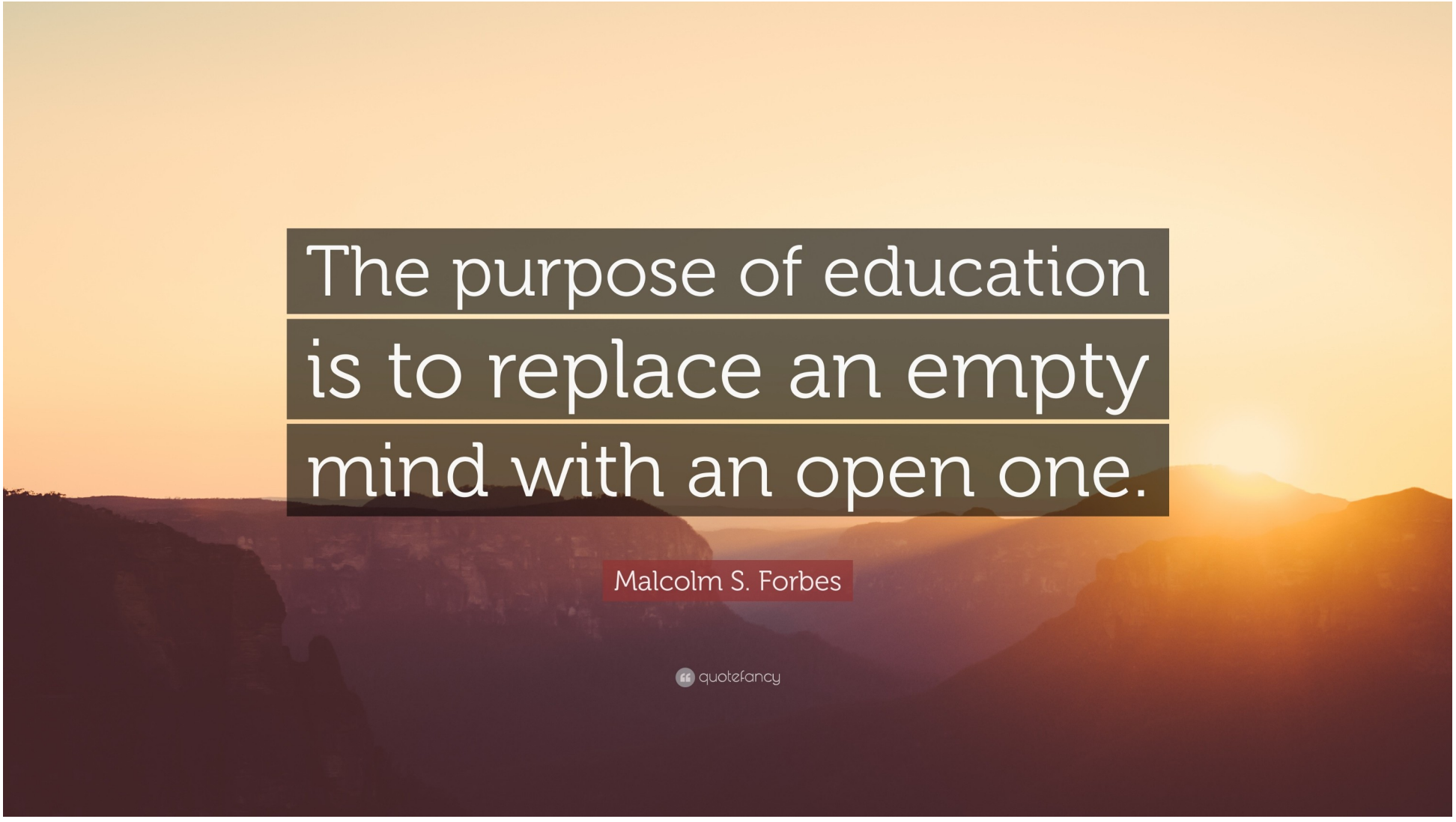
Programovanie 1

Prednášajúci:

Viliam Hromada, viliam.hromada[at]stuba.sk
C-510


The purpose of education is to replace an empty mind with an open one.
Malcolm S. Forbes
quotefancy


I ADDED TEXT TO
THIS SUNRISE
NOW IT IS INSPIRATIONAL
AS FUCK
quickmeme.com


a good place to put
inspirational quotes is
up your ass
ifunny.co

Najdôležitejšia informácia na dnešnej prednáške:

Programovať sa môžete naučiť iba tak, že **BUDETE VEĽA PROGRAMOVAŤ** !

Treba si to odsedieť za počítačom a **preriešiť veľa príkladov** !

30 kreditov = 40 hodín práce do týždňa

PROG1 = 6 kreditov = 8 hodín práce do týždňa (minimum)

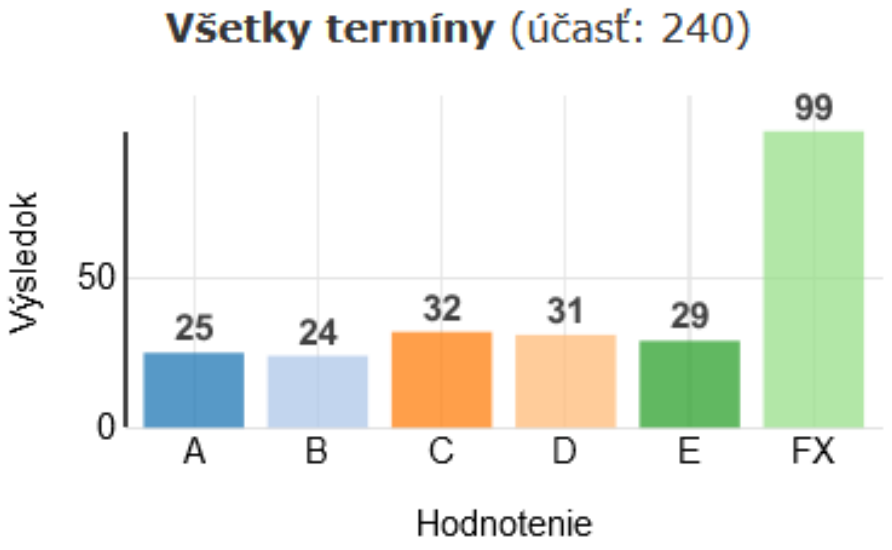
Druhá najdôležitejšia informácia na dnešnej prednáške:

Treba začať naplno **od prvého týždňa!**

Ak zaspíte na začiatku, po pár týždňoch budete stratení!

(Platí aj pre tých, ktorí si myslia, že preberanú látku už ovládajú zo strednej školy.)

Výsledky z 2023/2024



Organizačné pokyny

Ako bude prebiehať výučba

1) Prednáška – na prednáške odznie príslušné učivo. Pôjde o rôzne programátorské koncepty a ich použitie v jazyku Python.

2) Cvičenie – na cvičeniach budete samostatne riešiť úlohy, ktorých cieľom je precvičiť koncepty, ktoré odzneli na prednáške.

3) Samoštúdium – ak nestihnete vyriešiť všetky úlohy z cvičení, odporúčam venovať sa im vo voľnom čase a na ďalšom cvičení sa opýtať na to, čo ste nevedeli/nerozumeli.

Literatúra

Odporúčaná literatúra (podľa nej pôjdeme na prednáškach)

Allen B. Downey: **Think Python 2e**

Kniha je dostupná na:

http://greenteapress.com/thinkpython2/thinkpython2.pdf

Literatúra 2

Ďalšia odporúčaná literatúra:

Ben Stephenson:
The Python Workbook: A Brief Introduction with Exercises and Solutions

Obsahuje niekoľko programovacích úloh (často aj s riešeniami), ktoré sú podobné tým, ktoré budeme riešiť aj my.

Pekný youtube kanál o informatike

https://www.youtube.com/c/InformatikasMi%C5%A1om/featured

Obsahuje aj kurz Pythonu!

Cvičenia – prihlasovacie údaje

Pozrite si email v AISe!

Nájdete tam správu s vašimi **prihlasovacími údajmi do počítačových učební.**

Tieto údaje budete potrebovať na všetkých cvičeniach!

Webstránka predmetu

https://uim.fei.stuba.sk/predmet/b-prog1/

Budú na nej:

- Prezentácie z prednášok.
- Zadania cvičení.
- Zadania projektov.
- Pomocné materiály.

Podmienky absolvovania predmetu

1) 50 bodov za semester:
   - 2 testy, súhrnne za 40 bodov
   - 1 projekt za 10 bodov

2) 50 bodov skúška

Podmienky absolvovania:
1. aspoň 25 bodov za semester (podmienka pre účasť na skúške)
2. aspoň 25 bodov zo skúšky
3. dokopy aspoň 56 bodov

Vitajte na univerzite!

1) Ak bude počas semestra / skúšky zistené, že ste podvádzali, podľa študijného poriadku STU, čl. 13, bod 5, budete hodnotení známkou FX (t.j. „prepadli“ ste na predmete / neurobili ste ho).

2) Prednášky a cvičenia sú na FEI STU **povinné** zo všetkých predmetov.

3) Na cvičeniach sa bude robiť kontrola dochádzky („prezenčka“).

Ospravedlnenie z výuky

1) Ak ochoriete / nemôžete sa zúčastniť na výuke, je potrebné postupovať podľa Smernice dekana 4/2023:

„V prípade neúčasti na vzdelávacej činnosti (ďalej neúčasti) podľa čl. 5 ods. 5 Študijného poriadku STU v Bratislave je študent **povinný najneskôr do 5 pracovných dní** od jej **ukončenia** doručiť na **Pedagogické oddelenie fakulty doklad preukazujúci dôvod neúčasti.** Pri nesplnení tejto povinnosti sa neúčasť pokladá za neospravedlnenú.“

O čom bude tento predmet?

