

Základy jazyka

Jiří Zacpal



KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZPP1 Základy programování v Pythonu 1



Doporučená literatura

1. **Mark Lutz: *Learning Python*.**
2. Libovolné další učebnice jazyka Python.

Požadavky na zápočet



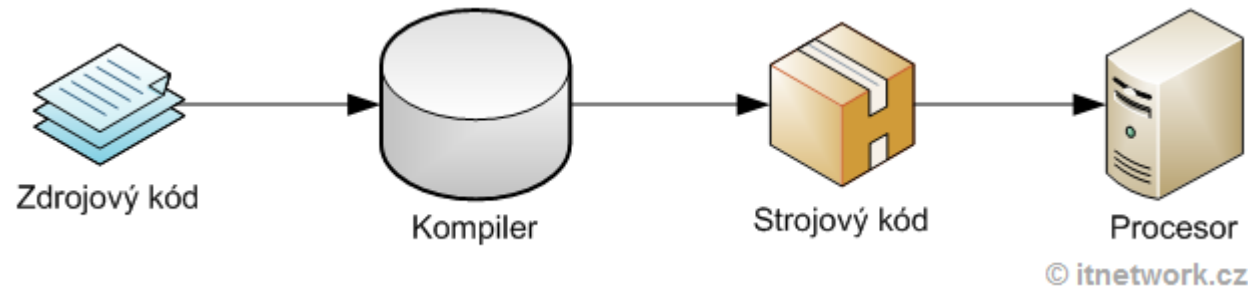
- Pro zápočet je potřeba:
 1. mít alespoň 75% docházku,
 2. získat **40 bodů**:
 - 2 domácí úkoly, každý za 15 bodů,
 - 24 bodů za úkoly na cvičeních,
- Kdo již umí programovat v Pythonu (nebo si to myslí), může příští týden zkusit napsat vstupní test.

Konzultace



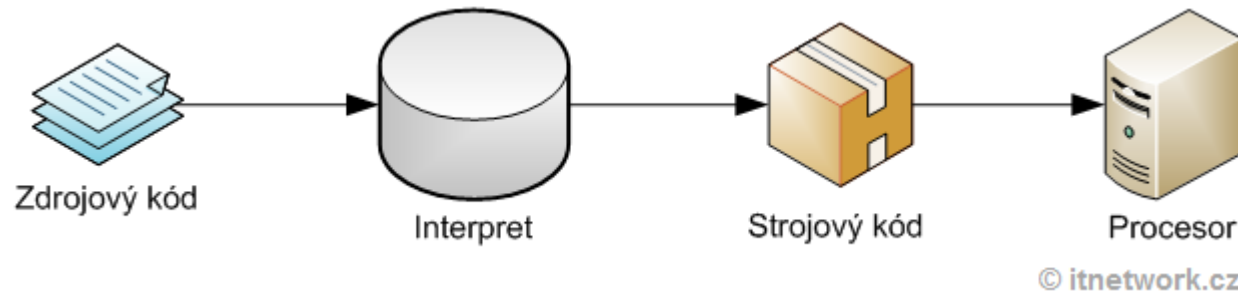
- v pracovně 5.071,
- každou středu (12.00 – 13.00),
- jindy po vzájemné domluvě,
- email: jiri.zacpal@upol.cz,
- MS Teams

Programovací jazyky



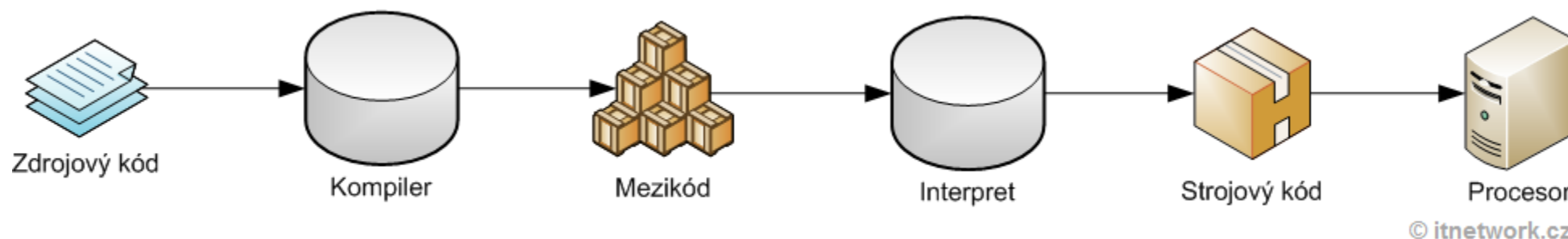
- **Výhody:**
 - **Rychlost.**
 - **Nepřístupnost zdroj. kódu** - Program se šíří již zkompilovaný, není jej možné jednoduše modifikovat pokud zároveň nevlastníte jeho zdroj. kód.
 - **Snadné odhalení chyb ve zdroj. kódu** - Pokud zdrojový kód obsahuje chybu, celý proces kompilace spadne a programátor je s chybou seznámen. To značně zjednodušuje vývoj.
- **Nevýhody:**
 - **Závislost na platformě.**
 - **Nemožnost editace.**
 - **Memory management** - Vzhledem k tomu, že počítač danému programu nerozumí a jen mechanicky vykonává instrukce, můžeme se někdy setkat s velmi nepříjemnými chybami s přetečením paměti. Kompilované jazyky obvykle nemají automatickou správu paměti a jsou to jazyky nižší (s nižším komfortem pro programátora). Běhové chyby způsobené zejména špatnou správou paměti se kompilací neodhalí.
- Příklad: [C](#), [C++](#).

