Vlastnosti jayzka Python

- **Interpretovaný**: kód jazyka sa spracováva postupne za behu programu, takzvaným interpreterom.
- **Interaktívny**: možno písať riadok po riadku program priamo do konzoly, ktorý sa okamžite bude vykonávať.
- **Objektovo orientovaný**: podporuje princípy OOP.
- **Vhodný pre začiatočníkov**: kód sa podobá jednoduchej angličtine.
- **Široké použitie**: databázy, machine learning, web aplikácie.

Komentáre

- slúžia na dokumentáciu kódu, na program nemajú efekt,
- v Pythone existujú:
  - jednoriadkové,
  - blokové.

Jednoriadkoý kometár

- začína znakom: (#).

In [ ]:
```
# toto je jednoriadkový komentár
```

Blokový komentár

- je ohraničený úvodzovkami: (') alebo (").

In [ ]:
```
"""
Toto je
blokový
komentár
"""

'''
Aj toto
je
blokový
komentár
'''
```

Štandardný vstup a výstup

- štadnardný vstup (klávesnica): **input()**.
- štadnardný výstup (konzola): **print()**.

Použitie štandardného výstupu

In [1]:
```
# Príklad použitia štandardného výstupu:
print("Ahoj Svet!")
print(5)
```

```
Ahoj Svet!
5
```

Použitie štandardného vstupu aj výstupu

In [3]:
```
# Príklad použitia štandardného vstupu aj výstupu:
meno = input("Zadajte svoje meno: ")

cislo = float(input())

print("Vaše meno je:", meno)
print(cislo)
```

```
Zadajte svoje meno: aaa
100
Vaše meno je: aaa
100.0
```

Dátové typy

- dynamická typová kontrola - netreba deklarovať typ premennej,
- môžu niesť akúkoľvek hodnotu - celé číslo, reálne číslo reťazec,
- case senzitivita - rozlišujú sa malé a veľké písmená,
- názvy premenných sa môžu skladať iba z písmen, čísel a podčiarkovníkov.
- existujú:
  - jednoduché dátové typy,
  - zložené dátové typy.

Jednoduché dátové typy

- uchovávajú len jednu hodnotu,
- sú to:
  - booleovské hodnoty: **bool**,
  - celé čísla: **integer**,
  - reálne čísla: **float**,
  - komplexné čísla: **complex**.

Poznámka: Znak "char" v Pythone neexistuje.

Vytvorenie jednoduchých dátových typov

In [3]:
```
a = 20      # int
b = 17.5    # float
c = 2+3j    # complex
d = True    # bool

print(a, b, c, d)
```

```
20 17.5 (2+3j) True
```

Operácie s jednoduchými dátovými typmi

- aritmetické: (+, -, *, /, %, //, * *),
- porovnávacie: (<, >, ==, !=, <=, >=),
- pretypovanie: (int(), float(), bool(), complex()),
- logické: (and, or, not),
- bitové: (&, |, ~, ^, <<, >>),
- priraďovacie (+=, -=, *=, ...).

Aritmetické operácie

In [4]:
```
# Príklad aritmetických operácií:
a = 4
b = 3

sucet = a + b
rozdiel = a - b
sucin = a * b
podiel_r = a / b
podiel_c = a // b
mocnina = a ** b
zvysok = a % b

print("súčet: ", sucet)
print("rozdiel: ", rozdiel)
print("súčin: ", sucin)
print("podiel (reálny): ", podiel_r)
print("podiel (celočíselný): ", podiel_c)
print("mocnina: ", mocnina)
print("zvyšok po delení: ", zvysok)
```

```
súčet:  7
rozdiel:  1
súčin:  12
podiel (reálny):  1.3333333333333333
podiel (celočíselný):  1
mocnina:  64
zvyšok po delení:  1
```

Porovnávacie operácie

In [4]:
```
# Príklady porovnávacích operácií:
a = 100
b = 10

print("a > b:", a > b)
print("a < b:", a < b)
print("a == b:", a == b)
print("a != b:", a != b)
print("a >= b:", a >= b)
print("a >= b:", a <= b)
```

```
a > b: True
a < b: False
a == b: False
a != b: True
a >= b: True
a >= b: False
```

