Milí Junioři,

jak mnozí z vás poznali, třetí úkol „Poznej chytrý dům a navrhni svůj!“ vám představil pražskou Villu Sophiu s „vlastním mozkem“!

Výherci, kteří zaslali správné odpovědi a své návrhy vysněného domu do včerejší půlnoci, byli o své výhře informováni e-mailem.

Čtvrtý úkol Juniorády se zaměří na další z osmi fakult VUT. Tentokrát se jedná o oblast INFORMATIKY!

Vaše řešení opět zasílejte na e-mail kralovan@vutbr.cz do středeční půlnoci (22.4. 23:59).

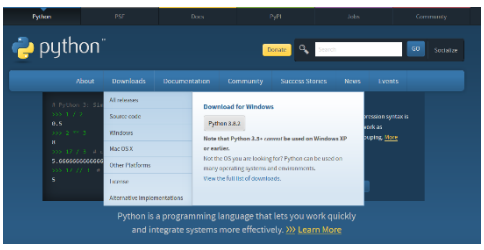
Tři vylosovaní výherci budou kontaktováni a mohou se těšit na vysloužen

INFORMATIKA a s ní spojené chytré technologie nás v dnešní době obklopují na každém kroku. Abychom si však s počítači rozuměli, musíme se naučit komunikovat jejich jazykem. Proto se tento v pořadí čtvrtý úkol Juniorády do hloubky zaměří na jeden z jednodušších programovacích jazyků, a tím je *Python*.

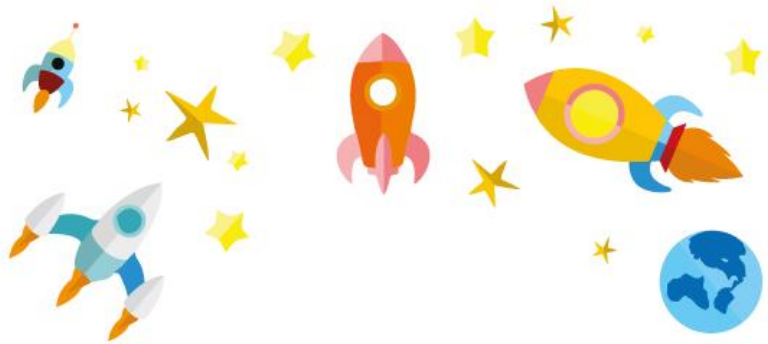
„Komunikuj s počítačem!“

Abys mohl/a prozkoumat Python a nahlédnout do vzorců jeho komunikace, budeš potřebovat:

a) Zadarmo stáhnout Python do svého počítače z oficiálních stránek vývojářů: https://www.python.org/



b) Trochu trpělivosti, pokud ti to hned napoprvé nepůjde 😊



1. **Drž se návodu**, který začíná na další straně (do programu vepisuj texty v šedých rámečcích).

2. **Kreativitě se meze nekladou** – můžeš použít i jiná slova, než která jsou v návodu (např. jméno Adam lze nahradit tvým jménem).

3. **Pamatuj**, nejde o to mít vše dokonale zpracované, důležitější je seznámit se s Pythonem a alespoň malinko pochopit jeho fungování.