Budeme sa učiť ako písať počítačové programy.

Ak chceme, aby počítač niečo vykonal, musíme mu krok po kroku povedať, čo má robiť. (programovanie je tak trochu ako písanie receptov na varenie :)

Počítač má jednu dobrú vlastnosť:
Vždy urobí presne to, čo mu poviete.
Ale má aj jednu zlú vlastnosť:
Vždy urobí presne to, čo mu poviete.

O čom bude tento predmet?

Cieľom predmetu je **naučiť sa základné programátorské koncepty.**

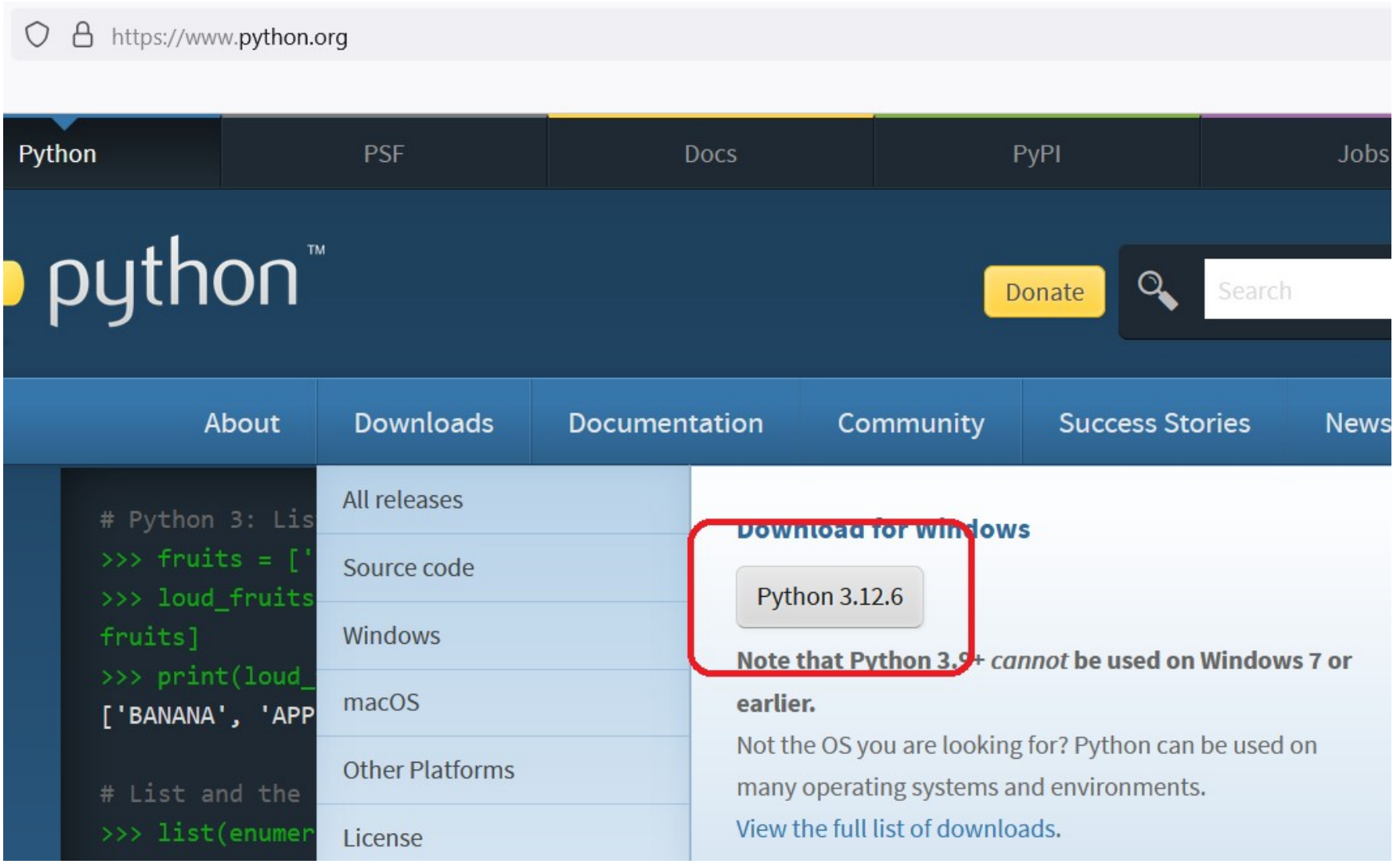
Pre ich aplikáciu sme vybrali jazyk Python.

Avšak cieľom predmetu **nie je** naučiť Vás Python.

Inštalácia Pythonu

Inštalujte si Python 3 z

https://www.python.org/

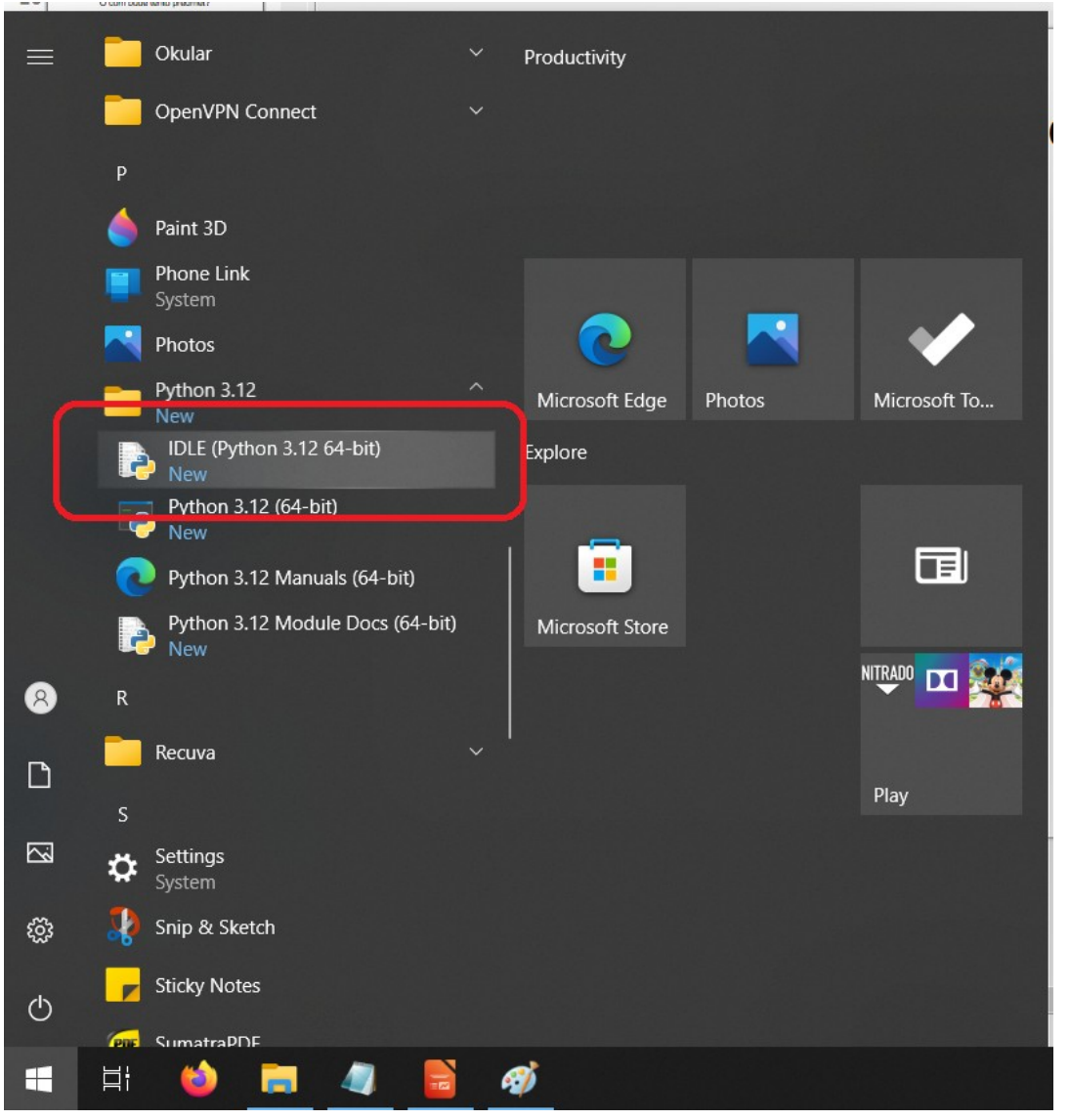

https://www.python.org
Python
PSF
Docs
PyPI
Jobs
python™
Donate
Search
About
Downloads
Documentation
Community
Success Stories
News
# Python 3: Lis
>>> fruits = ['
>>> loud_fruits
fruits]
>>> print(loud_
['BANANA', 'APP
# List and the
>>> list(enumer
All releases
Source code
Windows
macOS
Other Platforms
License
Download for Windows
Python 3.12.6
Note that Python 3.9+ cannot be used on Windows 7 or earlier.
Not the OS you are looking for? Python can be used on many operating systems and environments.
View the full list of downloads.