Interpretované jazyky



- **Výhody:**
 - **Přenositelnost.**
 - **Jednodušší vývoj** - Ve vyšších jazycích jsme odstíněni od správy paměti, kterou za nás dělá tzv. garbage collector (řekneme si o něm v seriálu více). Často také nemusíme ani zadávat datové typy a máme k dispozici vysoce komfortní kolekce a další struktury.
 - **Stabilita** - Díky tomu, že interpret kódu rozumí, předejde chybám, které by zkompilovaný program jinak klidně vykonal. Běh interpretovaných programů je tedy určitě bezpečnější, dále umožňuje zajímavou vlastnost, tzv. reflexi, kdy program za běhu zkoumá sám sebe, ale o tom později.
 - **Jednoduchá editace.**
- **Nevýhody:**
 - **Rychlost.**
 - **Často obtížné hledání chyb** - Díky kompilaci za běhu se chyby v kódu objeví až v tu chvíli, kdy je kód spuštěn. To může být někdy velmi nepříjemné.
 - **Zranitelnost** - Protože se program šíří v podobě zdrojového kódu, každý do něj může zasahovat nebo krást jeho části.
- Příklad: [PHP](#).

Jazyky s virtuálním strojem



■ Výhody:

- **Odhalení chyb ve zdrojovém kódu** - Díky kompilaci do CIL (Common Intermediate Language) jednoduše odhalíme chyby ve zdrojovém kódu.
- **Stabilita** - Díky tomu, že interpret kódu rozumí, zastaví nás před vykonáním nebezpečné operace a na chybu upozorní. Můžeme také provádět reflexi (i když pro CIL, ale od toho jsme většinou odstíněni).
- **Jednoduchý vývoj** - Máme k dispozici hitech datové struktury a knihovny, správu paměti za nás provádí garbage collector.
- **Slušná rychlost** - Rychlost se u virtuálního stroje pohybuje mezi interpretem a kompilerem. Virtuální stroj již výsledky své práce po použití nezahazuje, ale dokáže je cachovat, sám se tedy optimalizuje při čtenějších výpočtech a může dosahovat až rychlosti kompilery (Just In time Compiler). Start programu bývá pomalejší, protože stroj překládá společně využívané knihovny.
- **Málo zranitelný kód** - Aplikace se šíří jako zdrojový kód v CIL, není tedy úplně jednoduše lidsky čitelná.
- **Přenositelnost** - Asi je jasné, že hotový program poběží na každém železe, na kterém se nachází virtuální stroj. To ale není vše, my jsme dokonce nezávislí i na samotném jazyce. Na jednom projektu může dělat více lidí, jeden v C#, druhý ve [Visual Basic](#) a třetí v C++. Zdrojové kódy se poté vždy přeloží do CILu.
- Příklad: [Java](#), [C#](#), [Python](#).

Język Python

Historie jazyka Python



- Programovací jazyk Python vytvořil v roce 1989 Guido van Rossum, který se na jeho vývoji podílí dodnes.
- Každý rok vycházejí průběžné aktualizace jazyka.
- Verze:
 - Python 2 – poslední verze 2.7.18 (20. duben 2020).
 - Python 3 – poslední verze 3.10.2 (14. ledna 2022).
- Jazyk je pojmenován po britské televizní show Monty Pythonův létající cirkus.
- V průběhu let se stal jazyk Python velice populární.

Proč používat Python?

- Kvalitní software
 - čitelný
 - přehledný
- Vysoká produktivita práce
- Multiplatformnost
- Podpora knihoven
- Integrace komponent

Vlastnosti jazyka Python

- Python je dynamický **interpretovaný jazyk**.
- Někdy bývá zařazován mezi takzvané **skriptovací jazyky**.
- Python je hybridní jazyk (nebo také **multiparadigmatický**).
- K význačným vlastnostem jazyka Python patří jeho jednoduchost z hlediska učení. Bývá dokonce považován za jeden z nejvhodnějších programovacích jazyků pro začátečníky.
- Produktivnost z hlediska rychlosti psaní programů.

Implementace jazyka Python



- **CPython**

- Standardní Python je implementován v jazyce C, tato implementace je označována CPython. V ní probíhá další vývoj jazyka Python. Verze jazyka Python jsou zveřejňovány jak v podobě zdrojového kódu, tak v podobě přeložených instalačních balíčků pro různé cílové platformy.

- **Jython**

- Jython je implementace Pythonu pro prostředí JVM. Je implementován v jazyce Java.

