Vstup a výstup v Pythonu

Funkce

`print(obsah)`

- tisk obsahu
- obsahem může být více položek oddělených čárkami
- příklady:
  - `print("hello world")`
  - `print("Delka je",delka, "cm")`
  - `print("Odjezdy\nTyn \t7:40\nTabor\t8:30")`
- escape sekvence:
  - \n – nový řádek
  - \t – tabelátor

input(text)

- Interaktivní vstup proměnné
- Vstupem je vždy řetězec, potřebujeme-li číslo je nutná konverze (viz dále)
- Příklad:
  - `a = input("Zadej a: ")`
  - `cislo = eval(input("Zadej hodnotu "))`

Práce s řetězci

`2+3` – odpověď `5` (číslo)

ale

`"2"+"3"` – odpověď `"23"` (řetězec) – tzv. zřetězení

Konverze řetězec na číslo a naopak

Číslo na řetězec:

`retezec = str(cislo)`

Řetězec na číslo

`cislo = int(retezec)` (celé číslo)

`cislo = double(retezec)` (reálné číslo)

pokud je řetězec matematickým výrazem (libovolným číslem) stačí i:

`cislo = eval(retezec)`

Např:

`eval(1/2)` dá odpověď `0.5`

Druhá odmocnina

Na začátku programu:

```
from math import sqrt
```

A pak lze použít funkci sqrt např.:

```
odmocnina = sqrt(2)
```

Práce s grafikou pomocí knihovny tkinter

Zavedení knihovny

```
import tkinter
```

Vytvoření nového okna

```
okno = tkinter.Canvas()
okno.pack()
```

Volitelně lze nastavit i velikost okna v pixelech:

```
tkinter.Canvas(width=X,height=Y)
```

Vytvoří se nový objek okno ze třídy tkinter. Nyní mu posíláme zprávy (metody), které mu říkají, co má kreslit.

V následujících příkazech používáme soustavu souřadnou s počátkem v levém horním rohu okna (0,0), kde *x* se zvětšuje směrem vpravo a *y* směrem dolů. Pokud jsme použili tkinter`.Canvas(width=X,height=Y)` pak souřadnice pravého dolního rohu jsou (X,Y).

Kreslení obdélníku (čtverce):

```
okno.create_rectangle(X1,Y1,X2,Y2,volitelně další parametry)
```

(X1,Y1) – souřadnice levého horního rohu

(X2,Y2) – souřadnice pravého dolního rohu

Další parametry – nastavují chování kresleného objektu, např:

outline=“red“

fill=“green“

Např tedy:

```
okno.create_rectangle(20,30,60,90,fill="yellow",out="blue")
```

Kreslení elipsy (kruhu):

```
okno.create_oval(X1,Y1,X2,Y2, volitelně další parametry)
```

Elipsa (kruh) se nakreslí dovnitř obdélníku (čtverce) jehož levý horní roh (X1,Y1) a pravý dolní roh je (X2,Y2). Elipsa (kruh) se tomuto obdélníku (čtverci) vepíše. Jedná-li se o obdélník, pak vznikne elipsa, jedná-li se o čtverec vznikne kruh.

Pokud dáme nulovou velikost (X1=X2 a Y1=Y2) vznikne bod.

Kreslení mnohoúhelníku

```
okno.create_polygon(minimálně tři body, další parametry)
```

Mnohoúhelník vznikne tak, že se body postupně pospojují, přičemž poslední se spojí s prvním. Je nutné tedy zadat minimálně tři body (vrcholy), pak vznikne trojúhelník.

Např.:

```
okno.create_polygon(10,10,100,50,10,90,fill="Red")
```

Psaní textu

```
okno.create(pozice,text="text k zobrazení")
```

pozice X,Y počátku

Např.:

```
okno.create_text(50,50,text="Hello world")
```

Poznámka

Pozor při kopírování textu do Pythonu na správné uvozovky. Mezery mezi parametry v programu Idle mohou a nemusí být (jiné programy je mohou vyžadovat).

Podmíněný příkaz

Provede se pouze je-li splněna podmínka

Základní tvar

```
if (podmínka):
    příkaz1
    příkaz2
příkaz3
```

Příkazy 1 a 2 se provedou pouze je-li splněna podmínka, příkaz 3 vždy. Pro odsazení se používají **čtyři mezery**. Většina editorů pozná podmíněný příkaz díky dvojtečce na konci podmínky a odsazení zařídí sama. Konec podmínky je dán ukončením odsazování.

Příklad:

```
if (a<>0):
    b = 1 /a
    print(b)
```

Předchozí podmínku lze psát pouze

```
if (a):
```

Pokud totiž výraz v závorce (hodnota a) je různý od nuly je podmínka brána jako splněná. Pokud je roven nule pak jako nesplněná.

