Предавање 3

Python

Контрола на тек на извршување - наредби и циклуси

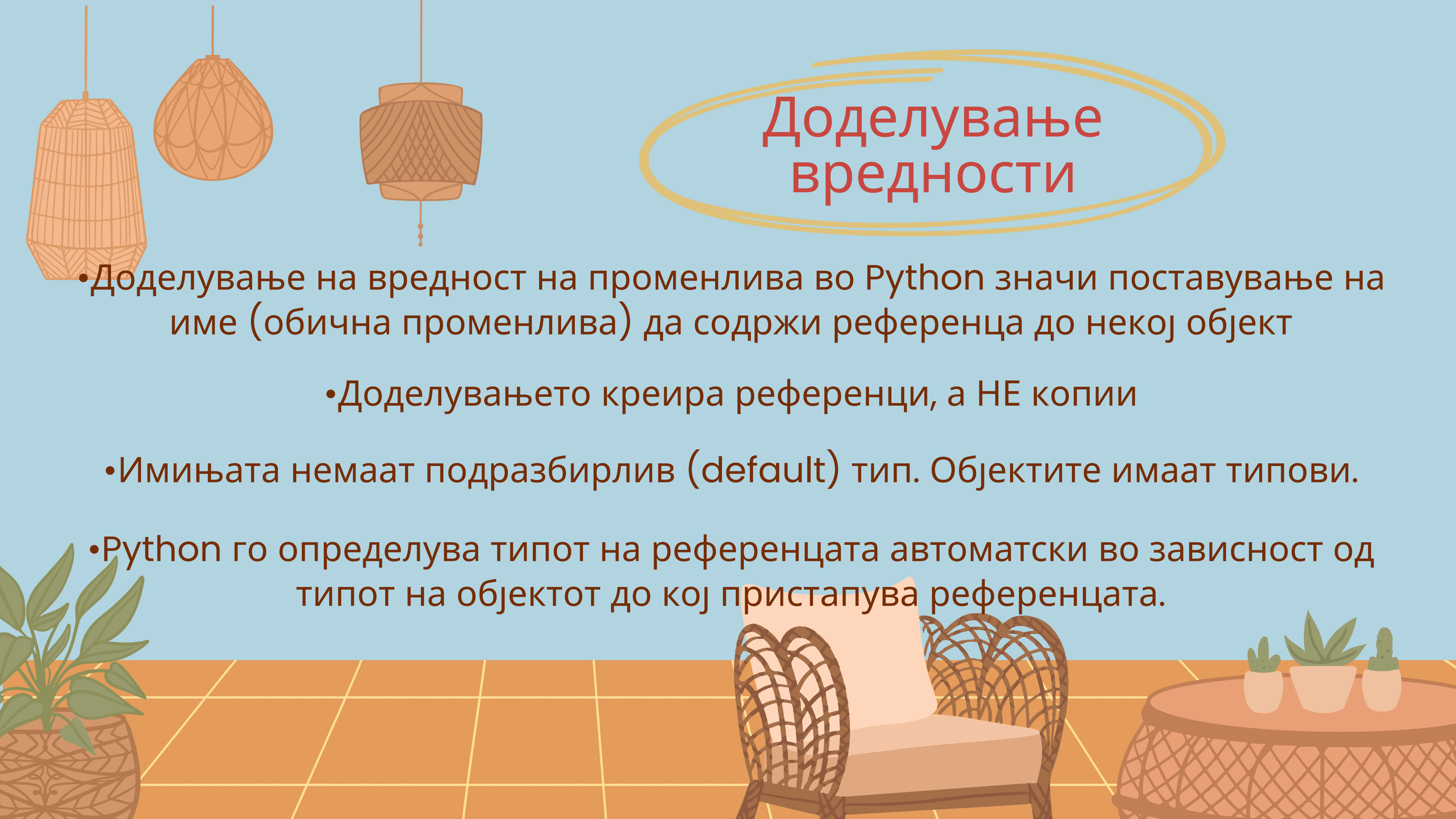
На денешниот час:

1. Доделување вредности
2. Коментари
3. Празен простор (Whitespace)
4. If , if else
5. Тернарен оператор
6. While
7. For циклуси

python

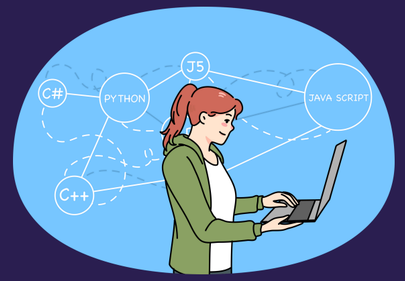
Доделување вредности

- Доделување на вредност на променлива во Python значи поставување на име (обична променлива) да содржи референца до некој објект
- Доделувањето креира референци, а НЕ копии
- Имињата немаат подразбирлив (default) тип. Објектите имаат типови.
- Python го определува типот на референцата автоматски во зависност од типот на објектот до кој пристапува референцата.