- **IronPython**

- IronPython je implementace Pythonu pro prostředí .NET/Mono. Za výhody lze považovat to, že se Python tímto stává jedním z jazyků pro platformu .NET. To současně znamená, že jej lze přímo využívat ve všech jazycích platformy .NET.

- **Brython**

- Brython je implementace Pythonu 3 v JavaScriptu. Jejím cílem je umožnit ve webovém prohlížeči programovat v jazyce Pythonu místo v JavaScriptu.

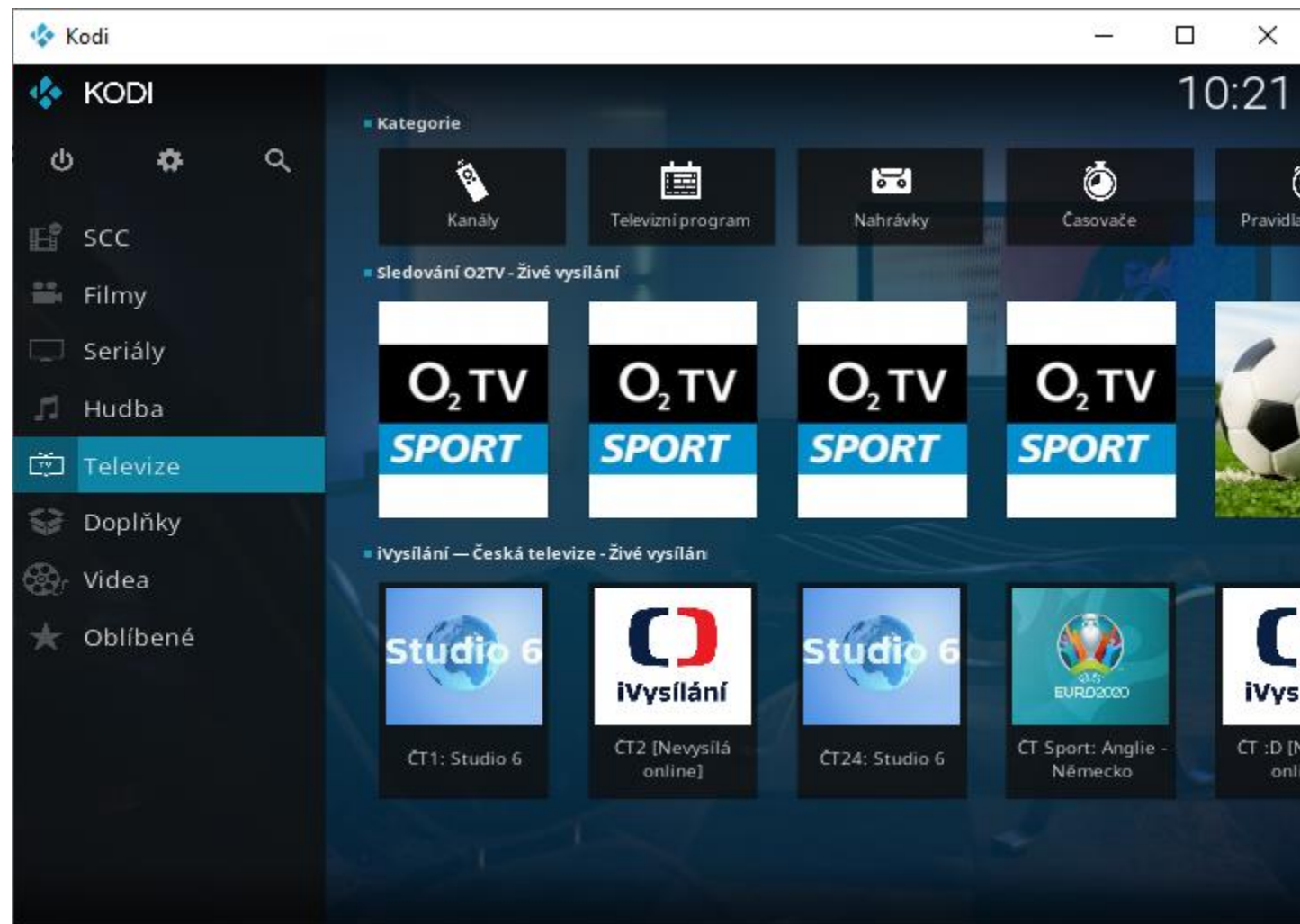
- **PyPy**

- alternativní interpret jazyka Python, který je zaměřen na výkon. Poslední verze PyPy implementuje Python 2.7.13 a 3.6.9.

Kodi



- Vytvořený v Pythonu.
- Verze 19 – přechod na verzi Python 3 - > velké problémy s doplňky.



Calibre



- Správce e-knih.
- Vytvořený v Pythonu.
- Verze 5 – přechod na verzi Python 3 - > nefunkční doplňky.