Používá se:

< - menší

> - větší

== - rovno (neplést s = přiřazovací příkaz)

!= - nerovno

<= - menší nebo rovno

>= - větší nebo rovno

Logické spojka **and** (a zároveň), **or** (nebo), **not** (negace , ne)

Příklad:

```
if ((a>1) and (a<2)):
```

totéž je:

```
if not((a<=1) or (a>=2)):
```

Poznámka – závorky nejsou povinné, ale pomáhají v přehlednosti

Rozšířený podmíněný příkaz

```
if (podminka1):
    příkazy
else:
    příkazy
```

Není-li splněna podmínka provedou se příkazy za else. Příklad

```
if (a):
    b = 1/a
    print("b je ",b)
else:
    print("Nelze dělit nulou")
```

Příkaz lze dále rozvětvit pomocí části elif, ktero lze opakovat libovolně:

```
if (podminka1):
    příkazy
elif                                                    (podminka2):
    příkazy
elif(podminka3):
    příkazy
else:
   příkazy
```

Poznámka – část s else není povinná

Poznánka – nezapomínat na dvojtečku za podmínkou nebo else – častá chyba

Podprogramy

Definice

def podprogram():

příkaz 1

příkaz 2

příkaz 3

Poznámky:

1. pozor na syntaxi – závorky a dvojtečka
2. podprogram musí být definován před prvním použitím
3. pozor hodnota proměnných se přenáší do podprogramu, ale ne z něj do nadřazeného programu. Vyhněte se stejným jménům

Př:

```
A = 1
def tisk()
    A = 2
    print(A)

A = 0
print(A)
tisk()
print(A)
```

Vytiskne 0, 1, 0

Předání parametrů

Př.

```
def tisk2(A):
    print(A)

print(tisk2(2))
```

Vytiskne 2

Návratová hodnota

Pomocí příkazu return s jedním nebo více proměnnými

return ukončí proceduru a vrátí uvedenou hodnotu

Př:

```
def rovnice(a,b):
    # Reseni rovnice ax+b=0
    if (a):
        return (-b/a)
   else:
        return „Nemá řešení“
print (rovnice(2,1))
```

Pokud nemáme žádnou hodnotu pro návrat můžeme napsat

```
return None
```

Cyklus while

Jedná se o cyklus s proměnným počtem opakování – před začátkem nevíme kolikrát proběhne. Nemusí proběhnout ani jednou

Syntaxe

while (podmínka):

    příkazy

else:

    příkazy

Část za else se provede, pokud není podmínka splněna. Tato část je nepovinná a obvykle se nepoužívá.

Př:

```
i = 0
while (i < 5):
    i = i+1
    print(i)
else:
    print("Konec")
```

Je možné použít příkazy

- continue – skok na začátek cyklu s opětovným vyhodnocením podmínky.
- break – vyskočení z cyklu (část za else se neprovede)

Příklad:

```
i = 0
```

```
while (i < 5):
    i = i+1
    if (i == 2):
        continue
    if (i == 6):
        break
    print(i)
else:
    print("Konec")
```

Pokud by v těle cyklu neměl být žádný příkaz použijte příkaz pass – prázdný příkaz

Např:

```
while neni_pripojeni():
    pass
```

Nekonečný cyklus – jako podmínku použijeme True. Např:

```
while True:
    vstup = input("Zadej matematicky vyraz: ")
    print(eval(vstup))
```

Jediná možnost, jak takový cyklus ukončit je pomocí příkazu break.

Podmíněné spuštění kódu

Pomocí příkazu try:

např. v původním příkazu doplníme, včetně nastavení ukončení:

```
from math import sqrt, sin, cos

while True:
    vstup = input("Zadej matematický výraz (q=konec): ")
    if (vstup=="q"):
        break
    try:
        print(eval(vstup))
    except:
```

```
        pass
```

Část za except znamená, co se má udělat, když v části za try dojde k chybě. V našem případě se neudělá nic a znovu se zadává výraz. ¨Některé chyby umí Python odlišit např:

```
try:
    print(eval(vstup))
except ZeroDivisionError as err:
    print("Nulou nedělíme")
except:
    pass
```

Pro zájemce více např. zde:

https://docs.python.org/3/tutorial/errors.html

Cyklus for

Jedná se o cyklus s pevným počtem opakování – víme kolikrát proběhne

Syntaxe

for promenna in (iterovatelný objekt):

    příkazy

else:

    příkazy

Část za else se provede, pokud není podmínka splněna. Tato část je nepovinná a obvykle se nepoužívá.

iterovatelný objekt – lze jej rozložit na části

range(A) – od 0 do A-1, po jedné

range(A,B) – od A do B-1, po jedné

range(A,B,C) – od A do B-1 po C

Př:

```
for I in range (4):
    print(I)
```

(provede se pro 0,1,2,3)

```
for I in range (2,4):
    print(I)
```

provede se pro 2,3

```
for I in range (2,8,2):
    print(I)
```

provede se pro 2,4,6

```
for I in range (6,3,-1):
    print(I)
```

provede se pro 6,5,4

Je možné použít příkazy

- continue – skok na začátek cyklu s opětovným vyhodnocením podmínky.
- break – vyskočení z cyklu (část za else se neprovede)

Příkaz je možno použít i pro řetězec, např.

```
for znak in ("Ahoj"):
    print(znak)
```

Vytiskne

```
A
h
o
j
```

Celočíselné dělení a zbytek

A // B – celočíselné dělení A děleno B

A % B – zbytek po celočíselném dělení A číslem B

Např:

6 // 3 je 2

6 % 3 je 0

5 //2 je 2

5 % 2 je 1