Inštalácia Pythonu

- Po inštalácii budete mať nainštalované:
  - Programovací jazyk Python 3
  - Jednoduché vývojové prostredie IDLE (pre naše účely absolútne postačujúce)

Spustenie Pythonu

Spustite aplikáciu **IDLE**

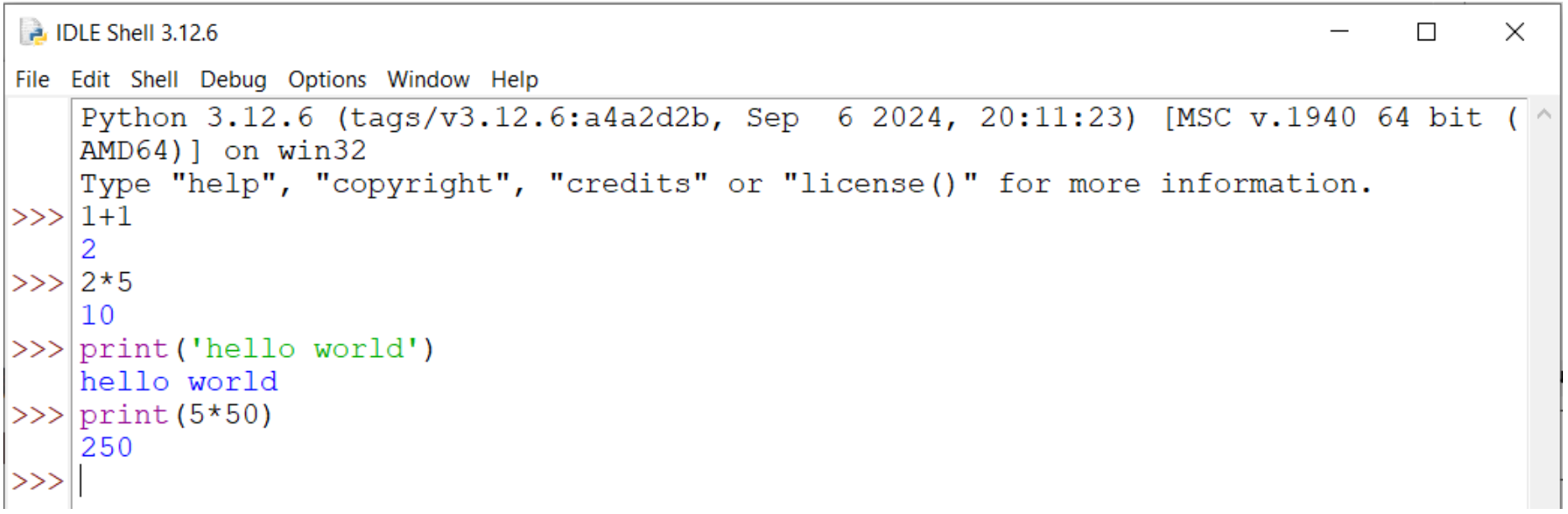


Python = interpretovaný jazyk

- Programovacie jazyky delíme do 2 hlavných skupín:
  - Prekladané jazyky (kompilované)
    - Program sa najprv musí skompilovať do tzv. strojového kódu, t.j. inštrukciám, ktorým rozumie procesor
  - Interpretované jazyky
    - Program sa priamo vykonáva, príkaz za príkazom
    - Sem patrí napr. aj Python

Python – interaktívny mód (interpreter / shell)

- V interaktívnom móde vieme pomocou Pythonu počítaču zadávať inštrukciu za inštrukciou a počítač ich bude rovno vykonávať (t.j. interpretovať)

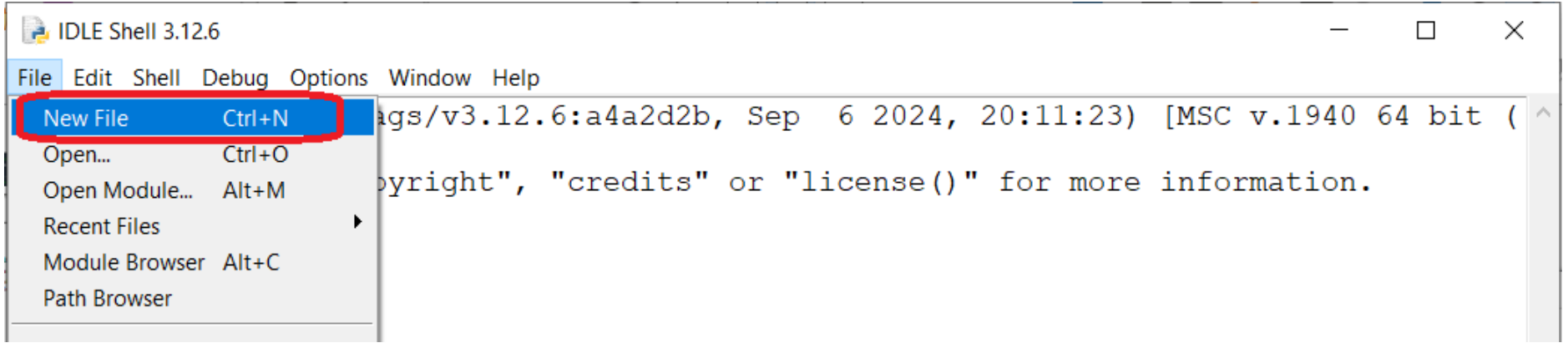


Python – skriptovací mód

- V skriptovacom móde najprv napíšeme celý program v jazyku Python.
- Program je tvorený postupnosťou príkazov (inštrukcií). Takýto zápis inštrukcií sa niekedy nazýva **zdrojový kód (**prípadne „slangovo“ len **kód).**
- A následne celý program spustíme a inštrukcie sa budú postupne vykonávať.

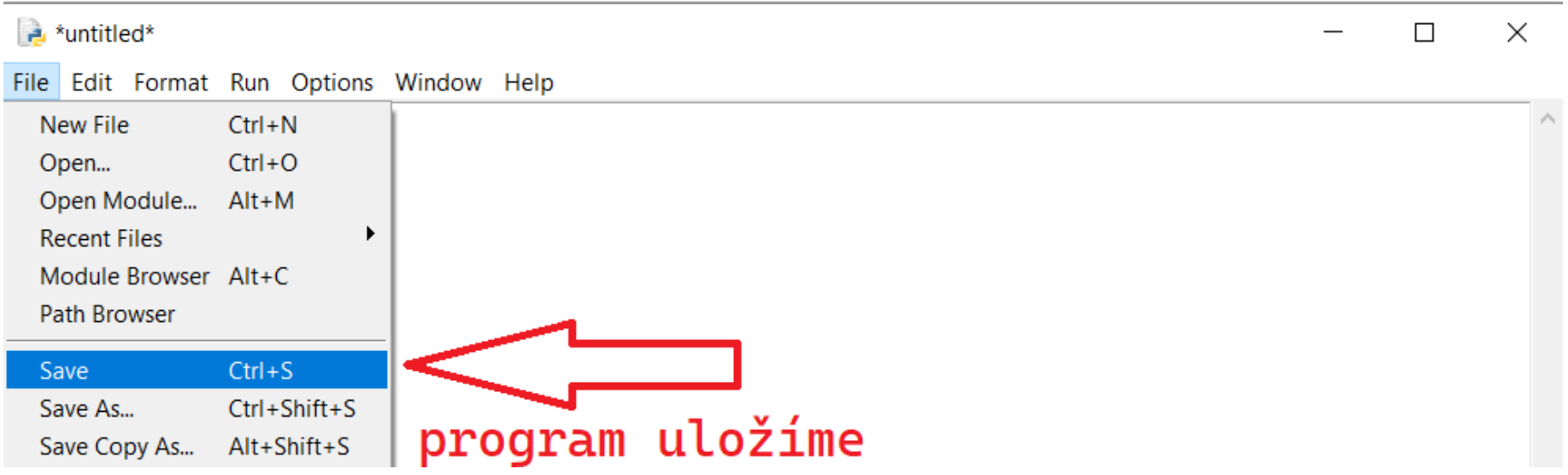
Skriptovací mód

- Na použitie skriptovacieho módu najprv vytvoríme nový súbor, kam napíšeme všetky príkazy / inštrukcie, ktoré chceme, aby počítač pomocou jazyku Python vykonal.

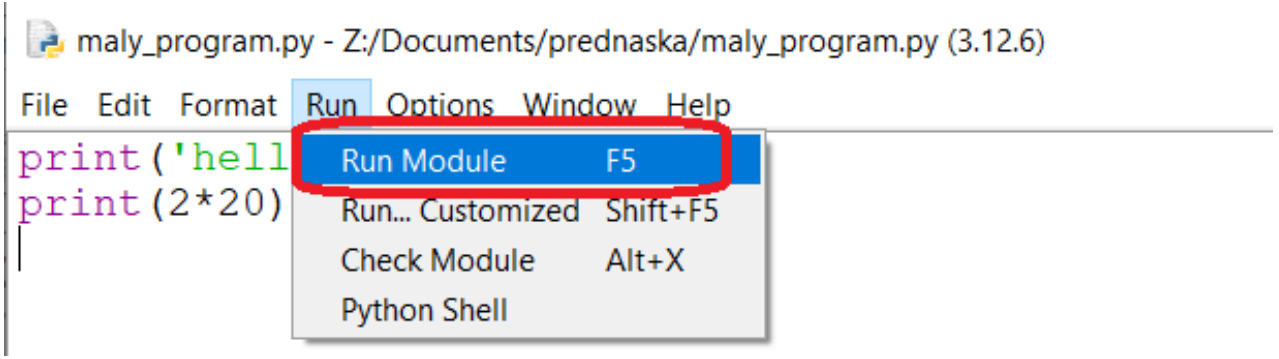


Najprv vytvoríme program (Skript), t.j. napíšeme požadované príkazy:

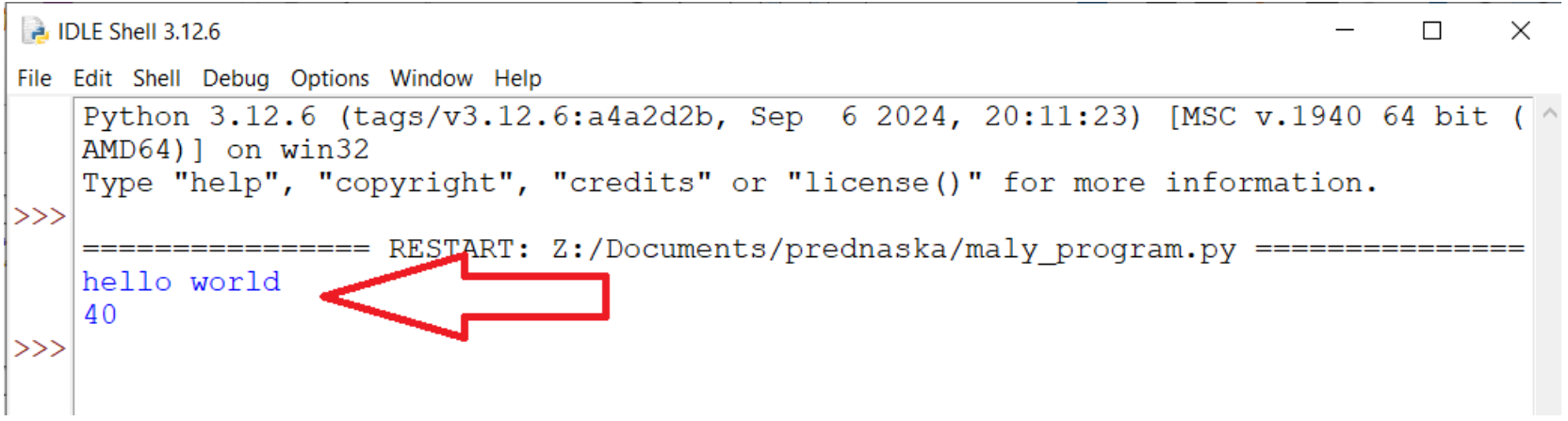
Potom program (skript) uložíme:



Takto program (skript) spustíme:



V interpretéri (shelli), vidíme výstup programu:



Program (skript / zdrojový kód)

- Program (skript) je postupnosť inštrukcií (príkazov), ktorá špecifikuje, ako sa má vykonať nejaký výpočet. Takto zapísanú postupnosť príkazov nazývame aj **zdrojový kód.**
- Výpočet môže byť matematický (napr riešenie sústavy rovníc).
- Výpočet môže byť implementácia nejakého algoritmu (hľadanie reťazca v texte, šifrovanie dát, triedenie hodnôt od minima po maximum, generovanie textu pomocou umelej inteligencie, …)
- Výpočet je v podstate vykonanie ľubovoľnej postupnosti príkazov.

Program (skript)

- Každý programovací jazyk obsahuje nejakú množinu inštrukcií, ktoré je možné použiť na programovanie.
- Každá takáto inštrukcia niečo prikazuje počítaču – čo má robiť.
- Rôzne programovacie jazyky (Python, C, Java, …) majú rôzne inštrukcie (príkazy), avšak vo všeobecnosti každý programovací jazyk obsahuje nasledovné typy príkazov (inštrukcií)

Príkazy (inštrukcie)

- Vstup – príkazy, pomocou ktorých je možné načítať nejaké dáta do programu od používateľa (napr. z klávesnice, z disku, atď.)
- Výstup – príkazy, pomocou ktorých je možné zobraziť dáta z programu (napr. na obrazovku, vytlačiť, uložiť na disk, poslať po sieti, atď.)
- Aritmetické inštrukcie (s dátami v programe je možné vykonávať matematické operácie súčet, súčin, rozdiel, podiel, atď.)
- Volania iných funkcií / podprogramov
- Podmienky (vetvenie) – výpočet môže pokračovať rôznymi cestami podľa toho, či platí nejaká podmienka
- Opakovanie (iterácia) – časti programu sa môžu opakovane vykonávať.

Výstup - print

- Základným príkazom, pomocou ktorého vieme zobraziť dáta na obrazovku, je v jazyku Python príkaz (lepšie povedané funkcia) **print**
- Použitie (do zátvoriek za **print** uvedieme, čo chceme vypísať):

  print(*tu sa napíše to, čo chceme vypísať na obrazovku*)

Všimnite si rôzne použitia print() a ako sa hodnota, ktorá je uvedená v zátvorke, vypisuje na obrazovku:
- ak chceme vypísať nejaký konkrétny text, uvedieme ho medzi apostrofy alebo úvodzovky
- ak chceme vypísať nejakú číselnú hodnotu, priamo ju uvedieme
- Python dokonca sám vypočíta hodnotu výrazu, napr. 1+5*6 ako 31 a vypíše rovno 31

```
>>> print("Hello world")
Hello world
>>> print('Hello world')
Hello world
>>> print(5)
5
>>> print(5+4)
9
>>> print(5*4)
20
>>> print(1+5*6)
31
>>> print('a')
a
```

Aritmetické operátory

- V matematických výrazoch v jazyku Python vieme používať štandardné matematické operátory, napríklad:

  + pre súčet, - pre rozdiel

  * pre súčin, / pre podiel

  ** pre umocnenie

  % pre zvyšok po delení

  // pre celočíselnú časť podielu

  () zátvorky pre ozátvorkovanie výrazu

  a iné :)

Aritmetické operátory

- Názvoslovie:
- Symbol, ktorý predstavuje danú operáciu, nazývame **operátor**
- Hodnoty, s ktorými sa operácia vykonáva, nazývame **operandy**
- Napríklad v zápise: 1 + 2
  - + je operátor (sčítania)
  - Hodnoty 1 a 2 sú operandy

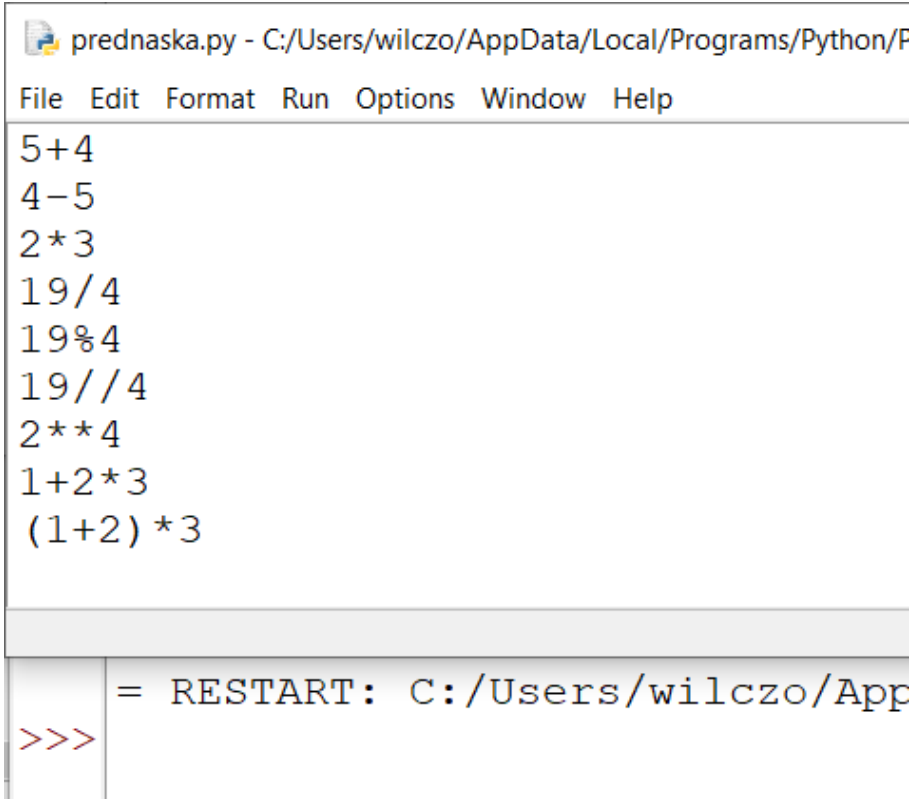
Aritmetické operátory

V interaktívnom móde vieme matematické výrazy priamo zadávať a Python vypočíta ich hodnoty (funguje ako kalkulačka)

```
>>> 5+4
9
>>> 4-5
-1
>>> 2*3
6
>>> 19/4
4.75
>>> 19%4
3
>>> 19//4
4
>>> 2**4
16
>>> 1+2*3
7
>>> (1+2)*3
9
```

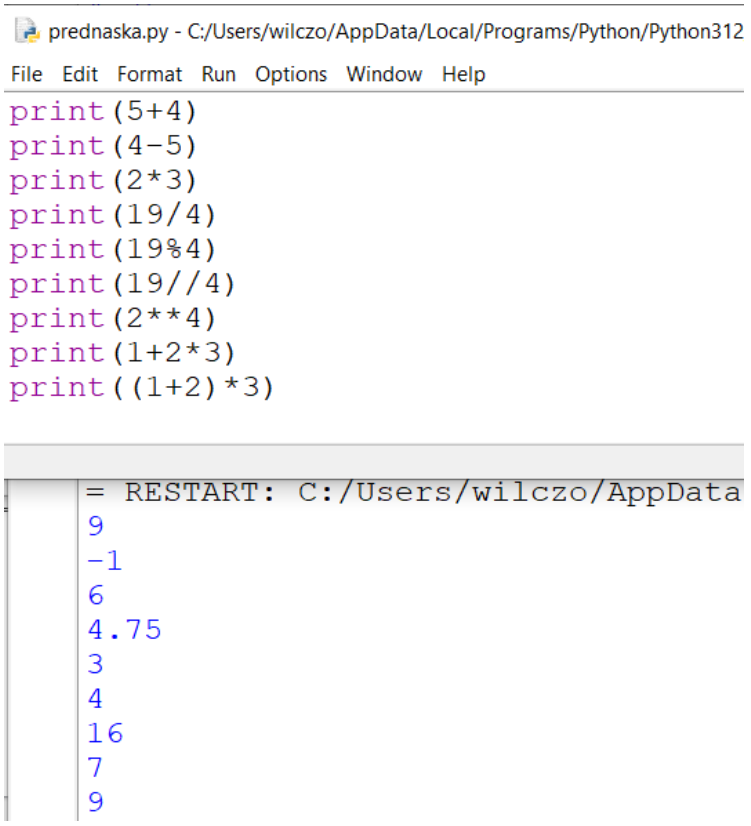
Aritmetické operátory

V skriptovacom móde by predošlá postupnosť výrazov navonok neurobila nič...