ПОВЕЌЕКРАТНО ДОДЕЛУВАЊЕ

```
>>> x, y = 2, 3
>>> x
2
>>> y
3
```



Може да се доделуваат вредности на повеќе променливи истовремено.

Доделување вредности

За едноставни вградени податочни типови (integer, float, стрингови), доделувањето се однесува како што очекуваме:

```
>>> x = x + 1          # Creates 3, name x refers to
>>>print( x ))         # No effect on x, still ref
3
>>> y = x              # Creates name y, refers to
>>> y = 4              # Creates ref for 4. Changes
```

КОМЕНТАРИ

•Коментарите започнуваат со # остатокот од линијата се игнорира.

•Може да се постават “документациски стрингови” како прва линија на која било нова функција или класа што се дефинира.

•Ова се употребува од развојните околини, дебагерот и други алатки и е добар стил да се поставува документациска линија:

```
def my_function (x , y):
    """Ova e dokumentaciskiot
    string. Ovaa funkcija sluzhi za
    bla bla bla. """
    # Kodot na funkcijata sleduva ovde...
```

python

ПРАЗЕН ПРОСТОР (WHITESPACE)

- Празниот простор има значење во Python: особено порамнувањето и новите линии
- Различно во однос на многу други јазици
- Нова линија означува крај на наредба!!
  - Со \ може да се продолжи наредбата во повеќе линии
- Не се потребни загради за да се означат блокови на код
- Наместо тоа, се употребува конзистентно порамнување!
  - Првата линија со помало вовлекување (indentation) е надворешниот блок
  - Првата линија со поголемо вовлекување (indentation) започнува вгнезден блок
- Често две точки (:) се појавуваат на почетокот на нов блок (пр. за декларција на класи и функции)

IF

```
x = 100
if x :        # equal to if(x != 0) :
    print ( "1 - Got a true expression value. " )
    print (x)
y = 0
if y:         # equal to if(y != 0) :
    print ( "2 - Got a true expression value. " )
    print (y)
    print ( "Good bye" )
```

IF-ELSE

```
x = 100
if x :        # equal to if(x != 0) :
    print ( "Got a true expression value. ")
    print (x)
else:
    print ( "Got a false expression value. ")
    print (x)
    print ( "Good bye")
```

1

Забелешка

- Користете порамнување за блокови од наредби
- Две точки (:) после логички (boolean) изрази

2

Забелешка

- По клучниот збор else задолжително се поставуваат две точки
- Може да има најмногу еден else блок придружен на дадена if наредба else блокот е опционален

Тернарен оператор

•Тернарните оператори во Python се познати како условни изрази (conditional expressions).

•Овие оператори евалуираат нешто во зависност дали условот е вистинит или не.

•Дозволува брза проверка на условот, наместо повеќелиниски if израз .

•Најчесто може да биде значајно важен и може да го направи кодот
компактен, а сепак одржлив.

```
fruit = 'apple'
is_apple = True if fruit == 'apple' else False
```

while

```
>>> x = 3
>>> while x < 5:
        print(x, "still in the loop")
        x = x + 1
3 still in the loop
4 still in the loop
>>> x = 6
>>> while x < 5:
        print(x, "still in the loop")
>>>
```

Може да дефинирате и else дел за while наредбата доколку завршувањето на циклусот треба да резултира во специфична операција.

```
>>> x = 3
>>> while x < 5:
        print(x, "still in the loop")
        x = x + 1
    else: print(x, "out of the loop")
3 still in the loop
4 still in the loop
5 out of the loop
```

FOR ЦИКЛУСИ

• For циклусот ги изминува сите елементи од некоја колекција, или било каков друг податочен тип низ кој може да се итерира.

```
for <item> in <collection>:
    <statements>
```

• Ако <collection> е листа или торка, тогаш <statements> се извршуваат за секој елемент на секвенцата.
• Ако <collection> е стринг, тогаш <statements> се извршуваат за секој знак во стрингот .

```
for char in "Hello world":
    print(char)
```

For циклуси

```
for <item> in <collection>:
    <statements>
```

- <item> може да има и посложена структура од име на една променлива .
  - Кога и самите елементи на <collection> се некакви секвенци, тогаш <item> може да има иста структура како тие елементи .
  - Ваквото „повеќекратно“ доделување може да го олесни пристапот до поединечните делови на сложената структура на елементите.

```
>>> for (x, y) in [('a', 1), ('b', 2), ('c', 3), ('d', 4)]:
        print(x)
```

```
a
b
c
d
```




print("Thank you for listening")
love