: 5
Spr2: 10
```

Problem se lahko pojavi, če znotraj funkcije definiramo spremenljivko z enakim imenom, ki že obstaja kot globalna spremenljivka.

V tem primeru bo python spremenljivki označil kot dve različni spremenljivki. Ena dostopna znotraj funkcije, druga dostopna zunaj funkcije.

[164]:
```
spr1 = 5
print(f"Spr1: {spr1}")

def funkcija():
    spr1 = 100
    print(f"Spr1: {spr1}")

funkcija()
print(f"Spr1: {spr1}")
```

```
Spr1: 5
Spr1: 100
Spr1: 5
```

Parameter se obnaša kot lokalna spremenljivka.

[175]:
```
spr1 = 5
print(f"Spr1: {spr1}")

def funkcija(spr1):
    print(f"Spr1: {spr1}")

funkcija(100)
print(f"Spr1: {spr1}")
```

```
Spr1: 5
Spr1: 100
Spr1: 5
```

Paziti je potrebno, ko posredujemo list ali dictionary kot argument.

[74]:
```
def funkcija(l):
    print(l)
    l[0] = 100

seznam = [3, 7, 13]
funkcija(seznam)
print(seznam)
```

```
[3, 7, 13]
[100, 7, 13]
```

[75]:
```
def funkcija(d):
    print(d)
    d["a"] = 100

dict_ = {"a": 5, "b": 6, "c": 7}
funkcija(dict_)
print(dict_)
```

```
{'a': 5, 'b': 6, 'c': 7}
{'a': 100, 'b': 6, 'c': 7}
```

[ ]:
```

```

Če želimo spreminjati globalno spremenljivko znotraj funkcije (znotraj local scope) moramo uporabiti besedo **global**.

[76]:
```
spr1 = 5
print(f"Spr1: {spr1}")

def funkcija():
    global spr1
    spr1 = 100
```

```
    print(f"Spr1: {spr1}")

funkcija()
print(f"Spr1: {spr1}")
```

```
Spr1: 5
Spr1: 100
Spr1: 100
```

S to besedo lahko tudi ustvarimo novo globalno spremenljivko, znotraj localnega scopa.

[77]:
```
def funkcija():
    global spr1
    spr1 = 5
    print(f"Spr1: {spr1}")

funkcija()
print(f"Spr1: {spr1}")
```

```
Spr1: 5
Spr1: 5
```

[ ]:

Naloga:

Napišite funkcijo, kjer lahko igramo vislice.

Funkcija vislice() naj ima 2 parametra. Prvi je besedo katero se ugiba in drugi število možnih ugibov. Če števila ugibov ne podamo naj bo default vrednost 10.

Uporabnika konstantno sprašujte naj vnese črko. Nato izpišite iskano besedo. Črke katere je uporabnik uganil izpišite normalno, črke katere še ni uganil pa nadomestite z _.

Dodatno zraven prikazujte katere vse črke je uporabnik že preizkusil.

Če uporabnik besedo uspešno ugani v danih poizkusih naj funkcija vrne vrednost True. V nasprotnem primeru naj vrne vrednost False.

Primeri:

```
Input:
vislice("jabolko")

Output:
Guesses so far [].
What is your guess? a
_ a_ _ _ _ _

Guesses so far ['a'].
What is your guess? e
_ a_ _ _ _ _
```

```
Guesses so far ['a', 'e'].
What is your guess? o
_ a_ o_ _ o

Guesses so far ['a', 'e', 'o'].
What is your guess? p
_ a_ o_ _ o

Guesses so far ['a', 'e', 'o', 'p'].
What is your guess? r
_ a_ o_ _ o

Guesses so far ['a', 'e', 'o', 'p', 'r'].
What is your guess? l
_ a_ ol_ o

Guesses so far ['a', 'e', 'o', 'p', 'r', 'l'].
What is your guess? k
_ a_ olko

Guesses so far ['a', 'e', 'o', 'p', 'r', 'l', 'k'].
What is your guess? j
ja_ olko