Pretypovaie

In [5]:
```
# Príklad pretypovania:

cislo1 = 3.14

cislo2 = int(cislo1) # integer
print("Typ:", type(cislo2), ", Hodnota:", cislo2)

cislo2 = float(cislo1) # float
print("Typ:", type(cislo2), ", Hodnota:", cislo2)

cislo2 = complex(cislo1) # complex
print("Typ:", type(cislo2), ", Hodnota:", cislo2)

cislo2 = bool(cislo1) # bool
print("Typ:", type(cislo2), ", Hodnota:", cislo2)
```

```
Typ: <class 'int'> , Hodnota: 3
Typ: <class 'float'> , Hodnota: 3.14
Typ: <class 'complex'> , Hodnota: (3.14+0j)
Typ: <class 'bool'> , Hodnota: True
```

Logické operácie

In [6]:
```
# Príklady logických operácií:
x = True
y = False

z = x and y          # a && b
print("a and b:", z)

z = x or y           # a || b
print("a or b:", z)

z = not x            # !a
print("not a:", z)
```

```
a and b: False
a or b: True
not a: False
```

Priraďovacie operácie

In [7]:
```
# Príklad priraďovacích operácií:
a = 3
a = a + 2

print(a)

a += 2

print(a)
```

```
5
7
```

Zložené dátové typy

- uchovávajú viacero hodnôt,
- sú to:
  - reťazce: **strings**,
  - zoznamy: **lists**,
  - množiny: **sets**,
  - n-tice: **tuples**,
  - slovníky: **dictionaries**.

String

- skupina znakov,
- **nie je modifikovateľný**,
- **znaky sú číslované od 0**.

List

- usporiadaná skupina prvkov,
- prvky môžu byť **rovnakého i rôzneho dátového typu**,
- je modifikovateľný,
- sú povolené duplicity,

- **prvky sú číslované od 0**.

Vytvorenie zoznamov a reťazcov

In [8]:
```
# Príklad vytvorenia reťazcov:
prazdny = ""
slovo = "Ahoj Svet"

print("Reťazce:")
print(prazdny)
print(slovo)

# Príklad vytvorenia zoznamov:

prazdny = []
cisla = [1, 2, 3, 4, 5]
mix = ["Jablko", 1, 2, True, 3+1j]


print("\n")
print("Zoznamy:")
print(prazdny)
print(cisla)
print(mix)
```

```
Reťazce:

Ahoj Svet


Zoznamy:
[]
[1, 2, 3, 4, 5]
['Jablko', 1, 2, True, (3+1j)]
```

Operácie so zloženými dátovými typmi

- zlučovanie: (+, *),
- porovnanie dĺžky reťazcov: (<, >, ==, !=, <=, >=),
- vyhľadávanie prvkov: (in, not in),
- indexovanie: (pre list, string, tuple),
- pretypovanie: (str(), list(), ...),
- pridávanie / mazanie prvkov: (pre list, set, dictionary),
- modifikácia prvkov: (pre list, dictionary),
- užitočné funkcie, špecifické pre konkrétny dátový typ.

Zlučovanie

In [9]:
```
# Príklady zlučovania reťazcov:
meno = "Janko" + "Hraško"
cislo = "1" + "2"

print("Reťazce:")
print(meno)
print(cislo)

# Príklady zlučovania zoznamov:
cisla = [10, 20, 30]
ovocie = ["jablko", "hruška", "slivka"]

retazec = ["a", "h", "o", "j"]

mix = cisla + ovocie
cisla_3x = cisla * 3

print("\n")
print("Zoznamy:")
print("Mix: ", mix)
print("Cisla 3x: ", cisla_3x)
```

```
Reťazce:
JankoHraško
12


Zoznamy:
Mix:  [10, 20, 30, 'jablko', 'hruška', 'slivka']
Cisla 3x:  [10, 20, 30, 10, 20, 30, 10, 20, 30]
```

Porovnanie dĺžky

In [10]:
```
# Príklady porovnávania dĺžky reťazcov:
r1 = "pomaranč"
r2 = "kiwi"

print("r1 > r2:", r1 > r2)
```

```
r1 > r2: True
```