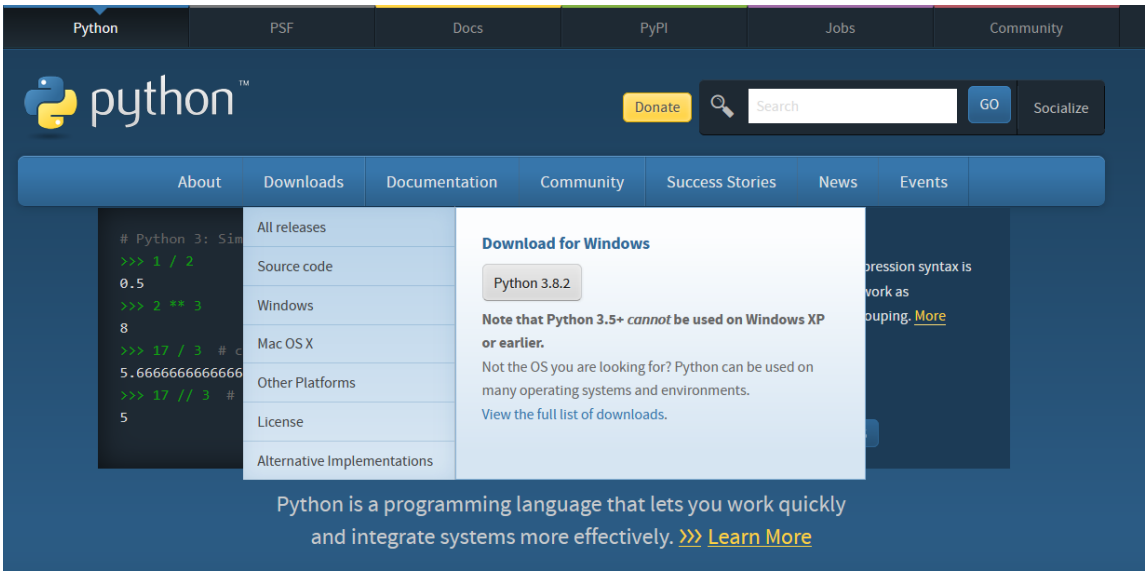
    a. Proto je návod rozdělen na dvě části. Druhá část je pro nadšené IŤáky, kteří se s programovacím jazykem dokonale domluví a budou moci splnit úkoly na poslední straně zadání.

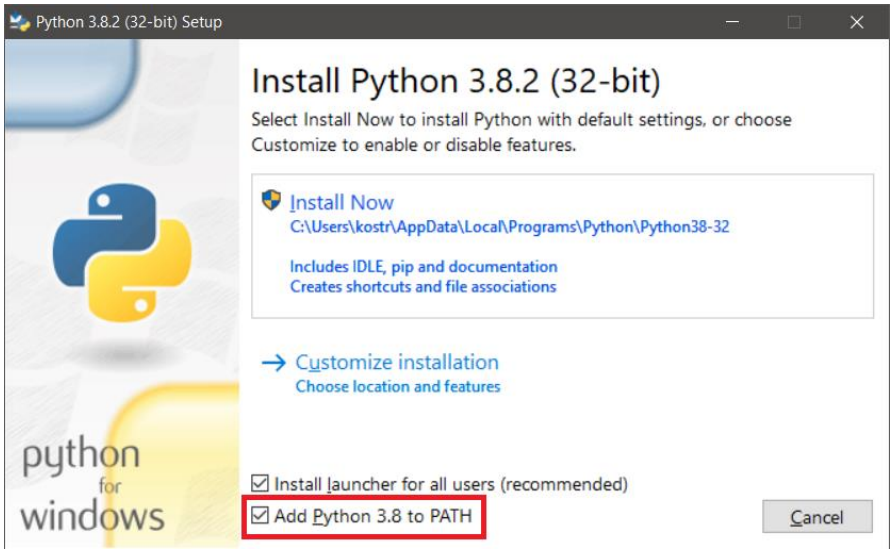
4. Tvou **komunikaci s Pythonem ulož** a zašli ho na e-mail: kralovan@vutbr.cz jako všechny předchozí úkoly 😊

NÁVOD, JAK S TVÝM POČÍTAČEM MLUVIT

Aby počítače a naše společnost mohly fungovat, musí mít schopnost se spolu domluvit. Ale jak jsme toho docílili, když například já mluvím česky a počítač jenom v jedničkách a nulách? On nedokáže mluvit česky a já mu nedokážu říct nic v číslech (např. malé „a“ se do řeči počítače přeloží jako 111101, a pokud bychom chtěli takto s počítačem mluvit, tak by to bylo přinejmenším nepraktické). A proto člověk vytvořil **programovací jazyky.** To jsou jazyky, které jsou podobné angličtině a dokážeme jim rozumět my i počítače. **A my si teď jeden ukážeme a něco se v něm i naučíme.**