The screenshot shows the Calibre application window. The top toolbar contains icons for adding, editing, converting, viewing, downloading, rating, refreshing, saving, deleting, and saving to disk. The left sidebar shows a virtual library with search filters like 'Autoři', 'Jazyky', 'Série', 'Formáty', 'Vydavatel', 'Hodnocení', 'Zprávy', 'Štítky', 'Identifikátory', and 'Uložená hledání'. The main table lists books with columns for 'Název', 'Autoři', 'Přečteno', 'Hodnocení', 'Kdy přečteno', and 'Datum'. The book 'Noc světlohoše' by Peter Tremayne is selected. A detailed view on the right shows the book's cover, author, series, tags, and format. A description of the book is also visible.

| | Název | Autoři | Přečteno | Hodnocení | Kdy přečteno | Datum |
|----|---|---------------------|----------|-----------|--------------|-------------|
| 1 | Noc světlohoše | Peter Tremayne | | | | 22 čen 2021 |
| 2 | Bláto a mršiny | Ann Granger | ✓ | ★★★★ | 12 čen 2021 | 21 kvě 2021 |
| 3 | Pech | Klára Aycox | ✗ | | | 21 kvě 2021 |
| 4 | Když se vosy vyrojí | Marie Rejfová | ✓ | ★★★★★ | 21 kvě 2021 | 17 kvě 2021 |
| 5 | Inspektor s velkým mozem | František Niedl | ✓ | ★★★★★ | 03 kvě 2021 | 21 dub 2021 |
| 6 | Vánoční případ c. a k. policejního k... | Jiří Slaviček | ✗ | | | 13 dub 2021 |
| 7 | Lars pod palbou | Daniel Gris | ✓ | ★★★★★ | 19 bře 2021 | 11 bře 2021 |
| 8 | Lars útočí | Daniel Gris | ✓ | ★★★★★ | 11 bře 2021 | 27 úno 2021 |
| 9 | Tady, teď a tehdy | Mike Chen | ✓ | | 24 úno 2021 | 22 úno 2021 |
| 10 | Královská hraničářka | John Flanagan | ✗ | | | 16 úno 2021 |
| 11 | Ztracené příběhy | John Flanagan | ✗ | | | 16 úno 2021 |
| 12 | Spiknutí oběšenců | Vlastimil Vondruška | ✓ | ★★★★★ | 27 úno 2021 | 12 úno 2021 |
| 13 | Muž s kanadským trávníkem | Miloslav Švandrlík | ✓ | ★★★★★ | 16 úno 2021 | 28 led 2021 |
| 14 | Tři případy policejního komisaře P... | Jiří Slaviček | ✓ | | 12 dub 2021 | 16 led 2021 |
| 15 | Jarmarečník | Vlastimil Vondruška | ✗ | | | 16 led 2021 |
| 16 | Zločin na stříbrném plátně | Vilém Křížek | ✓ | ★★★★ | 20 bře 2021 | 13 led 2021 |
| 17 | Nejlepší víkend | Patrik Hartl | ✓ | ★★★★★ | 20 úno 2021 | 30 pro 2020 |
| 18 | V temnotách | Martin Stručovský | ✓ | ★★★★★ | 23 led 2021 | 18 pro 2020 |
| 19 | Zavátá sněhem | Martin Stručovský | ✓ | ★★★★★ | 10 led 2021 | 18 pro 2020 |

Autoři: Peter Tremayne
Série: XXXVIII z *Sestra Fidelma*
Štítky: Historická detektivka
Formáty: EPUB
Cesta: [Klikněte pro otevření](#)

V předvečer pohanského svátku Samhain najdou Eadulf a válečník Aidan v hranici mrtvého. Muž je oblečený jako řeholník a kdosi na něm vykonal rituální „trojnou smrt“. Případu se ujímá Fidelma s Eadulfem a pátrání je zavádí do tajemného kláštera, o jehož existenci neměli ponětí, byť se nachází nedaleko Cashelu.

calibre 5.1 vytvořil Kovid Goyal [677 z 909 knih, 1 vybrána] Nalezena aktualizace: 5.22.1 Rozložený Úlohy: 0