Aritmetické operátory

Ak chceme v skriptovacom móde vypisovať hodnoty matematických výrazov, musíme ich vložiť do funkcie **print()**



Hodnoty a ich typy (values / types)

- V uvedených príkladoch sme pri použití (volaní) funkcie **print()** do zátvoriek napísali **hodnotu**, aká sa má vypísať na obrazovku.

- **Hodnota** je vo všeobecnosti nejaký údaj, s ktorým môže program pracovať. V našich príkladoch spočívala „práca“ s týmito hodnotami v tom, že sa vypísali na obrazovku pomocou **print().**

- V uvedených príkladoch sme mali 3 **typy hodnôt**:
  - a) celé číslo (v Pythone **int** od slova *integer*)
  - b) číslo s pohyblivou čiarkou (v podstate desatinné číslo, v Pythone **float** od slova *floating-point*)
  - c) reťazce (postupnosť znakov**,** v Pythone **str** od slova *string*).

- Funkcia **type()** v jazyku Python povie, akého typu je nejaká hodnota

Hodnoty a typy

```
>>> type(10)
<class 'int'>
>>> type(8/3)
<class 'float'>
>>> type(10.5)
<class 'float'>
>>> type(10.0)
<class 'float'>
>>> type('hello world')
<class 'str'>
```

Hodnoty a typy

- Hodnota typu **int** (celé číslo) je v podstate ľubovoľné celé číslo, napr. 10
- Hodnota typu **float** je desatinné číslo, t.j. musí obsahovať desatinnú čiarku (lepšie povedané „bodku“), napr. 10.5 alebo 10.0
- Hodnota typu **string** (reťazec) je ľubovoľná hodnota, ktorá je uvedená medzi apostrofmi (‘) alebo úvodzovkami (“).

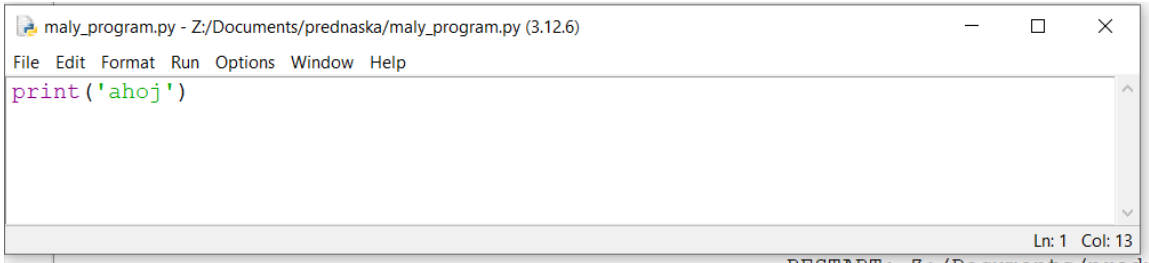
Úskalie programovacích jazykov

- Programovacie jazyky **vyžadujú**, aby v nich písané programy dodržovali všetky stanovené pravidlá (tzv. syntax).
- Ak to nedodržíte, program sa nebude dať korektne spustiť.
- Našťastie, Python vás o tom vždy informuje, čiže:

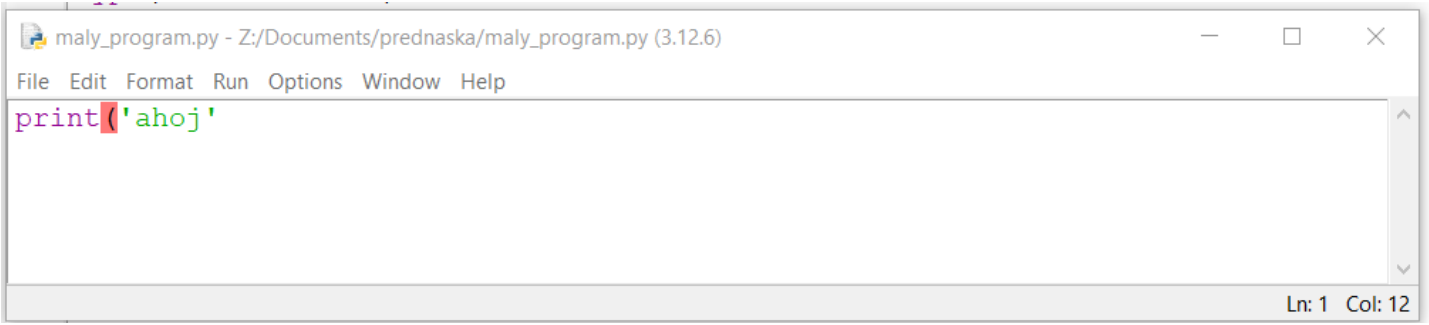
Čítajte chybové výpisy!!!

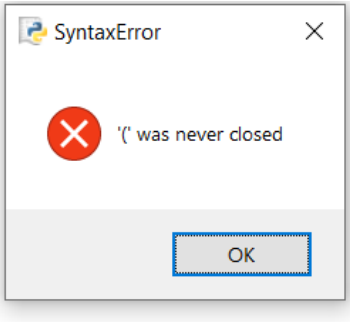
Zátvorky, zátvorky, zátvorky

- Napríklad, ak používam funkciu **print(),** musí byť korektne ozátvorkovaná!



Vyššie uvedené korektné použitie print(), nižšie chýba pravá zátvorka a Python nás upozornil!





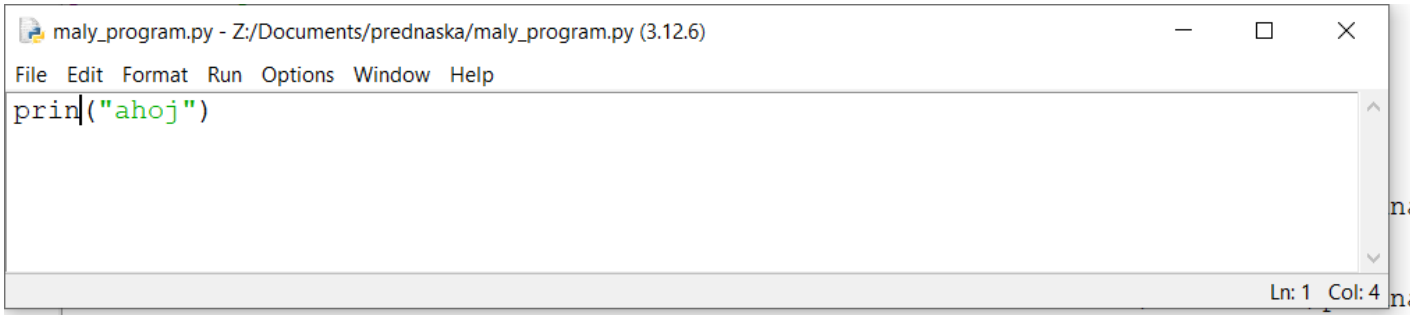
Iné syntaktické chyby

- Ak použijem binárnu operáciu (takú, čo pracuje s 2 hodnotami) a zabudnem dodať 1 hodnotu:

- Alebo ak zabudnem na ukončovacie úvodzovky pri reťazci:

Iné syntaktické chyby

- Rovnako, ak urobím preklep v názve príkazu/funkcie, tak ma Python upozorní, že taký príkaz/funkciu nepozná!



(samozrejme, existuje teoretická šanca, že funkcia s názvom „prin“ by existovala, v takom prípade by Python použil funkciu „prin“ a nedovtípil by sa, že ste mysleli „print“ :))

Reklamná prestávka

- To, ako počítač vôbec dokáže:
  - spracovať nejaký program zapísaný napr. v Pythone,
  - ako ho vie postupne vykonávať,
  - ako kontroluje, že dodržiava syntax (pravidlá zápisu príkazov),
  - ako vie, že sú tam zle ozátvorkované výrazy,
  - ako vie, že tam chýbajú hodnoty,
  - a iné veci súvisiace so spracovaním programu (skriptu),

- To všetko sa učí na inžinierskom štúdiu na predmete Automaty a formálne jazyky, ktorý prednáša Ing. Viliam Hromada, PhD.

- Takže o tom viac snáď o 3-4 roky :)

Premenné (variables)

- Počítač samozrejme dokáže realizovať rôzne výpočty na dátach, t.j. pracovať s rôznymi hodnotami a meniť ich.
- Pre jednoduchšiu prácu s dátami v pamäti počítača používame **premenné.**
- Premenná je v podstate miesto v pamäti počítača, ktoré má svoje meno a nejakú hodnotu.
- Keď chceme s touto hodnotou pracovať, tak k nej pristúpime pomocou mena príslušnej premennej.