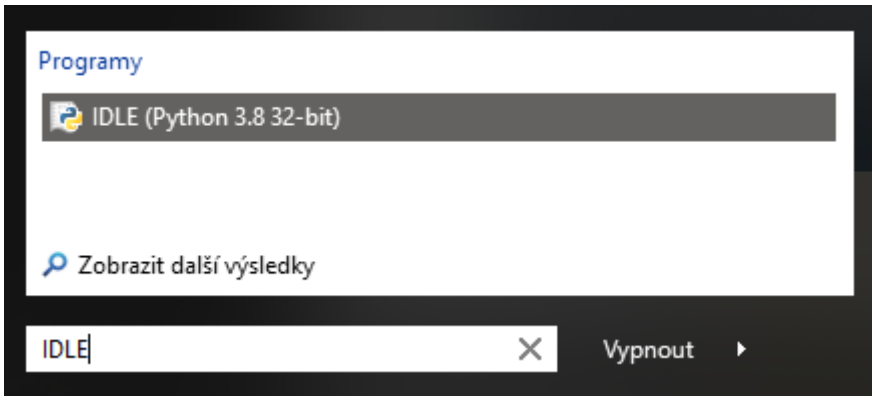
Jazyk, který jsme vybrali, se jmenuje **Python.** Jeho výhodou je právě jeho jednoduchost, a proto je naprosto dokonalý pro začátečníky. Některé jeho příkazy se čtou téměř jako normální anglická věta. Python ale musíte nejprve do svého počítače **zdarma stáhnout z oficiálních stránek vývojářů:** https://www.python.org/





Stačí jednoduše kliknout na **Install Now** a všechno se nainstaluje za vás, **ale pozor!** Je důležité, abyste zaškrtli **„Add Python 3.8 [1] to PATH“.**

[1] 3.8 není jediné číslo, které se tam může nacházet, verze se může lišit, na instalaci ani běh námi vytvořených programů to nebude mít žádný vliv



Poté stačí jednoduše vyhledat ve vašem počítači (ikonka lupy – hledání), do kterého jste program stáhli, integrované vývojové prostředí jazyka Python (IDLE) a **začít konečně programovat!!**

Každý programátor, ať už začátečník, nebo profesionál, u každého nového jazyka začne základním programem známým jako **„Hello World!“.** Tento program umí pouze to, že vypíše „Hello World!“, a přitom je to naprosto geniální zkouška všeho, co programátor potřebuje odzkoušet před složitějším kódováním. **Vyzkoušíte si základy syntaxe** (programovacího pravopisu) a jak program spustit. Zvládneme to takto:

1) Spustíme IDLE[2]
2) Kurzorem myši přejedeme na **File** a zvolíme **New File** nebo použijeme klávesovou zkratku **Ctrl + N**
3) Do volného textového pole (které se nám nyní otevřelo) vepíšeme následující příkaz[3] [4]:

```
print('Hello World!')
```

4) Kurzorem myši přejedeme na **Run** a vybereme **Run Module** nebo stiskneme **F5**
5) V prostředí IDLE by se nám měla vypsat zpráva „Hello World!“

Co se vlastně stalo? Pythonu jste napsali, že byste chtěli zadat funkci print() (všimněte si, že to můžeme přeložit jako „vytiskni“) a té jste předali argument „Hello World!“. Tato funkce následně vypíše zadaný argument na obrazovku.

A co je vlastně ten „Hello World!“? Samozřejmě, je to věta, ale před malou chvílí jsme si řekli, že počítač vidí a rozumí jenom jedničkám a nulám. Tak jak je možné, že tohle všechno poznal? Naštěstí to vývojáři vymysleli chytře a jednotlivým písmenům už velmi dávno přiřadili číselné hodnoty (např. malé „a“ má přiřazenou hodnotu 61 a tuto hodnotu jsme už dříve vyjádřili binárně, tedy v jedničkách a nulách). Takže funkci print() jsme vlastně předali strašně dlouhou řadu jedniček a nul. **Jak ale počítač poznal, že myslíme „a“ a ne 61?** Proto se zavedl pojem **datový typ.**

Datový typ je vlastně přiřazení významu některé řadě jedniček a nul. Ačkoliv je umí počítač číst, význam jim dal člověk. Věta „Hello World“ je typu **String**, což by se dalo velmi volně přeložit právě jako věta. Dalšími typy jsou například **int** (integer, neboli decimální celé číslo, jako je 1, 42, 98445, apod.) nebo třeba **float** (floating-point-number, číslo s desetinnou čárkou jako je 1.0[5], 0.00005, apod.)