Vývojová prostředí

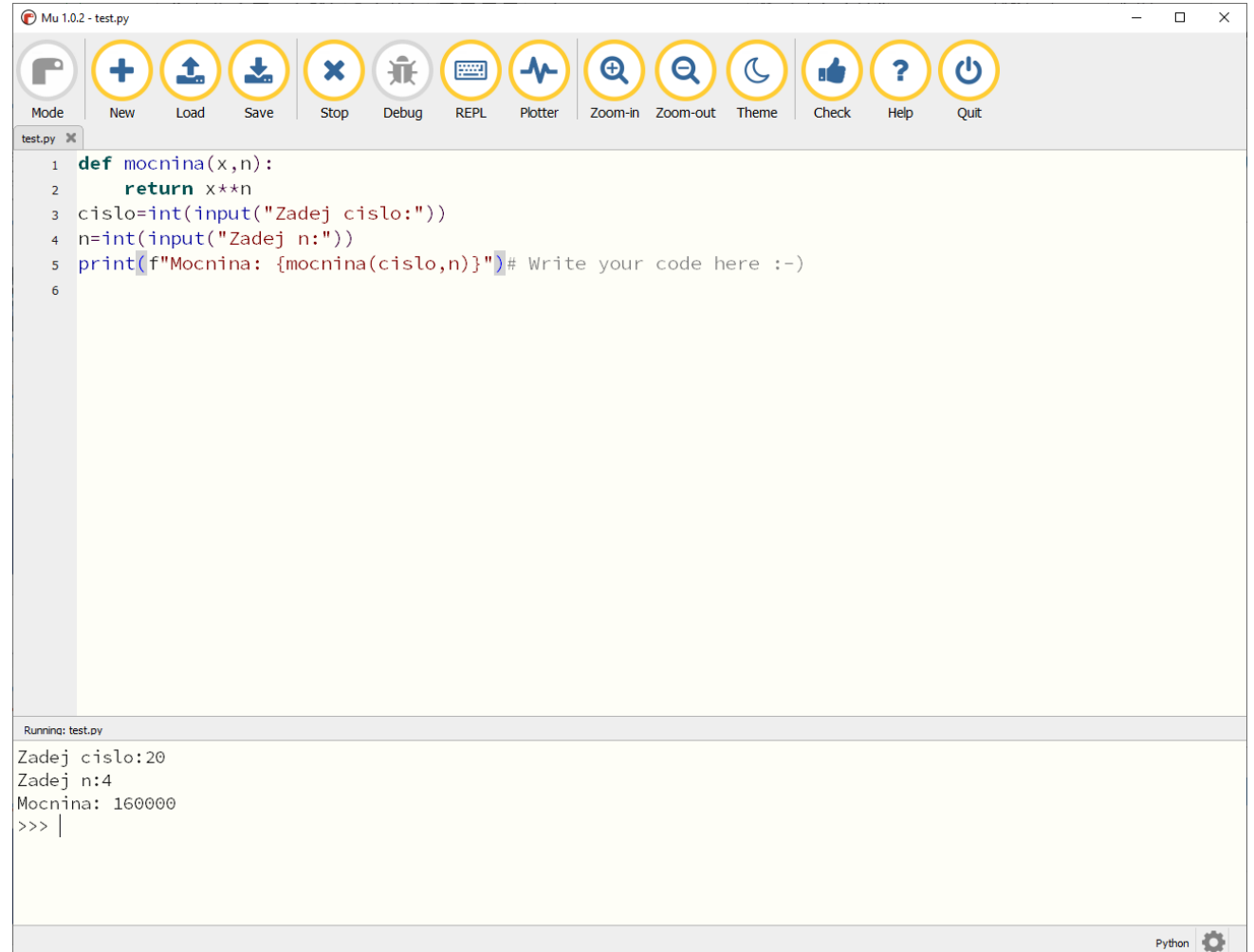
Program Python



- Instalační program na adrese <https://www.python.org/>
- Napsat program v jakémkoliv textovém editoru.
- Spustit program v shellu.

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64>python.exe  
d:/priklad.py  
Zadej cislo:15  
Zadej n:2  
Mocnina: 225
```

- Editace
- Spouštění
- Ladění



The screenshot shows the Mu Python IDE window titled "Mu 1.0.2 - test.py". The interface includes a toolbar with icons for Mode, New, Load, Save, Stop, Debug, REPL, Plotter, Zoom-in, Zoom-out, Theme, Check, Help, and Quit. The main editor area contains the following Python code:

```
1 def mocnina(x,n):  
2     return x**n  
3 cislo=int(input("Zadej cislo:"))  
4 n=int(input("Zadej n:"))  
5 print(f"Mocnina: {mocnina(cislo,n)}")# Write your code here :-)  
6
```

Below the editor is a console window titled "Running: test.py" showing the execution output:

```
Zadej cislo:20  
Zadej n:4  
Mocnina: 160000  
>>> |
```

The bottom right corner of the window shows the Python logo and a settings gear icon.

Visual Studio Code



- Lze použít na různé jazyky.
- Editace, ladění, spuštění.
- Rozšíření – extension.

The screenshot displays the Visual Studio Code editor with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The left sidebar shows icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The main editor area has two tabs: 'příklad.py' (active) and 'zpc1_02_typy_operatory.c'. The 'příklad.py' file contains the following Python code:

```
1 def mocnina(x,n):
2     return x**n
3 cislo=int(input("Zadej cislo:"))
4 n=int(input("Zadej n:"))
5 print(f"Mocnina: {mocnina(cislo,n)}")
```

Below the editor, the 'TERMINAL' panel is open, showing the command prompt output:

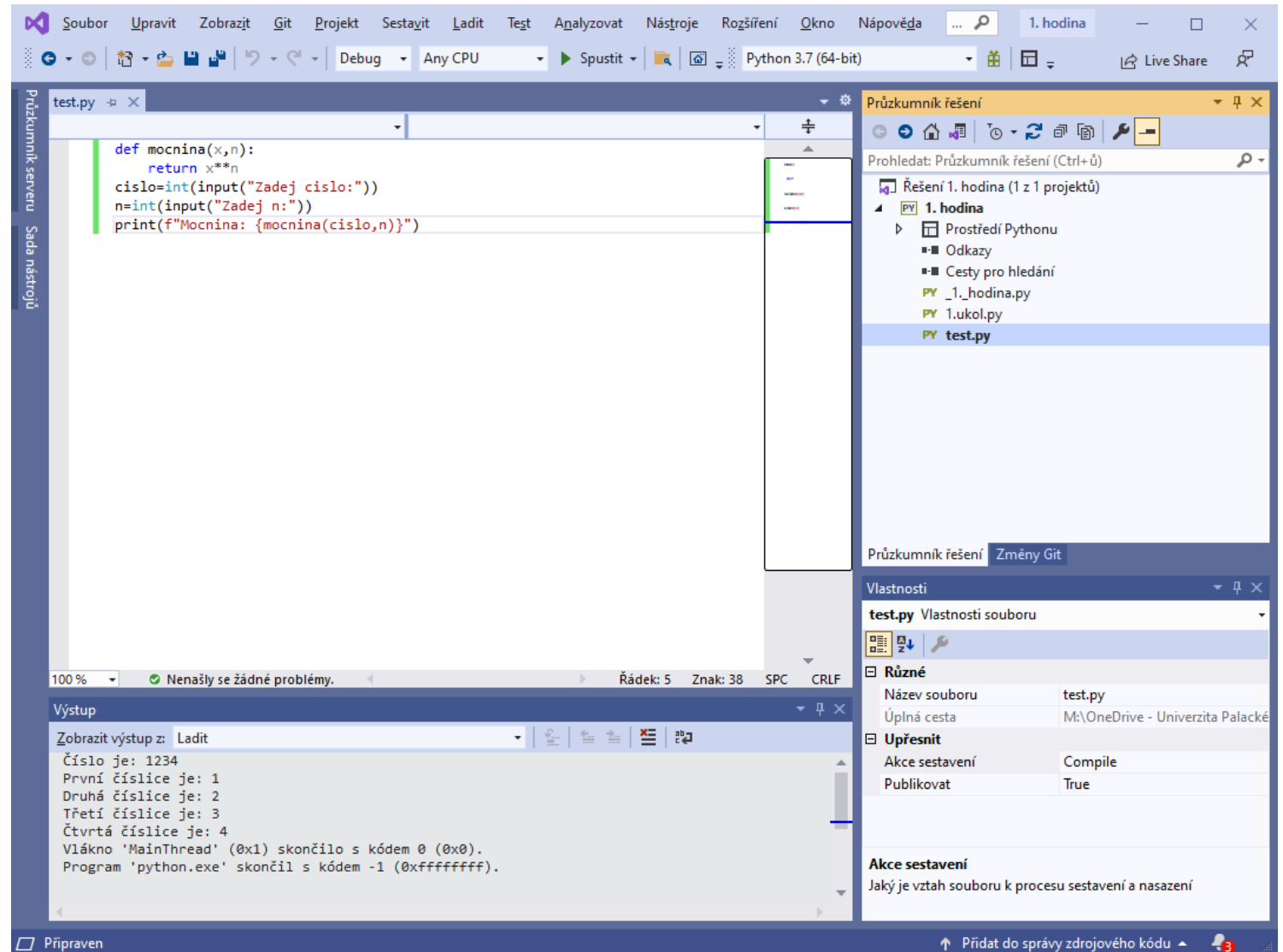
```
PS M:\OneDrive - Univerzita Palackého v Olomouci\dokumenty_pracovni\vyuka\ZS\xzpp1_zaklady_programovani_v_pythonu_1\priklady\1. hodina> .& 'C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\python.exe' 'c:\Users\Jirka\.vscode\extensions\ms-python.python-2021.6.944021595\pythonFiles\lib\python\debugpy\launcher' '58498' '--' 'm:\OneDrive - Univerzita Palackého v Olomouci\dokumenty_pracovni\vyuka\ZS\xzpp1_zaklady_programovani_v_pythonu_1\priklady\1. hodina\priklad.py'
Zadej cislo:20
Zadej n:4
Mocnina: 160000
PS M:\OneDrive - Univerzita Palackého v Olomouci\dokumenty_pracovni\vyuka\ZS\xzpp1_zaklady_programovani_v_pythonu_1\priklady\1. hodina>
```

The status bar at the bottom indicates 'Python 3.7.8 64-bit', '0 0 0' errors/warnings/hints, and 'Ln 5, Col 38'.

Visual Studio



- Lze použít na různé jazyky.
- Velmi robustní nástroj.
- Editace, ladění, spuštění.



Základy programování

Základní pojmy

- **hodnota**

- reprezentuje informace data,
- jsou reprezentována objekty.
- **Příklad:** číselná hodnota 123

```
>>> print(123)  
123
```

- **výraz**

- slouží pro vytváření hodnot.
- **Příklad:** 15 + 10