Vyhľadávanie

In [12]:
```
# Príklady vyhľadávania v reťazcoch:
meno = "Janko Hraško"

print("Reťazce:")
print("Je Janko v reťazci meno? ", "Janko" in meno)

# Príklady vyhľadávania v zoznamoch:
cisla = [1, 2, 3, 4, 5]

print("\n")
print("Zoznamy:")
print("Chýba číslo 6 v zozname cisla? ", 6 not in cisla)
```

```
Reťazce:
Je Janko v reťazci meno?  True


Zoznamy:
Chýba číslo 6 v zozname cisla?  True
```

Indexovanie

In [13]:
```
# Príklady indexovania reťazcov:
meno = "Janko Hraško"

print("Reťazce:")
print(meno[0])
print(meno[2:4])
print(meno[:4])
print(meno[1:])

# Príklady indexovania zoznamov:
ovocie = ["jablko", "hruška", "slivka"]

print("\n")
print("Zoznamy:")
print(ovocie[0])
print(ovocie[0:2])
print(ovocie[:2])
print(ovocie[1:])
```

```
Reťazce:
J
nk
Jank
anko Hraško


Zoznamy:
jablko
['jablko', 'hruška']
['jablko', 'hruška']
['hruška', 'slivka']
```

1. Nacitanie 5 cisel
2. ulozit ich do pola
3. vypocitat priemer

Pretypovanie

In [14]:
```
# Príklad pretypovania - súčet cifier:

cislo = 945

cifry = str(cislo)    # pretypovanie na reťazec

cifra1 = int(cifry[0]) # pretypovanie znakov na čísla
cifra2 = int(cifry[1])
cifra3 = int(cifry[2])

vysledok = cifra1 + cifra2 + cifra3

print(vysledok)
```

```
18
```

Pridávanie, mazanie a modifikácia prvkov

In [15]:
```
# Príklady pridávania, mazania a modifikácie zoznamov:
ovocie = ["jablko", "hruška", "slivka"]
print("Pôvodný zoznam:", ovocie)

# modifikácia prvku
ovocie[0] = "mango"      # modifikovanie prvku na 0-tom indexe
print("Modifikácia prvku:", ovocie)

# pridanie prvkov
ovocie.append("banán")    # pridanie prvku na koniec zoznamu
ovocie.insert(0, "kiwi")  # pridanie prvku na 0-tú pozíciu
print("Pridanie prvkov:", ovocie)

# mazanie prvkov
ovocie.pop(1)              # vzmazanie prvku s indexom 1
ovocie.remove("kiwi")      # vymazanie prvku s hodnotou "kiwi"
print("Mazanie prvkov:", ovocie)
```

```
Pôvodný zoznam: ['jablko', 'hruška', 'slivka']
Modifikácia prvku: ['mango', 'hruška', 'slivka']
Pridanie prvkov: ['kiwi', 'mango', 'hruška', 'slivka', 'banán']
Mazanie prvkov: ['hruška', 'slivka', 'banán']
```

Špecifické funkcie

In [16]:
```
# Príklad špecifickej funkcie reťazca:
jablko = "jablko"
jablko = jablko.upper()

print("Velke jablko:", jablko)

# Príklad špecifickej funkcie zoznamu:
ovocie = ["jablko", "hruška", "slivka"]
ovocie.sort()

print("Zoradné ovocie:", ovocie)
```

```
Velke jablko: JABLKO
Zoradné ovocie: ['hruška', 'jablko', 'slivka']
```

Podmienky

- predstavujú rozhodovanie programu,
- kľúčové slová: **if**, **elif**, **else**,
- jednotlivé vetvy musia byť odsadené: **1x TAB**.