Priradenie hodnoty premennej (assignment)

- Ak ste už programovali, možno ste zvyknutí z iných jazykov (Java, C), že premennú pred použitím treba deklarovať.
- V Pythone nie je potrebné premennú deklarovať, premenná sa automaticky vytvorí napríklad pri priradení hodnoty.
- Ak chceme vytvoriť premennú s názvom *meno_premennej* a vložiť do nej nejakú hodnotu, urobíme to pomocou operátora priradenia „=“:

meno_premennej = hodnota

Priradenie hodnoty premennej (assignment)

- Príklady vytvorenia premennej a jej priradenia hodnoty:

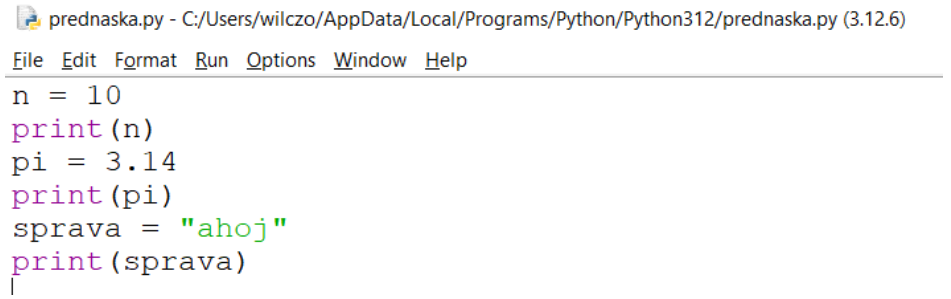
  N = 10 (vytvorí premennú s názvom „N“ a celočíselnou hodnotou 10)

  pi = 3.14 (vytvorí premennú s názvom „pi“ a desatinnou hodnotou 3.14)

  sprava = “ahoj“ (vytvorí premennú s názvom „sprava“ a vloží do nej reťazec „ahoj“)

Výpis hodnoty premennej na obrazovku

- Ak chceme vypísať, aká hodnota je uložená v premennej, urobíme to napríklad pomocou funkcie **print()** kam „do zátvoriek“ vložíme meno príslušnej premennej
  Program:

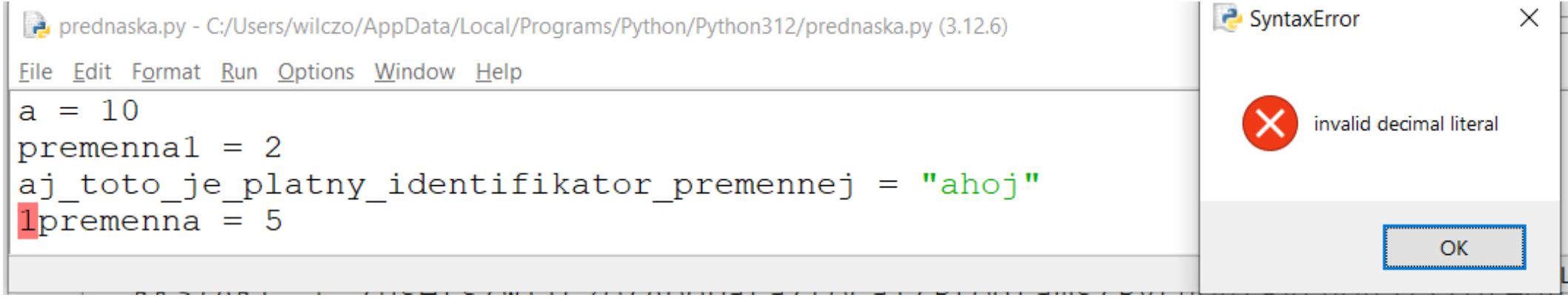


Výpis:

```
10
3.14
ahoj
```

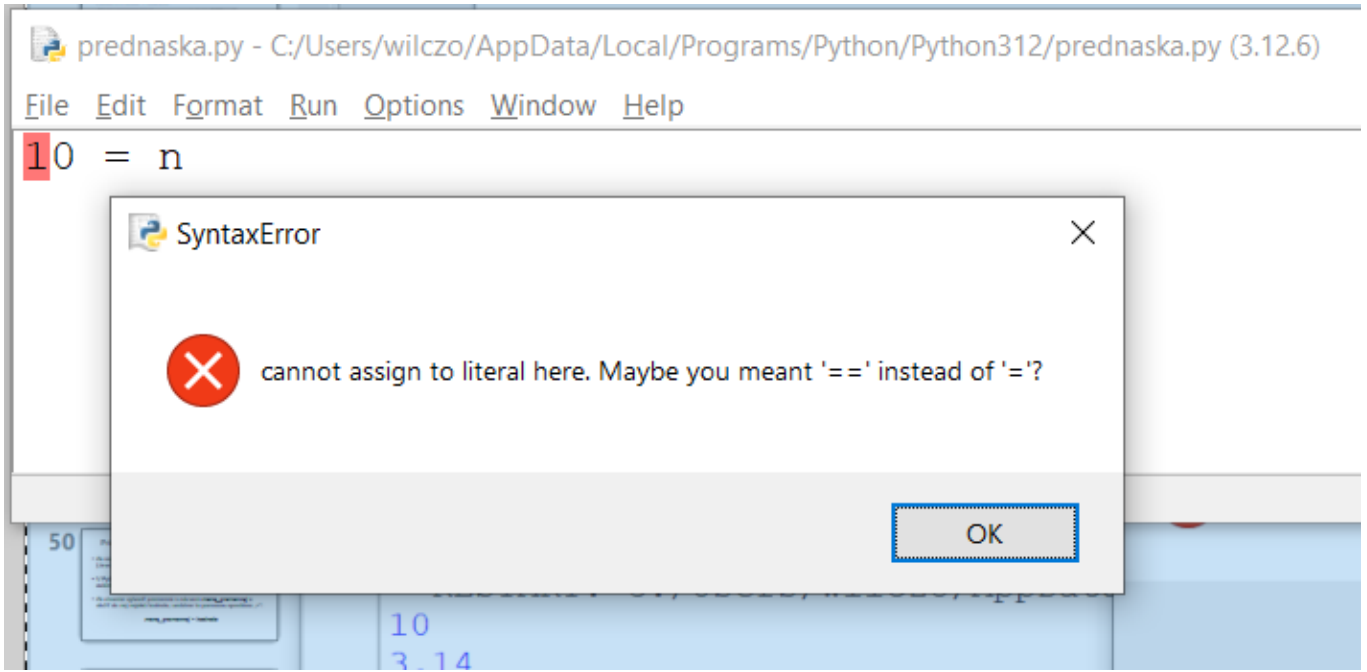
Meno premennej

- Meno (identifikátor) premennej môže v jazyku Python obsahovať veľké/malé písmená anglickej abecedy, číslice alebo podtržník (_)
- Avšak NESMIE začínať číslicou!!!
- V uvedenom príklade sú prvé 3 platné mená premenných, pri 4. Python vypíše chybu!



Vytvorenie premennej

- POZOR!!! Pri vytváraní premennej musí byť meno premennej na ľavej strane a priraďovaná hodnota na pravej strane!
  Ak by som napríklad chcel do premennej „n“ vložiť číslo 10, ale spravím to naopak: 10 = n, tak Python ma upozorní na chybu!



Premenné

- Do premennej môžem vložiť v princípe ľubovoľnú hodnotu (celé číslo, desatinné číslo, reťazec,…), ale aj **hodnotu z inej premennej. Napríklad**, ak máme v premennej „a“ nejakú hodnotu (napr. číslo 10), ktorú chceme uložiť do premennej „b“, urobíme to ako:

b = a

Program:

```
a = 10
b = a
print(b)
```

- Výpis:

```
10
```

Premenné

- Avšak pozor! Ak sa pokúsim vypísať hodnotu premennej, do ktorej som ešte žiadnu hodnotu nevložil – to jest Python danú premennú ešte nevytvoril, teda de facto neexistuje, nastane chyba!
  Program:

```
a = 10
print(b)
```

- Vytvorí premennú „**a**“ do ktorej sa vloží 10, ale **print(b)** znamená výpis premennej „**b**“, ktorá ešte **nebola vytvorená!** Preto Python vypíše chybu, že nepozná premennu „**b**“, teda že meno „b“ ešte nebolo definované!

```
  print(b)
NameError: name 'b' is not defined
```

Premenné

- Ak už v premennej nejaká hodnota existuje, tak po priradení novej hodnoty sa obsah premennej prepíše.

- Kód:

```
a = 10
print(a)
a = 12
print(a)
```

Po spustení dôjde k výpisom: (všimnite si, že premenná „a“ najprv obsahovala 10 a po prepísaní 12):

```
10
12
```

Premenné

- V doterajších príkladoch sme do premenných vkladali nejaké konkrétne konštantné hodnoty (čísla, reťazce)

- Do premenných však môžeme vkladať hodnoty aj také, ktoré predstavujú nejaký zložitejší výraz, ktorý sa najprv vyhodnotí a do premennej sa vloží až výsledná hodnota!

Premenné

- Napríklad program:

prednaska.py - C:/Users/wilczo/AppData/Local/Programs/Python/Python312/prednaska.py (3.12.6)

File Edit Format Run Options Window Help

```
strana_a = 10
strana_b = 20
obvod_obdlznika = 2*(strana_a+strana_b)
obsah_obdlznika = strana_a*strana_b
print(obvod_obdlznika)
print(obsah_obdlznika)
```

- Vypíše ako hodnoty v premenných „obvod_obdlznika“ a „obsah_obdlznika“ hodnoty:

```
60
200
```

Výraz (expression)

- Vo všeobecnosti, akákoľvek kombinácia:
  - Hodnôt
  - Premenných
  - Operátorov

  sa v jazyku Python nazýva **výraz** (expression).

- **Výraz** je vlastne matematickým výrazom, ktorý nadobúda nejakú hodnotu.

- Na predošlom príklade bol
  2(strana_a + strana_b)*
  výraz, ktorý nadobudol hodnotu 60 ak v premennej strana_a bola hodnota 10 a v premennej strana_b bola hodnota 20.

Výraz (expression)

- Teda do premenných v jazyku Python môžeme okrem konštantných hodnôt (konkrétne hodnoty čísiel, reťazcov, atď.) priradovať aj výrazy.

- Pri priradení výrazu do premennej sa najprv vyhodnotí hodnota celého výrazu a až tá sa vloží do premennej.

- V podstate to celé funguje presne tak, ako ste zvyknutí z matematiky :)

Elementárne matematické funkcie

- V jazyku Python môžeme využívať okrem matematických operácií aj základné matematické funkcie (sin(), cos(), sqrt(), …).

- Tieto funkcie sú už **predprogramovanou** súčasťou Pythonu a môžeme ich taktiež používať ako súčasť výrazov a ich hodnoty priraďovať do premenných.

- Tieto matematické funkcie sú súčasťou tzv. **modulu** *math*. Modul je meno pre nejakú zbierku predprogramovaných funkcií (t.j. niečo, čo už niekto iný naprogramoval). Ak chcem takýto modul použiť, musím ho najprv ***importovať*** do môjho programu.

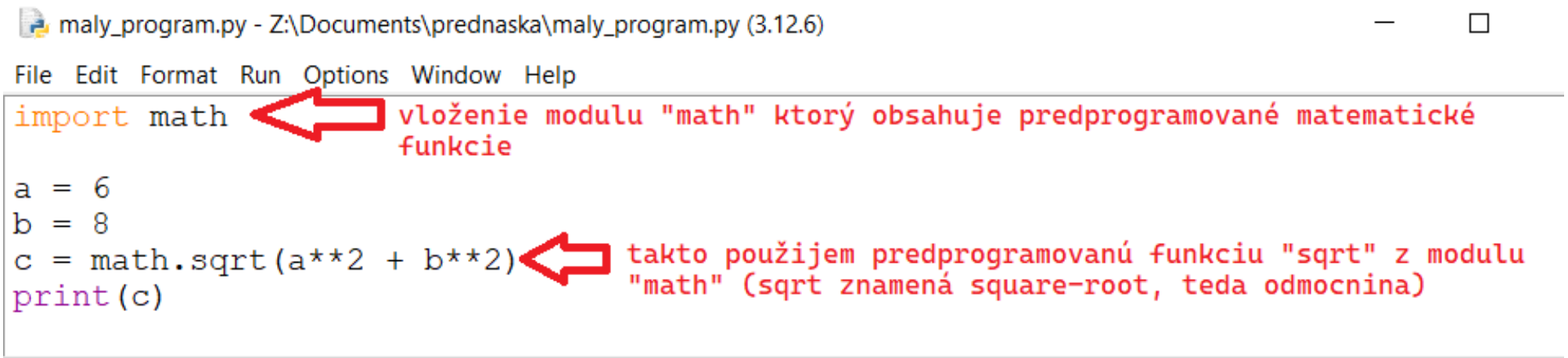
- Následne budem mať tieto predprogramované časti k dispozícii.

Elementárne matematické funkcie a import

- Príklad programu, ktorý používa predprogramovanú funkciu „sqrt“ na výpočet odmocniny. Táto funkcia sa nachádza v predprogramovanom module *math,* ktorý musím najprv „naimportovať“ do môjho programu.

- Následne, ak chcem použiť funkciu „sqrt“, tak ju použitem spôsobom: *math.sqrt*(*hodnota z ktorej chcem odmocninu)*

- Všimnite si, že akoby celý výraz *math.sqrt(a**2+b**2)* nadobudne hodnotu $\sqrt{a^2 + b^2}$



Funkcie

- Viac si k funkciám, ako základnej súčasti programovania, povieme na ďalšej prednáške.

Príkaz (statement)

- Príkaz (statement) je základná jednotka kódu, ktorú je schopný počítač v danom programovacom jazyku vykonať / spracovať.
- Ekvivalentom by mohol byť pojem „inštrukcia“.
- V jazyku Python píšeme spravidla každý príkaz na nový riadok, pre lepšiu prehľadnosť.
- Príkazmi sú napríklad:
  - Priradenie hodnoty do premennej
  - Volanie rôznych funkcií (napr. print())
  - Vetvenia, Cykly, atď.

Príkaz (statement)

- Napríklad program:

prednaska.py - C:/Users/wilczo/AppData/Local/Programs/Python/Python312/prednaska.py (3.12.6)

File Edit Format Run Options Window Help

```
strana_a = 10
strana_b = 20
obvod_obdlznika = 2*(strana_a+strana_b)
obsah_obdlznika = strana_a*strana_b
print(obvod_obdlznika)
print(obsah_obdlznika)
```

- Obsahuje 6 príkazov („statementov“):
  - 2 priradenia konkrétnych hodnôt do premenných (riadky č. 1,2)
  - 2 priradenia hodnôt výrazov do premenných (riadky č. 3,4)
  - 2 volania funkcií print() (riadky č. 5,6)

Priorita (precedencia) operátorov

- Python pri vyhodnocovaní výrazov používa prioritu operátorov tak, ako sme zvyknutí z matematiky, podľa tzv. PEMDAS princípu, v nasledovnom poradí:

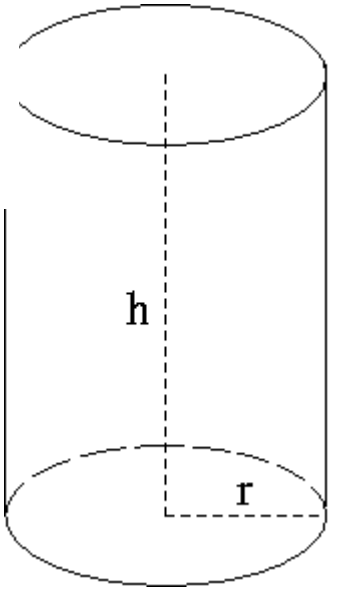
  - 1) Parentheses (zátvorky)
  - 2) Exponentiation (umocňovanie)
  - 3) Multiplication-Division (násobenie-delenie)
  - 4) Addision-Subtraction (sčítanie-odčítanie)

Priorita (precedencia) operátorov

- Teda napríklad v programe na výpočet obsahu/objemu valca s výškou h = 3 a polomerom r = 2 kde podľa matematiky:

objem: $\pi r^2 h$

povrch/obsah: $2\pi r(r+h)$




```
pi = 3.14
h = 3
r = 2
objem_valca = pi*r**2*h
povrch_valca = 2*pi*r*(r+h)
print(objem_valca)
print(povrch_valca)
```

Inkrementácia/dekrementácia premennej

- Ako sme si povedali, hodnotu v premennej môžeme meniť (t.j. hodnota v premennej sa môže zmeniť – preto sa premenná volá premenná :))

- Typickou situáciou v programovaní je tzv. *inkrementácia premennej*, t.j. zvýšenie jej hodnoty o 1, resp. *dekrementácia premennej*, t.j. zníženie jej hodnoty o 1.

- V Pythone sa to vykoná nasledovným spôsobom. Ak je meno premennej povedzme „i“, potom:

Zvýšenie o 1:

i = i + 1
alebo
i += 1

Zníženie o 1:

i = i - 1
alebo
i -= 1

Inkrementácia premennej

- Ak by sme chceli zvýšiť hodnotu premennej, povedzme „i“ o 1:

```
i = 10
print(i)
i = i + 1
print(i)
```

spôsobí výpis:

```
10
11
```

Alternatívne je to možné urobiť aj takto:

```
i = 10
print(i)
i += 1
print(i)
```

tiež vypíše:

```
10
11
```

Dekrementácia premennej

- Analogicky, zníženie premennej o 1 sa vykoná:

```
i = 10
print(i)
i = i - 1
print(i)
```

vypíše:

```
10
9
```

alebo:

```
i = 10
print(i)
i -= 1
print(i)
```

tiež vypíše:

```
10
9
```

Premenné: príklad

```
N=10
N=5
N=N+1
print(N)
```

Tento skript vypíše: 6

V príkaze N=N+1 sa najskôr ohodnotí pravá strana a tá sa potom priradí do N.

Pri priradení sa najskôr ohodnotí pravá strana a jej hodnota sa potom priradí ľavej strane!

Operátory pre prácu s reťazcami

- Teda v premenných môžu byť čísla a pre aritmetické operácie s nimi máme rôzne operátory, ktoré poznáte z matematiky.

- Avšak v Pythone môžu byť ako hodnoty v premenných aj reťazce, t.j. nejaké postupnosti znakov, napríklad program:

```
text1 = "Ahojte"
text2 = "studentky a studenti!"
print(text1)
print(text2)
```

- vypíše:

```
Ahojte
studentky a studenti!
```

Zreťazenie reťazcov (operátor +)

- 2 reťazce môžeme spojiť do 1 reťazca pomocou operátora zreťazenia, ktorý má v Pythone označenie +
  teda rovnako, ako bežný súčet.

  POZOR!!! Zreťazenie **nie je komutatívne! Záleží teda na poradí,** v akom reťazce zreťazím!

Zreťazenie reťazcov (operátor +)

- Všimnite si, čo urobí program:

```
text1 = "Ahojte"
text2 = "studentky a studenti!"
text3 = text1+text2
text4 = text2+text1
print(text3)
print(text4)
```

- Vypíše:

```
Ahojtestudentky a studenti!
studentky a studenti!Ahojte
```

Zreťazenie reťazcov: operátor +

```
ret1=“student sa musi“
ret2=“ vela ucit“
ret3= ret1 + ret2
print(ret3)
```

Vypíše:

student sa musi vela ucit

Opakovanie reťazcov (operátor *)

- Vytvorenie nového reťazca, ktorý predstavuje opakovanie nejakého iného reťazca daný počet krát dosiahneme pomocou operátora * (t.j. akoby klasické násobenie).

- V tomto operátore musí byť jeden operand reťazec (t.j. reťazec, ktorý cheme zopakovať) a druhý operand nejaké celé číslo (počet opakovaní)!

*reťazec * počet_opakovaní*

- Ak by sme teda ako operandy pri * uviedli 2 reťazce, nastane chyba!

Opakovanie reťazcov (operátor *)

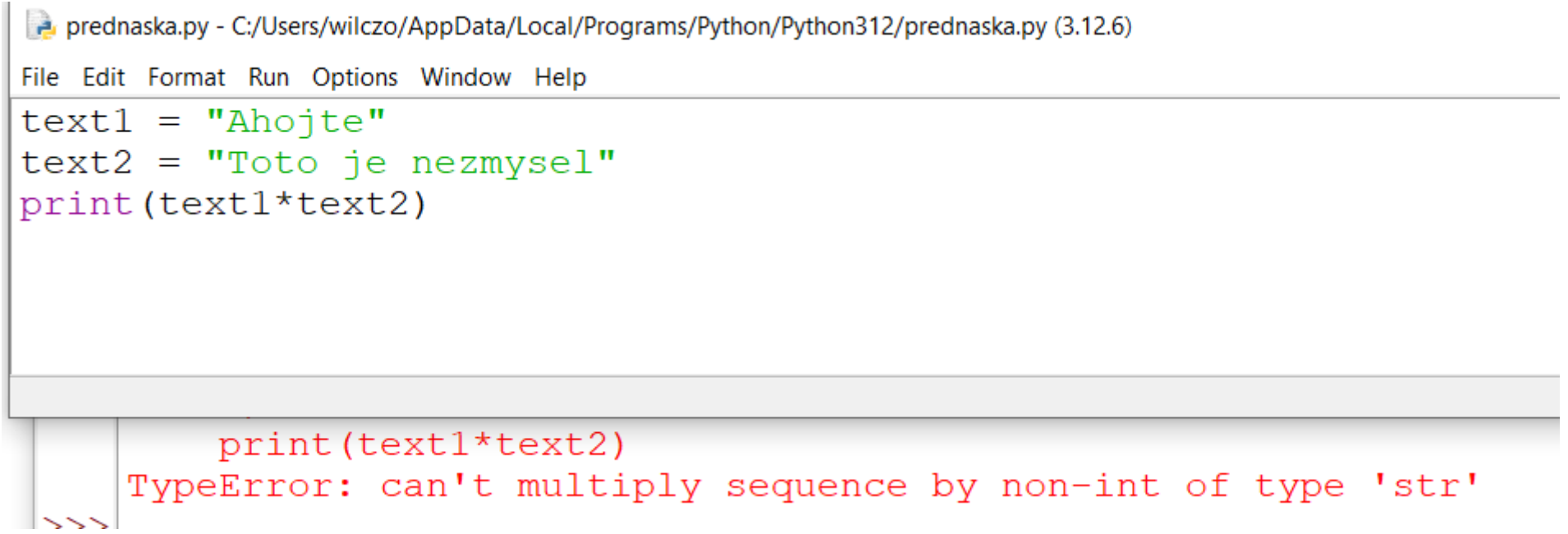
- Korektný program spolu s výpisom:

```
text1 = "Ahojte"
opakovani = 5
print(2*text1)
print(text1*opakovani)
```

```
AhojteAhojte
AhojteAhojteAhojteAhojteAhojte
```

Opakovanie reťazcov (operátor *)

- Chyba, ak ako oba operandy * uvediem nejaké reťazce
- (samozrejme, že je to chyba, lebo Python pri * očakáva, že uviediete, koľkokrát sa má daný reťazec zopakovať!)



Opakovanie reťazcov: operátor *

```
ret1=“Programovat! “
ret2=ret1*3
print(ret2)
```

Vypíše:

Programovat! Programovat! Programovat!

Komentáre

- Pri programovaní je často užitočné vkladať si do kódu tzv. komentáre

- Komentár je slovný popis toho, čo vykonáva nejaká časť kódu.

- Python pri spracovaní kódu vie, že komentár je len pre informáciu, a teda ho pri vykonávaní programu ignoruje!
- Aby toto však Python vedel, je nutné komentár označiť!

- Komentáre slúžia **len pre programátorov**, aby lepšie popisovali, čo ktorá časť kódu robí!

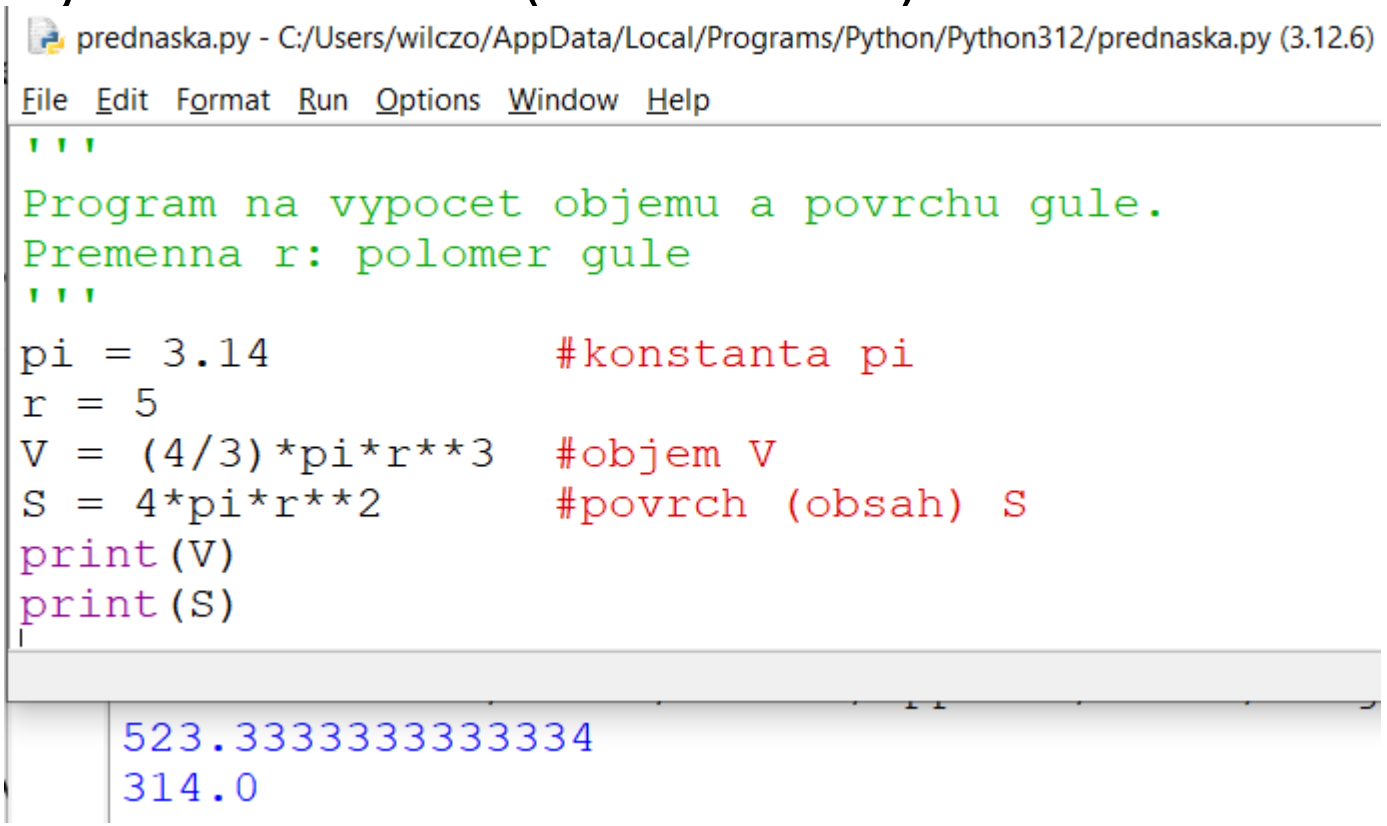
Komentáre

- Komentár na 1 riadku kódu je označený symbolom # na začiatku komentára

- Python umožňuje aj viacriadkové komentáre – tie sa označujú tak, že na začiatok a koniec viacriadkového komentára sa dajú 3 apostrofy

Komentáre

- Ukážka programu s jednoriadkovými (červená farba) a viacriadkovými komentármi (zelená farba).



Záverečné zhrnutie

- Ukázali sme si, ako sa používa prostredie IDLE pre prácu s Pythonom v interaktívnom a skriptovacom móde.
- Ukázali sme si, ako sa dá používať výstup z programu v Pythone cez funkciu print()
- Povedali sme si, čo je premenná, uviedli sme príklad na aritmetické / reťazcové operácie v Pythone.
- Ukázali sme si, čo sú komentáre a ako sa používajú.

Záverečné zhrnutie

- Táto prednáška pokrýva prvé 2 kapitoly z knihy Think Python.
- **Dôrazne odporúčam** v rámci domáceho štúdia si prečítať tieto prvé 2 kapitoly a na cvičeniach sa opýtať na veci, ktorým ste nerozumeli a ktoré Vám nie sú jasné.