---

[2] Toto není jediný způsob, dokonce ani nejpraktičtější, ale pro naše účely je naprosto ideální.
[3] Silně doporučuji skutečně jakékoliv programy, které budete chtít použít z jakéhokoliv návodu, na začátku vaší programovací cesty doopravdy ručně opisovat, naučíte se tak víc a budete i rychleji psát.
[4] POZN. Je jedno, jestli použijete jednoduché uvozovky, nebo zdvojené uvozovky.
[5] Je velmi důležité psát tečku místo čárky, jak jsme jinak z češtiny zvyklí.

Ale co když chceme tisknout i jiné zprávy, než „Hello World!“? Samozřejmě, vždy jsme schopni do kódu přímo vepsat jinou zprávu, např. něco jako „Hello Adam!“. Ne však každý člověk se jmenuje Adam, takže potřebujeme, aby se kód dokázal měnit. Proto existuje něco jako **proměnná,** anglicky variable, která nám toto umožní.

Pozor: Proměnné musíte vždy vytvořit, než je budete používat, protože počítač jinak neví, co to znamená.

Proměnná vlastně zastupuje jistou **část paměti počítače.** Říkáme tím počítači, že by si měl dávat pozor a informaci si zapamatovat, protože ji bude potřebovat později. Proměnným dáváme jméno my sami. Je tady jenom pár pravidel, např. že nesmí začínat číslem.

Funkci proměnné si můžeme vyzkoušet na tomto malém programu:

```
jmeno = 'Adam'
print('Hello' + ' ' + jmeno)
```

V Pythonu můžeme stringy sčítat. V příkladu výše jsme sečetli slovo Hello s mezerou a čímkoliv co je ve jméně, problém by ale nastal, pokud bychom do jména nastavili číslo bez uvozovek (ty značí, že se jedná o string), proto bychom měli vždy tyto proměnné převádět pomocí funkce str(), která svým názvem napovídá, že se jedná o převod na **str**ing.

```
jmeno = 'Adam'
print('Hello' + ' ' + str(jmeno))
```

Většina užitečných aplikací, které známe, většinou nepracuje jenom s něčím takto naprogramovaným, a proto vyžadují určitou aktivitu ze strany uživatele (např. abychom mohli v chatu poslat zprávu, musíme ji nejdřív napsat a předat chatovací aplikaci).

Aby náš program v Pythonu dostal od uživatele vstup (tak se tomu odborně říká), používá funkci input(), která jako argument přijímá string, který se zobrazí. Poté program čeká na vstup, který příjme **po stisku klávesy enter.**

```
jmeno = input('Zadejte jmeno: ')
print('Hello' + ' ' + str(jmeno))
```

To už je sám o sobě dost dobře fungující program. Ale co kdybychom chtěli, aby se počítač zachoval jinak, když mu dáme nějaké konkrétní jméno? Na to musíme počítači dát určité podmínky, podle kterých se bude rozhodovat. K tomu využijeme konstrukce if-elif-else, což by se dalo volně přeložit jako když-jinakkdyž-jinak[6], a budeme muset použít **operátory na porovnávání, známe tyto:**

1) == rovná se a != nerovná se
2) < je menší a > je větší
3) <= je menší nebo rovná se a >= větší nebo rovná se

Mimochodem elif můžeme přidat kolikrát chceme a celá konstrukce funguje, jako bychom byli na rozcestí a pokoušeli si vybrat jednu z cest. To, co patří pod else, se stane pouze pokud by ani if, nebo ani jeden elif neplatil.

[6] elif je naprosto dobrovolný a nemusíme ho využít, v některých případech nemusíme využívat ani else

```
jmeno = input('Zadejte jmeno: ')
if jmeno == 'Adam':
    print('Jmeno Adam jsem uz videl')
elif jmeno == 'python':
    print('Python je programovací jazyk.')
else:
    print('Hello' + ' ' + str(jmeno))
```

Mezery před print() se nazývají odsazení, a pokud bychom je tam nenapsali, pak program nepojede správně, a navíc by se velmi špatně četl. **Napíšeme je pomocí klávesy tab.**

Všimněte si dvojtečky na konci řádku, dala by se velmi volně přeložit jako „tak“. Po ní můžete psát příkazy co se mají provést, pokud platí podmínka. Příkazy patří pod podmínku, dokud jsou odsazené.

KDO CHCE, MŮŽE V TÉTO FÁZI SKONČIT, KOMUNIKACI S POČÍTAČEM ULOŽIT NEBO VYFOTIT A ZASLAT NA EMAIL. 😊 TAKTO SPĚNÝ ÚKOL BUDE TAKTÉŽ ZAPOJEN DO SLOSOVÁNÍ O CENY!

Kdo si ale naopak přeje, může v objevování Pythonu pokračovat, objevit složitější příkazy a splnit následující úkoly pro pokročilé IŤáky!

TAK TEDY POKRAČUJEME:

Náš už poměrně složitý program má už jenom jednu chybu a to tu, že jede pouze jednou. Proto v programování používáme takzvané **cykly**. Ty už jsou trochu složitější, protože jich je hned několik typů, ale my se budeme bavit jenom o dvou nejrozšířenějších, a těmi jsou **for-loop a while-loop.**

```
for i in range(5):
    print(str(i))
    if i == 5:
        print('For loop končí')

print('A teď while loop:')

j = 0
while j < 5:
    print(str(j))
    j += 1
    if j == 5:
        print('While loop končí')
```

Tyto dva zápisy a cykly jsou naprosto identické a mají naprosto stejnou funkčnost. Abychom si ukázali, že se programovací jazyky dají skutečně číst téměř jako normální angličtina, tak si zkusíme s trochou představivosti přeložit větu **for i in range(5): „Pro i v rozsahu 5 tak…“** - tedy cyklus se bude provádět, dokud je v rozsahu pěti, a podle rozhodnutí vývojářů to znamená po pětku. **While j < 5: by se zase dalo přeložit jako „Dokud platí, že j je menší než 5 tak…“**
Ještě bychom si měli přeložit příkaz **j += 1: „K proměnné j přičti jedničku“**

Postupně jsme si tady vyzkoušeli většinu základních praktik při programování, na kterých se dá dále stavět. Pokud vás to bavilo, rozhodně pokračujte v samostudiu a rozšiřujte svůj IT rozhled 😊

Pokud byste se o Pythonu chtěli dozvědět více, navštivte stránky https://www.py.cz/FrontPage, které jsou česky a obsahují mnoho užitečných materiálů.

Mezi programátory je velmi oblíbený i jiný operační systém než Windows. Jedná se například o Linux, který je jedním z Unixových operačních systémů. Díky němu funguje internet tak, jak ho známe a používá ho naprostá většina serverů. Android je také založen na Linuxu, a Linux je i příbuzný operačnímu systému zařízení Apple. Pokud byste ho chtěli vyzkoušet na svém domácím počítači, můžete využít technologie Virtual Machine.

ÚKOL Č. 1

Se všemi výše uvedenými znalostmi ohledně fungování Pythonu můžete vytvořit program, který bude umět takto komunikovat:

1) třikrát se zeptá uživatele na jméno (použijte jeden z cyklů);
2) pokud uživatel zadá vaše jméno (které uložíte do proměnné a nebudete ho měnit), tak vám napíše speciální zprávu, pokud zadá VUT, tak taky vypíše speciální zprávu, ovšem ve všech ostatních případech (else) napíše „neznám“;
3) nakonec vypíše „končím!“.

ÚKOL Č. 2

Pro programování je matematika velmi důležitá. Nepotřebujete ji naprosto nutně, ale složitější věci bez ní neprovedete. Příkladem využití matematiky a logiky je algoritmus na hru „uhodni číslo“, který využívá metody bisection search. Myslete si v duchu číslo od 0 do 100 a pomocí příkazů „more“ nebo „less“ svému počítači odpovídejte do té doby, než vaše číslo uhodne 😊

Zde je návod:

```
x_min = 0
x_max = 100
msg = ''

print('Mysli si číslo mezi 0 a 100')

while msg != 'ano':
    guess = (x_min + x_max) / 2
    print('Je tvoje číslo: ' + str(int(guess)) + '?')
    print('Pokud ano. napiš "ano", jinak napiš "more", nebo "less"')
    msg = input('> ')
    if msg == 'more':
          x_min = guess
    if msg == 'less':
          x_max = guess

print('Jo! Našel jsem ho!')
```