```
>>> print(15+10)  
25
```

- **příkaz**

- dává počítači instrukce, co má dělat,
- v Pythonu se na jeden řádek zapisuje jeden příkaz.

Příklad 1



- Spustíte si interpret Pythonu v příkazovém řádku. Zadejte:

```
>>>12
```

```
12
```

```
>>>
```

- Postup interpretu:
 - R (read) – načtení vstupu
 - E (eval) – vyhodnocení vstupu
 - P (print) – vytisknutí výsledku
 - L (loop) – opakování procesu

```
>>> print 01
```

```
File "<stdin>", line 1
```

```
    print 01
```

```
        ^
```

```
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(01)?
```

```
>>>
```

Příklad 1

- Zapište výraz 1+2:

```
>>> 1+2
```

```
3
```

```
>>>
```

- Zapište výraz 1+:

```
>>> 1+
```

```
File "<stdin>", line 1
```

```
1+
```

```
^
```

```
SyntaxError: invalid syntax
```

```
>>>
```

- Zapište výraz 1+2+3:

```
>>> 1+2+3
```

```
6
```

```
>>>
```


Příklad 1

- Zapište výraz $(1+2)+3$ a $1+(2+3)$:

```
>>> (1+2)+3
```

```
6
```

```
>>> 1+(2+3)
```

```
6
```

```
>>>
```

- Zapište výraz $1+2+3$) a $1+(2+3$:

```
>>> 1+2+3)
```

```
File "<stdin>", line 1
```

```
1+2+3)
```

```
^
```

```
SyntaxError: invalid syntax
```

```
>>> 1+(2+3
```

```
... )
```

```
6
```

```
>>>
```

- umožňují kombinovat jednoduché výrazy a tím vytvářet výrazy složitější,
- operátor obvykle chápeme jako operaci, kterou lze provést s nějakými **operandy** (argumenty)
- vlastnosti:
- **Priorita**
 - udává pořadí, ve kterém se operace provádějí
 - pořadí lze pozměnit použitím kulatých závorek
(závorky je ovšem dobré používat i tam, kde nejsou nutné, protože zvyšují přehlednost kódu)
- **Asociativita**
 - udává „směr“, ve kterém se vyhodnocují operace se stejnou prioritou (zleva nebo zprava)
 - lze opět ovlivnit použitím závorek
- **Aritmetika**

Aritmetické operátory

- umožňují kombinovat jednoduché výrazy a tím vytvářet výrazy složitější,
 - mění znaménko operandu (unární operátor)
 - + součet dvou operandů
 - rozdíl dvou operandů
 - * násobení dvou operandů
 - / dělení dvou operandů
 - % operace modulo (zbytek po celočíselném dělení prvního operandu druhým)
 - ** mocnina
 - // výsledek dělení prvního operandu druhým zaokrouhlený směrem dolů (floor division)

Příklad 1

- Vyzkoušejme si nové operátory.

```
>>> 5-2
```

```
3
```

```
>>> 5*2
```

```
10
```

```
>>> 5//2
```

```
2
```

```
>>> 5%2
```

```
1
```

- Všimneme si, že pomocí operátoru - můžeme vytvořit záporné číslo.

```
>>> 0-1
```

```
-1
```

Příklad 1



- U operátorů `//` a `%` dochází k chybě dělení nulou v případě, že se druhý podvýraz vyhodnotí na nulu.

```
>>> 4%0
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
ZeroDivisionError: integer division or modulo by zero
```

```
>>> 4//0
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
ZeroDivisionError: integer division or modulo by zero
```

- Zapište výrazy $1+2*3$ a $(1+2)*3$:

```
>>> 1+2*3
```

```
7
```

```
>>> (1+2)*3
```

```
9
```

```
>>>
```

- entity, které mohou obsahovat hodnoty,
- každá proměnná má své jedinečné jméno, označované jako **identifikátor** proměnné,
- Identifikátory v jazyce Python jsou libovolná kombinace
 - malých písmen (a–z),
 - velkých písmen (A–Z),
 - číslic (0–9)
 - a znaku _ (podtržítko).
- Identifikátor nesmí začínat číslicí a jako identifikátor nelze použít klíčová slova jazyka, která mají speciální význam.

Použití proměnné



- **Typ proměnné** – co lze do proměnné vložit.
 - staticky typované a dynamicky typované jazyky.
 - Python je dynamicky typovaný jazyk. To znamená, že typ proměnné je určen typem objektu, který reprezentuje hodnotu, jež proměnná obsahuje.
 - Příklady:
 - 42 je objekt typu „celé číslo“,
 - 42.0 je objekt typu „desetinné číslo“.
- Přiřazení hodnoty proměnné
 - Pomocí příkazu přiřazení.
 - Příklad:
`a=123`
- Proměnnou není potřeba deklarovat.
- Vytvoří se při prvním použití.
- Pokud ji chceme použít jako součást výrazu, musí mít přiřazenou hodnotu.

Příklad



- Napište výraz:

```
>>> b=a
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'a' is not defined

```
>>>
```

Napište výraz:

```
>>> a=123
```

```
>>> b=a
```

```
>>> b
```

```
123
```

```
>>>
```


Standardní výstup

Visual Studio Code



- Spustíte si program Visual Studio Code.
- Vytvořte si adresář programy.
- Ve VSC zadejte příkaz File – Open folder a vyberte tento adresář.
- Vytvořte si soubor seminar01.py.

The screenshot shows the Visual Studio Code interface with a Python file named `priklad.py` open. The code in the editor is as follows:

```
1 def mocnina(x,n):
2     return x**n
3 cislo=int(input("Zadej cislo:"))
4 n=int(input("Zadej n:"))
5 print(f"Mocnina: {mocnina(cislo,n)}")
```

The left sidebar shows the 'RUN AND DEBUG: RUN' panel with a 'Run and Debug' button. The bottom panel shows the 'TERMINAL' with the following output:

```
PS M:\OneDrive - Univerzita Palackého v Olomouci\dokumenty_pracovni\vyuka\ZS\xzpp1_zaklady_programovani_v_pythonu_1\priklady\1. hodina> . & 'C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\python.exe' 'c:\Users\Jirka\.vscode\extensions\ms-python.python-2021.6.944021595\pythonFiles\lib\python\debugpy\launcher' '58498' '--' 'm:\OneDrive - Univerzita Palackého v Olomouci\dokumenty_pracovni\vyuka\ZS\xzpp1_zaklady_programovani_v_pythonu_1\priklady\1. hodina\priklad.py'
Zadej cislo:20
Zadej n:4
Mocnina: 160000
PS M:\OneDrive - Univerzita Palackého v Olomouci\dokumenty_pracovni\vyuka\ZS\xzpp1_zaklady_programovani_v_pythonu_1\priklady\1. hodina>
```

The status bar at the bottom indicates 'Python 3.7.8 64-bit', 'Ln 5, Col 38', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', and other settings.

Výstup na obrazovce

- K výstupu na obrazovku slouží funkce

`print(entita)`

- Jednoduchý výstup:
`print("Ahoj světe")` # zobrazí: Ahoj světe
- Výstup proměnné:
`a = 42`
`print("Ahoj světe", a)` # zobrazí: Ahoj svete 42
- Formátovaný výstup:
`a = 42`
`print(f"Ahoj svete {a}")` # zobrazí: Ahoj svete 42

`pi = 3.14159265359`
`print(f"{pi}")`
`print(f"{pi:.2f}")` # vypíše 3.14
`print(f"{pi:.0f}")` # vypíše 3

Příklad



- Napište do souboru tyto příkazy:

```
a=1
```

```
print(a)
```

- Program spusťte.
- Příkazy upravte takto:

```
a=1
```

```
print(a/0)
```

- Program spusťte.
- Program skončil chybou:

```
program-1.py seminar01.py Python - C
seminar01.py > ...
1 a=1
2 print(a/0)

Exception has occurred: ZeroDivisionError
division by zero

File "M:\OneDrive - Univerzita Palackého v Olomouci\dokumenty_pracovni\vyuka\ZS\zpp1_zaklady_programovani_v_pythonu\programy\seminar01.py",
line 2, in <module>
    print(a/0)
```

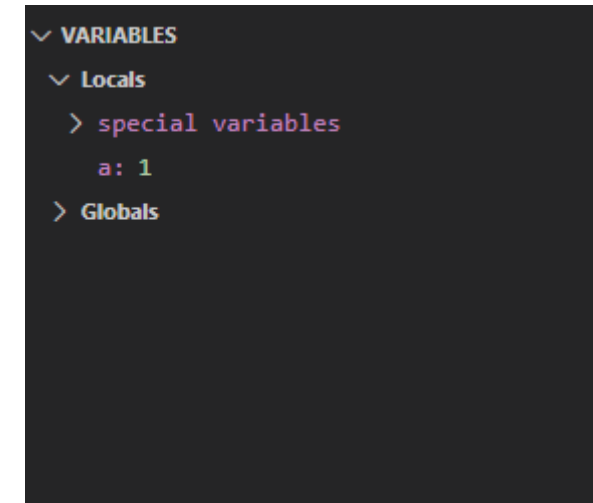
Příklad



- Program můžeme ladit tak, že před určitý příkaz vložíme zarážku.
- Po spuštění programu, se vykonávání programu zastaví na tomto příkazu.
- K ladění můžeme využít příkazy z tohoto panelu:



- Aktuální vazby proměnných jsou v sekci Variables:
- Do kódu můžeme zapsat komentář:
#První hodina.
X=5 #Promenna x



Bodovaný úkol



- Napište program, který pro zadané čtyřciferné číslo vypíše všechny jeho číslice. Povoleno je použít jen ty příkazy z Pythonu, které byly dnes představeny.

Příklad výstupu:

```
Číslo je: 1234  
První číslice je: 1  
Druhá číslice je: 2  
Třetí číslice je: 3  
Čtvrtá číslice je: 4  
Press any key to continue . . .
```