Jednochá podmienka

In [4]:
```
# Príklad jednoduchej podmienky - znamieko čísla:

cislo = int(input("Zadajte číslo: "))

if cislo > 0:
    print("kladné")
    print("kladné")
    print("kladné")
elif cislo < 0:
    print("záporné")
else:
    print("nula")
```

```
Zadajte číslo: 1
kladné
```

Vnorená podmienka

In [40]:
```
# Príklad vnorenej podmienky - znamieko čísla 2.0:

cislo = int(input("Zadajte číslo: "))


if cislo > 0:
    print("kladné")

    if cislo > 1000:    # vnorená podmienka
        print("veľké")
elif cislo < 0:
    print("záporné")
else:
    print("nula")
```

```
Zadajte číslo: 9047
kladné
veľké
```

Jednoriadková podmieka (ternárny operátor)

In [41]:
```
# Príklad jednoriadkovej podmienky - znamieko čísla 3.0:
# nepočítame so vstupom 0

cislo = int(input("Zadajte číslo: "))

vysledok = "kladné" if cislo > 0 else "záporné"

print(vysledok)
```

```
Zadajte číslo: 10
kladné
```

Cykly

- umožňujú opakovať tie isté príkazy viackrát za sebou,
- dva rôzne typy: **for** a **while** (v Pythone neexistuje do..while),
- Python podporuje príkazy: **break** a **continue**,
- telo cyklu musí byť odsadené: **1x TAB**.

For cyklus

- cyklus s **pevným počtom opakovaní** – dopredu vieme povedať, koľkokrát sa kód vykoná,
- slúži najmä na **prechádzanie dátových štruktúr**, ako sú reťazce, zoznamy, n-tice, množiny a slovníky.

Prechádzanie dátových štruktúr

In [2]:
```
# Príklad použitia cyklu for na prechádzanie zložených dátových typov:
# Súčet prvkov.

cisla = [1, 2, 3, 4, 5]
vysledok = 0

for i in range(0, 5):
    vysledok += cisla[i]

    print(vysledok)

print()

for element in cisla:
    print(element)
```

```
1
3
6
10
15

1
2
3
4
5
```

Vytvorenie postupností (zoznamov)

In [3]:
```
# Príklad vytvorenia postupnosti od 0 do 9 s využitím cyklu for:

postupnost1 = []

for prvok in range(1, 10, 1):
    postupnost1.append(prvok)

print("postupnosť č. 1", postupnost1)

# zjednodušená konštrukcia:

postupnost2 = [prvok for prvok in range(1, 10, 2)]

print("postupnosť č. 2", postupnost2)
```

```
postupnosť č. 1 [1, 2, 3, 4, 5, 6, 7, 8, 9]
postupnosť č. 2 [1, 3, 5, 7, 9]
```

While cyklus

- cyklus s podmienkou na začiatku,
- slúži na vykonávanie tých istých príkazov, pokiaľ je splnená **riadiaca podmienka**.

Použitie cyklu while

In [46]:
```
# Príklad použitia cyklu while:
# Výpis postupnosti prvkov od 0 do 9.

x = 0

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

```
0
1
2
3
4
5
6
7
8
9
```

Použitie cyklu while s využitím break a continue

In [47]:
```
# Príklad použitia break a continue v cykle while:
# Výpis postupnosti: iteruj od 0 do 9, vynechaj číslo 5.
z = 0

while True:        # podmienka je vždy splnená
    if z == 5:     # vynechaj číslo 5
        z += 1
        continue
    if z == 10:  # ukonči cyklus ak dosiahneš číslo 10
        break
    print(z)
    z += 1
```

```
0
1
2
3
4
6
7
8
9
```

Vnáranie cyklov

In [17]:
```python
# Príklad použitia vnorených cyklov:
# Výpis matice.

matica = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

for riadok in matica:
    for stlpec in riadok:
        print(stlpec, end=", ")
    print()  # nový riadok
```

```
1, 2, 3,
4, 5, 6,
7, 8, 9,
```

Funkcie

- sú bloky kódu, vykonávajúce určitú úlohu,
- kľúčové slová: **def** a **return**,
- môžu mať ľubovolný počet parametrov (aj žiadne),
- môžu vracať ľubovolný počet návratových hodnôt (aj žiadnu),
- telo funkcie musí byť odsadené: **1x TAB**,
- funkcie v Pythone podporujú rekurziu.

In [4]:
```python
# Príklady použitia funkcií:

# funkcia bez parametrov a návratovej hodnoty:
def ahojSvet():
    print("Ahoj Svet")

# funkcia s 2-mi parametrami a bez návratovej hodnoty:
def vypisMeno(x, y):
    print("Meno:", x, "Priezvisko:", y)

# funkcia, čo vracia súčet čísel x a y (2 parametre, 1 návratová hodnota):
def sucetCisel(x, y):
    return x + y

# funkcia, čo vracia viacero hodnôt (2 parametre, 2 návratové hodnoty):
def vratSucetRozdiel(x, y):
    return x + y, x - y

def vrat_priemer(pole):
    sucet = 0

    for element in pole:
        sucet += element

    return sucet / len(pole)

# volanie funkcií:
ahojSvet()
vypisMeno("Janko", "Hraško")

pole = [1, 2, 3, 4, 5, 10]

A = sucetCisel(10, 40)
S, R = vratSucetRozdiel(10, 40)

print("súčet:", A)
print("súčet:", S, "rozdiel:", R)
print("Priemer: ", vrat_priemer(pole))
```

```
Ahoj Svet
Meno: Janko Priezvisko: Hraško
súčet: 50
súčet: 50 rozdiel: -30
Priemer:  4.166666666666667
```

Výnimky

- sú to udalosti, vyvolané pri chybe programu,
- kľúčové slová **try**, **except**, **finally** a **raise**,
- jednotlivé bloky musia byť odsadené: **1x TAB**.
- môžu byť vyvolané:
  - automaticky,
  - programovo.

Automatické vyvolávanie výnimiek

In [1]:
```
# Príklad automatického vyvolania výnimky:

cislo = int(input("Zadajte číslo: " ))

try:
    vysledok = 2 / cislo   # môže nastať delenie 0
    print("Výseldok je:", vysledok)
except ZeroDivisionError:
    print("Chyba, nastalo delenie nulou!") # blok, ktorý sa vykoná pri chybe delenia
finally:
    print("Tento kód sa vykoná vždy.")     # blok, ktorý sa vykoná vždy
```

```
Zadajte číslo: 0
Chyba, nastalo delenie nulou!
Tento kód sa vykoná vždy.
```

Automatické aj programové vyvolávanie výnimiek

In [2]:
```
# Príklad automatického aj programového vyvolania výnimky:

cislo = int(input("Zadajte kladné číslo: " ))

try:
    if(cislo < 0):          # ak je číslo záporné, vyvolaj výnimku
        raise ValueError("Chyba, záporné číslo")

    vysledok = 2 / cislo    # môže nastať delenie 0
    print("Výseldok je:", vysledok)

except ValueError:                              # blok, ktorý sa vykoná, ak je číslo zá
    print("Chyba, číslo je záporné!")

except ZeroDivisionError:
    print("Chyba, nastalo delenie nulou!")    # blok, ktorý sa vykoná pri chybe delen
finally:
    print("Tento kód sa vykoná vždy.")        # blok, ktorý sa vykoná vždy
```

```
Zadajte kladné číslo: -1
Chyba, číslo je záporné!
Tento kód sa vykoná vždy.
```

Moduly

- sú to súbory, obsahujúce sadu preddefinovaných funkcií, procedúr a premenných,
- možno importovať **celý modul** alebo **iba jeho časť**,
- možno v aplikácii vytvárať vlasnté pomenovania modulov - **aliasy**.

Importovanie celého modulu

In [1]:
```
# Príklad importovania celého modulu:

import numpy # modul pre matematické funkcie

a = numpy.sqrt(100) # použitie funkcií sqrt a log10 z modulu numpy
b = numpy.log10(100)

print(a, b)
```

```
10.0 2.0
```

Importovanie celého modulu s aliasom

In [53]:
```
# Príklad importovania celého modulu s aliasom:

import numpy as np # modul pre matematické funkcie s aliasom np

a = np.sqrt(100) # použitie funkcií sqrt a log10 z modulu numpy
b = np.log10(100)

print(a, b)
```

```
10.0 2.0
```

Importovanie časti modulu

In [54]:
```
# Príklad importovania časti modulu:

from numpy import sqrt, log10 # z modulu numpy importuj iba funkcie sqrt a log10

a = sqrt(100) # použitie funkcií sqrt a log10 z modulu numpy
b = log10(100)

print(a, b)
```

```
10.0 2.0
```