Guesses so far ['a', 'e', 'o', 'p', 'r', 'l', 'k', 'j'].
What is your guess? b
jabolko
KONEC
True
```

[4]:
```python
# Rešitev
def vislice(beseda, n=10):
    correct_guesses = []
    all_guesses = []

    try_ = 0
    while try_ < n:
        print()
        guess = input(f"Guesses so far {all_guesses}. \nWhat is your guess? ")
        all_guesses.append(guess)
        if guess in beseda:
            correct_guesses.append(guess)

        beseda_print = ""
        for ch in beseda:
            if ch in correct_guesses:
                beseda_print += ch
```

```
            else:
                beseda_print += "_ "
        print(beseda_print)
        if len(set(correct_guesses)) == len(set(beseda)):
            print("KONEC")
            return True

        try_ += 1

    return False


print(vislice("jabolko"))
```

```
Guesses so far [].
What is your guess? j
j_ _ _ _ _ _

Guesses so far ['j'].
What is your guess? a
ja_ _ _ _ _

Guesses so far ['j', 'a'].
What is your guess? b
jab_ _ _ _

Guesses so far ['j', 'a', 'b'].
What is your guess? o
jabo_ _ o

Guesses so far ['j', 'a', 'b', 'o'].
What is your guess? l
jabol_ o

Guesses so far ['j', 'a', 'b', 'o', 'l'].
What is your guess? p
jabol_ o

Guesses so far ['j', 'a', 'b', 'o', 'l', 'p'].
What is your guess? e
jabol_ o

Guesses so far ['j', 'a', 'b', 'o', 'l', 'p', 'e'].
What is your guess? k
jabolko
KONEC
```

```
True
```

[ ]:

Naloga:

Ustvarite program Križci in Krožci

Igralno polje lahko predstavite kot liste znotraj lista, kjer E predstavlja prazno polje.

```
board = [["X", "E", "E"],
         ["O", "E", "E"],
         ["E", "E", "E"]]
```

Od igralcev nato izmenično zahtevajte polje v katerega želijo postaviti svoj znak. Privzememo lahko, da bodo igralci igrali pravično in vpisovali samo prazna polja.

Primeri:

```
Output:
['E', 'E', 'E']
['E', 'E', 'E']
['E', 'E', 'E']
It's X's turn. Make a move (exp: 12): '00

['X', 'E', 'E']
['E', 'E', 'E']
['E', 'E', 'E']
It's O's turn. Make a move (exp: 12): '12

['X', 'E', 'E']
['E', 'E', 'O']
['E', 'E', 'E']
It's X's turn. Make a move (exp: 12): '10

['X', 'E', 'E']
['X', 'E', 'O']
['E', 'E', 'E']
It's O's turn. Make a move (exp: 12): '12

['X', 'E', 'E']
['X', 'E', 'O']
['E', 'E', 'E']
It's X's turn. Make a move (exp: 12): '20
X je ZMAGOVALEC!
```

[ ]:

```
def display_board(board):
    for row in board:
        print(row)
```

```
def make_move(on_turn, board):
    move = input(f"It's {on_turn}'s turn. Make a move (exp: 12): '")
    row = int(move[0])
    col = int(move[1])
    board[row][col] = on_turn


def is_game_over(board):
    for row in board:
        if row[0] != "E":
            if row[0] == row[1] and row[0] == row[2]:
                return True

    for i in range(3):
        if board[0][i] != "E":
            if board[0][i] == board[1][i] and board[0][i] == board[2][i]:
                return True

    if board[0][0] != "E":
        if board[0][0] == board[1][1] and board[0][0] == board[2][2]:
            return True

    if board[0][2] != "E":
        if board[0][2] == board[1][1] and board[0][2] == board[2][0]:
            return True

    return False

def play():
    board = [["E", "E", "E"],
             ["E", "E", "E"],
             ["E", "E", "E"]]
    on_turn = "X"
    while True:
        display_board(board)
        make_move(on_turn, board)

        game_over = is_game_over(board)
        if game_over:
            print(f"{on_turn} je ZMAGOVALEC!")
            break
        else:
            if on_turn == "X":
                on_turn = "O"
            elif on_turn == "O":
                on_turn = "X"
        print()
```

```
play()
```

[ ]:

[ ]:

[ ]: