Úvod do jazyka Python a počítačového programování

(Určeno pro vnitřní potřebu SOUE Plzeň, kopírování bez předchozího souhlasu je zakázáno)

Modul 1

V tomto modulu se dozvíte o:

- základech počítačového programování, tj. jak funguje počítač, jak se provádí program, jak je definován a konstruován programovací jazyk
- rozdíl mezi kompilací a interpretací
- co je to Python, jaké je jeho postavení mezi ostatními programovými jazyky a čím se liší různé verze Pythonu.

1.1.1.2 Programování - úplné základy

Jak funguje počítačový program?

Cílem tohoto kurzu je ukázat vám, co je jazyk Python a k čemu se používá. Začneme od úplných základů.

Program umožňuje používat počítač. Bez programu není počítač, i ten nejvýkonnější, ničím jiným než objektem. Stejně tak bez přehrávače není klavír nic víc než dřevěná bedna.

Počítače jsou schopny provádět velmi složité úlohy, ale tato schopnost není vrozená. Povaha počítače je zcela jiná.

Dokáže provádět pouze extrémně jednoduché operace. Počítač například nedokáže sám od sebe pochopit hodnotu složité matematické funkce, i když to v blízké budoucnosti není mimo možnosti.

Současné počítače dokáží vyhodnotit pouze výsledky velmi základních operací, jako je sčítání nebo dělení, ale dokáží to velmi rychle a mohou tyto činnosti opakovat prakticky libovolný početkrát.

Představte si, že chcete zjistit průměrnou rychlost, které jste dosáhli během dlouhé cesty. Znáte vzdálenost, znáte čas, potřebujete rychlost.

Počítač to samozřejmě dokáže vypočítat, ale počítač nezná takové věci, jako je vzdálenost, rychlost nebo čas. Proto je nutné dát počítači pokyn:

- přijmout číslo představující vzdálenost;
- přijmout číslo představující dobu jízdy;

- vydělit první hodnotu druhou a výsledek uložit do paměti;
- zobrazit výsledek (představující průměrnou rychlost) v čitelném formátu.

Tyto čtyři jednoduché akce tvoří program. Tyto příklady samozřejmě nejsou formalizované a jsou velmi vzdálené tomu, čemu počítač rozumí, ale jsou dostatečně dobré na to, aby se daly přeložit do jazyka, který počítač dokáže přijmout.

Klíčovým slovem je jazyk.

1.1.1.2 Programování - úplné základy

Přirozené jazyky vs. programovací jazyky

Jazyk je prostředek (a nástroj) k vyjadřování a zaznamenávání myšlenek. Všude kolem nás je mnoho jazyků. Některé z nich nevyžadují ani mluvení, ani psaní, například řeč těla; je možné velmi přesně vyjádřit své nejhlubší pocity, aniž bychom řekli jediné slovo.

Dalším jazykem, který používáte každý den, je vaše mateřština, kterou používáte k projevení své vůle a k úvahám o realitě. Také počítače mají svůj vlastní jazyk, který se nazývá strojový jazyk a je velmi primitivní.

Počítač, i ten technicky nejdokonalejší, postrádá byť jen stopu inteligence. Dalo by se říci, že je jako dobře vycvičený pes - reaguje pouze na předem daný soubor známých příkazů.

Příkazy, které rozpoznává, jsou velmi jednoduché. Můžeme si představit, že počítač reaguje na příkazy typu "vezmi toto číslo, vyděl ho jiným a výsledek ulož".

Úplná sada známých příkazů se nazývá **seznam instrukcí**, někdy se zkracuje na **IL (instruction language)**. Různé typy počítačů se mohou lišit v závislosti na velikosti jejich **IL (instruction language)** a instrukce mohou být u různých modelů zcela odlišné.

Poznámka: strojové jazyky vyvíjejí lidé.

Žádný počítač není v současné době schopen vytvořit nový jazyk. To se však může brzy změnit. Stejně jako lidé používají řadu velmi odlišných jazyků, mají i stroje mnoho různých jazyků. Rozdíl je však v tom, že lidské jazyky se vyvinuly přirozeně.

Navíc se stále vyvíjejí a každý den vznikají nová slova, zatímco stará zanikají. Tyto jazyky se nazývají přirozené jazyky.

1.1.1.3 Programování - absolutní základy

Co tvoří jazyk?

Můžeme říci, že každý jazyk (strojový nebo přirozený, na tom nezáleží) se skládá z následujících prvků:

- **abeceda**: soubor symbolů, z nichž se skládají slova určitého jazyka (např. latinka pro angličtinu, cyrilice pro ruštinu, kandži pro japonštinu atd.).
- **lexikum**: (neboli slovník) soubor slov, která daný jazyk nabízí svým uživatelům (např. slovo "computer" pochází ze slovníku anglického jazyka, zatímco "cmoptrue" nikoli; slovo "chat" se vyskytuje jak v anglickém, tak ve francouzském slovníku, ale jejich významy jsou odlišné).
- **syntax**: soubor pravidel (formálních nebo neformálních, psaných nebo intuitivně pociťovaných), která slouží k určení, zda určitý řetězec slov tvoří platnou větu (např. "jsem krajta" je syntakticky správná věta, zatímco "já krajta jsem" nikoli)
- **sémantika**: soubor pravidel určujících, zda určitá věta dává smysl (např. věta "Snědl jsem koblihu" dává smysl, ale věta "Kobliha mě snědla" nikoli).

IL (instruction language) je ve skutečnosti **abeceda strojového jazyka**. Je to nejjednodušší a nejzákladnější sada symbolů, kterou můžeme použít k zadávání příkazů počítači. Je to mateřský jazyk počítače.

Žádný počítač není v současné době schopen vytvořit nový jazyk. To se však může brzy změnit. Stejně jako lidé používají řadu velmi odlišných jazyků, mají i stroje mnoho různých jazyků. Rozdíl je však v tom, že lidské jazyky se vyvinuly přirozeně.

Navíc se stále vyvíjejí a každý den vznikají nová slova, zatímco stará zanikají. Tyto jazyky se nazývají **přirozené jazyky**.

1.1.1.4 Programování - absolutní základy | Kompilace vs. interpretace

Žádný počítač není v současné době schopen vytvořit nový jazyk. To se však může brzy změnit. Stejně jako lidé používají řadu velmi odlišných jazyků, mají i stroje mnoho různých jazyků. Rozdíl je však v tom, že lidské jazyky se vyvinuly přirozeně.

Počítačové programování je činnost spočívající ve skládání prvků zvoleného programovacího jazyka v pořadí, které způsobí požadovaný efekt. Efekt může být v každém konkrétním případě jiný - záleží na představivosti, znalostech a zkušenostech programátora.

Taková kompozice musí být samozřejmě v mnoha ohledech správná:

- **abecedně** - program musí být napsán v rozpoznatelném písmu, například v latince, cyrilici atd.
- **lexikálně** - každý programovací jazyk má svůj slovník a je třeba jej ovládat; naštěstí je mnohem jednodušší a menší než slovník jakéhokoli přirozeného jazyka;
- **syntakticky** - každý jazyk má svá pravidla a ta je třeba dodržovat;
- **sémanticky** - program musí dávat smysl.

Bohužel i programátor se může dopustit chyb v každém z výše uvedených čtyř smyslů. Každý z nich může způsobit, že se program stane zcela nepoužitelným.

Předpokládejme, že jste úspěšně napsali program. Jak přesvědčíme počítač, aby jej provedl? Musíte svůj program převést do strojového jazyka. Naštěstí tento překlad může provést počítač sám, takže celý proces je rychlý a efektivní.

Existují dva různé způsoby transformace programu z vysokoúrovňového programovacího jazyka do strojového jazyka:

KOMPILACE - zdrojový program je přeložen jednou (tento úkon je však nutné opakovat při každé úpravě zdrojového kódu), a to tak, že získáte soubor (např. soubor .exe, pokud je kód určen ke spuštění pod MS Windows) obsahující strojový kód; nyní můžete soubor distribuovat po celém světě; program, který tento překlad provádí, se nazývá kompilátor nebo překladač;

INTERPRETACE - vy (nebo kterýkoli uživatel kódu) můžete zdrojový program přeložit pokaždé, když má být spuštěn; program provádějící tento druh transformace se nazývá interpret, protože interpretuje kód pokaždé, když má být spuštěn; to také znamená, že nemůžete zdrojový kód distribuovat jen tak, jak je, protože koncový uživatel potřebuje k jeho spuštění také interpret.

Z některých velmi zásadních důvodů je konkrétní vysokoúrovňový programovací jazyk navržen tak, aby spadal do jedné z těchto dvou kategorií.

Existuje jen velmi málo jazyků, které lze kompilovat i interpretovat. Obvykle se programovací jazyk projektuje s ohledem na tento faktor v hlavách jeho konstruktérů - bude kompilovaný, nebo interpretovaný?

1.1.1.5 Programování - absolutní základy | Kompilace vs. Interpretace

Co vlastně interpret dělá?

Předpokládejme ještě jednou, že jste napsali program. Nyní existuje jako **počítačový soubor**: počítačový program je vlastně kus textu, takže zdrojový kód je obvykle umístěn v **textových souborech**.

Pozor: musí to být čistý text, bez jakýchkoli ozdob, jako jsou různá písma, barvy, vložené obrázky nebo jiná média. Nyní je třeba vyvolat interpret a nechat jej přečíst zdrojový soubor.

Interpret čte zdrojový kód způsobem, který je běžný v západní kultuře: shora dolů a zleva doprava. Existuje několik výjimek - budeme se jimi zabývat později v kurzu.

Nejprve interpret zkontroluje, zda jsou všechny následující řádky správné (pomocí čtyř aspektů, které jsme probrali dříve).

Pokud interpret najde chybu, okamžitě ukončí svou práci. Jediným výsledkem je v tomto případě chybové hlášení.

Interpret vás informuje, kde se chyba nachází a co ji způsobilo. Tato hlášení však mohou být zavádějící, protože interpret není schopen přesně sledovat vaše záměry a může odhalit chyby v určité vzdálenosti od jejich skutečných příčin.

Například pokud se pokusíte použít entitu neznámého jména, způsobí to chybu, ale chyba bude odhalena v místě, kde se pokouší entitu použít, nikoli tam, kde bylo zavedeno nové jméno entity.

Pokud řádek vypadá dobře, interpret se jej pokusí provést (poznámka: každý řádek se obvykle provádí samostatně, takže trojice "read-check-execute" se může opakovat mnohokrát - vícekrát, než je skutečný počet řádků ve zdrojovém souboru, protože některé části kódu mohou být provedeny vícekrát).

Je také možné, že značná část kódu může být úspěšně provedena, než interpret najde chybu. To je normální chování v tomto modelu provádění.

Nyní se můžete ptát: Co je lepší? Model "kompilace" nebo model "interpretace"? Na tuto otázku neexistuje jednoznačná odpověď. Kdyby existovala, jeden z těchto modelů by již dávno přestal existovat. Oba mají své výhody i nevýhody.

1.1.1.6 Programování - absolutní základy | Kompilace vs. Interpretace

	Kompilace	Interpretace
Výhody	• provádění přeloženého kódu je obvykle rychlejší; • překladač musí mít pouze programátor - koncový uživatel může kód používat i bez něj; • přeložený kód je uložen pomocí strojového jazyka - protože je velmi obtížné mu porozumět, vaše vlastní vynálezy a programátorské triky pravděpodobně zůstanou vaším tajemstvím.	• můžete kód spustit ihned po dokončení - neprobíhají žádné další fáze překladu; • kód je uložen v programovacím jazyce, nikoli ve strojovém - to znamená, že jej lze spustit na počítačích s různými strojovými jazyky; kód nemusíte kompilovat zvlášť pro každou jinou architekturu.
Nevýhody	• samotná kompilace může být časově velmi náročná - kód nemusí být možné spustit ihned po případné změně; • musíte mít tolik kompilátorů, na kolika hardwarových platformách chcete svůj kód spustit.	• neočekávejte, že interpretace zvýší rychlost vašeho kódu - váš kód bude sdílet výkon počítače s interpretem, takže nemůže být skutečně rychlý; • vy i koncový uživatel musíte mít interpretr, abyste mohli spustit svůj kód.

Co to pro vás znamená?

Python je interpretovaný jazyk. To znamená, že zdědil všechny popsané výhody i nevýhody. Samozřejmě k oběma množinám přidává některé své jedinečné vlastnosti.

Pokud chcete programovat v jazyce Python, budete potřebovat interpret jazyka Python. Bez něj nebudete moci svůj kód spustit. Naštěstí je Python zdarma. To je jedna z jeho nejdůležitějších výhod.

Z historických důvodů se jazykům určeným k využití interpretačním způsobem často říká **skriptovací jazyky**, zatímco zdrojovým programům, které jsou v nich zakódovány, se říká **skripty**.

1.1.2.1 Python - nástroj, ne plaz

Python je široce používaný, interpretovaný, objektově orientovaný a vysokoúrovňový programovací jazyk s dynamickou sémantikou, který se používá k programování pro obecné účely.

A i když možná znáte krajtu jako velkého hada, název programovacího jazyka Python pochází ze starého komediálního seriálu televizní stanice BBC s názvem Monty Pythonův létající cirkus.

Na vrcholu svého úspěchu předváděl tým Monty Python své skeče živému publiku po celém světě, včetně Hollywood Bowl.

Protože Monty Python je považován za jednu ze dvou základních živin programátora (tou druhou je pizza), pojmenoval tvůrce Pythonu jazyk na počest televizního pořadu.

Kdo vytvořil Python?

Jednou z úžasných vlastností Pythonu je skutečnost, že je vlastně dílem jednoho člověka. Obvykle nové programovací jazyky vyvíjejí a publikují velké společnosti zaměstnávající spoustu odborníků a kvůli pravidlům autorského práva je velmi obtížné jmenovat některého z lidí, kteří se na projektu podíleli. Python je výjimkou.

Není mnoho jazyků, jejichž autoři jsou známí jménem. Python vytvořil Guido van Rossum, který se narodil v roce 1956 v nizozemském Haarlemu. Guido van Rossum samozřejmě nevyvinul a nevyvinul všechny komponenty Pythonu sám.

Rychlost, s jakou se Python rozšířil po celém světě, je výsledkem nepřetržité práce tisíců (velmi často anonymních) programátorů, testerů, uživatelů (mnozí z nich nejsou IT specialisté) a nadšenců, ale je třeba říci, že úplně první nápad (semínko, z něhož Python vyklíčil) se zrodil v jedné hlavě - Guidově.

1.1.2.2 Python - nástroj, ne plaz

Projekt hobby programování

Okolnosti vzniku Pythonu jsou poněkud záhadné. Podle Guida van Rossuma:

V prosinci 1989 jsem hledal "hobby" programátorský projekt, který by mě zaměstnal během vánočního týdne. Moje kancelář (...) by byla zavřená, ale měl jsem k dispozici domácí počítač a nic jiného mi nezbývalo. Rozhodl jsem se napsat interpret pro nový skriptovací jazyk, o kterém jsem v poslední době přemýšlel: potomka jazyka ABC, který by se líbil hackerům Unixu/C. V té době jsem se rozhodl napsat interpreter pro nový skriptovací jazyk, o kterém jsem v poslední době přemýšlel. Jako pracovní název projektu jsem zvolil Python, protože jsem měl lehce neuctivou náladu (a byl jsem velkým fanouškem Monty Pythonova létajícího cirkusu).
Guido van Rossum

Cíle Pythonu

V roce 1999 Guido van Rossum definoval své cíle pro Python:

- Snadný a intuitivní jazyk, který je stejně výkonný jako jazyky hlavních konkurentů;
- otevřený zdrojový kód, aby se na jeho vývoji mohl podílet každý;
- kód srozumitelný jako běžná angličtina;
- vhodný pro každodenní úkoly a umožňující krátkou dobu vývoje.

Zhruba o 20 let později je zřejmé, že všechny tyto záměry byly naplněny. Některé zdroje uvádějí, že Python je nejoblíbenějším programovacím jazykem na světě, jiné tvrdí, že je druhý nebo třetí.

John Cleese a jeho šílená chůze (Monty Python's Flying Circus).

Každopádně se stále drží v první desítce žebříčků PYPL PopularitY of Programming Language a TIOBE Programming Community Index.

Python už není mladý jazyk. Je vyspělý a důvěryhodný. Není to zázrak jednoho hitu. Je to jasná hvězda na programátorském nebi a čas strávený učením jazyka Python je velmi dobrou investicí.

1.1.2.3 Python - nástroj, ne plaz | Proč Python?

Čím je Python výjimečný?

Jak se stalo, že ho chtějí používat mladí i staří, zkušení i začínající programátoři? Jak se stalo, že si Python osvojily velké společnosti a implementovaly v něm své stěžejní produkty?

Důvodů je mnoho - některé z nich jsme již uvedli, ale pojďme si je vyjmenovat ještě jednou a praktičtěji:

- je snadné se ho naučit - doba potřebná k naučení Pythonu je kratší než u mnoha jiných jazyků; to znamená, že je možné začít rychleji programovat;
- je snadný na výuku - náročnost výuky je menší než u jiných jazyků; to znamená, že učitel může klást větší důraz na obecné (na jazyce nezávislé) techniky programování a neplýtvat energií na exotické triky, podivné výjimky a nesrozumitelná pravidla;
- je snadno použitelný pro psaní nového softwaru - při použití jazyka Python je často možné psát kód rychleji;
- je snadno pochopitelný - často je také snazší rychleji pochopit cizí kód, pokud je napsán v jazyce Python;
- je snadné jej získat, nainstalovat a nasadit - Python je zdarma, otevřený a multiplatformní; ne všechny jazyky se tím mohou pochlubit.

Python má samozřejmě i své nevýhody:

- není to rychlostní démon - Python nepřináší výjimečný výkon;
- v některých případech může být odolný vůči některým jednodušším testovacím technikám - to může znamenat, že ladění kódu Pythonu může být obtížnější než u jiných jazyků; naštěstí v Pythonu je vždy těžší udělat chybu.

Je třeba také uvést, že Python není jediným řešením svého druhu dostupným na trhu IT.

Má spoustu příznivců, ale je mnoho těch, kteří dávají přednost jiným jazykům a o Pythonu pro své projekty ani neuvažují.

1.1.2.4 Python - nástroj, ne plaz | Proč Python, proč ne?

Soupeři Pythonu?

Python má dva přímé konkurenty se srovnatelnými vlastnostmi a předpoklady. Jsou to:

- **Perl** - skriptovací jazyk, jehož autorem je Larry Wall
- **Ruby** - skriptovací jazyk, jehož autorem je Yukihiro Matsumoto

První z nich je tradičnější, konzervativnější než Python a připomíná některé staré dobré jazyky odvozené z klasického programovacího jazyka C. Druhý jmenovaný jazyk je tradičnější a konzervativnější než Python.

Naproti tomu druhý jmenovaný je inovativnější a plný svěžích nápadů než Python. Samotný Python leží někde mezi těmito dvěma výtvory.

Pokud se chcete o každém z těchto tří jazyků dozvědět více, internet je plný fór s nekonečnými diskusemi o nadřazenosti jednoho z nich nad ostatními.

Kde můžeme Python vidět v akci?

Setkáváme se s ním každý den a téměř všude. Hojně se používá k **implementaci složitých internetových služeb, jako jsou vyhledávače, cloudová úložiště a nástroje, sociální sítě atd**. Kdykoli některou z těchto služeb používáte, máte k Pythonu vlastně velmi blízko, i když byste to nevěděli.

V Pythonu je implementováno mnoho vývojových nástrojů. V Pythonu se píše stále více **aplikací pro každodenní použití**. **Spousta vědců opustila drahé proprietární nástroje a přešla na Python**. **Spousta testerů IT projektů začala používat Python k provádění opakovatelných testovacích postupů**. Seznam je dlouhý.

Proč ne Python?

Navzdory rostoucí popularitě Pythonu stále existují některé oblasti, kde se Python nevyskytuje nebo se s ním setkáváme jen zřídka:

- Pokud chcete implementovat extrémně efektivní ovladač nebo grafický engine, Python nepoužijete
- aplikace pro mobilní zařízení: ačkoli toto území stále čeká na dobytí Pythonem, jednou se tak pravděpodobně stane.

1.1.3.1 Python 2 vs. Python 3

Existuje více než jeden jazyk Python

Existují dva hlavní druhy jazyka Python, nazývané Python 2 a Python 3.

Python 2 je starší verze původního Pythonu. Jeho vývoj byl od té doby záměrně zastaven, i když to neznamená, že by neexistovaly jeho aktualizace. Naopak, aktualizace jsou vydávány pravidelně, ale nemají za cíl jazyk nijak výrazně upravovat. Spíše opravují všechny čerstvě objevené chyby a bezpečnostní díry. Cesta vývoje jazyka Python 2 se již dostala do slepé uličky, ale samotný Python 2 je stále velmi živý.

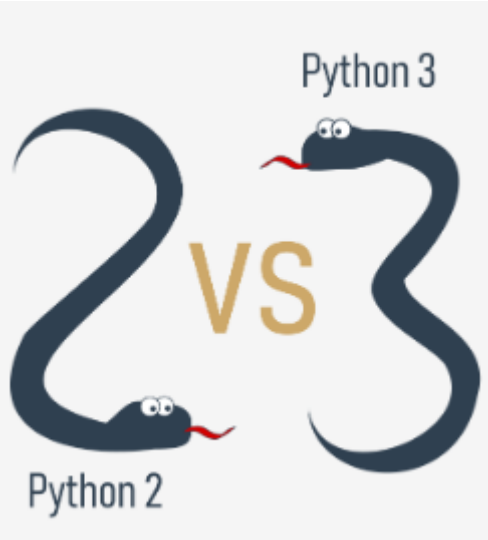
Python 3 je novější (přesněji řečeno aktuální) verze jazyka. Prochází vlastní cestou vývoje, vytváří si vlastní standardy a zvyklosti.

Tyto dvě verze jazyka Python nejsou vzájemně kompatibilní. Skripty Pythonu 2 nepoběží v prostředí Pythonu 3 a naopak, takže pokud chcete, aby starý kód Pythonu 2 běžel v interpretu Pythonu 3, jediným možným řešením je jeho přepsání, samozřejmě ne od základu, protože velké části kódu mohou zůstat nedotčeny, ale musíte celý kód přepracovat a najít všechny možné nekompatibility. Tento proces bohužel nelze plně automatizovat.

Migrace staré aplikace v jazyce Python 2 na novou platformu je příliš náročná, zdlouhavá, drahá a riskantní. Je možné, že přepsání kódu do něj vnese nové chyby. Je jednodušší a rozumnější nechat tyto systémy na pokoji a vylepšit stávající interpret, než se snažit pracovat uvnitř již fungujícího zdrojového kódu.

Python 3 není jen lepší verzí Pythonu 2 - je to úplně jiný jazyk, i když je svému předchůdci velmi podobný. Při pohledu z dálky se zdá, že jsou stejné, ale když se podíváte blíže, všimnete si mnoha rozdílů.

Pokud upravujete staré existující řešení v jazyce Python, je velmi pravděpodobné, že bylo kódováno v jazyce Python 2. To je důvod, proč se Python 2 stále používá. Existuje příliš mnoho existujících aplikací v Pythonu 2 na to, aby byl zcela zavržen.



Poznámka

Pokud se chystáte začít nový projekt v jazyce Python, měli byste použít Python 3, což je verze jazyka Python, která bude použita v tomto kurzu.

Je důležité si uvědomit, že mezi následujícími verzemi Pythonu 3 mohou být menší či větší rozdíly (např. v Pythonu 3.6 byly v rámci implementace CPython zavedeny uspořádané klíče slovníků ve výchozím nastavení) - dobrou zprávou však je, že všechny novější verze Pythonu 3 jsou zpětně kompatibilní s předchozími verzemi Pythonu 3. Pokud to bude smysluplné a důležité, budeme se vždy snažit na tyto rozdíly v kurzu upozornit.

Všechny ukázky kódu, které v průběhu kurzu najdete, byly testovány proti verzím Python 3.4, Python 3.6, Python 3.7 a Python 3.8.

1.1.3.2 Existuje více než jeden Python: CPython a Cython

Python alias CPython

Kromě Pythonu 2 a Pythonu 3 existuje více než jedna verze každého z nich.

Logo Python Software Foundation

Především jsou tu Pythony, o které se starají lidé sdruženi kolem PSF (Python Software Foundation), komunity, jejímž cílem je vyvíjet, zlepšovat, rozšiřovat a popularizovat Python a jeho prostředí. Prezidentem PSF je sám Guido von Rossum, a proto se těmto Pythonům říká **kanonické**. Jsou také

považovány za **referenční Pythony**, protože jakákoli jiná implementace jazyka by měla dodržovat všechny standardy stanovené PSF.

Guido van Rossum použil programovací jazyk "C" pro implementaci první verze svého jazyka a toto rozhodnutí platí dodnes. Všechny Pythony pocházející z PSF jsou napsány v jazyce "C". Pro tento přístup existuje mnoho důvodů a má mnoho důsledků. Jedním z nich (pravděpodobně tím nejdůležitějším) je, že díky němu lze Python snadno přenášet a migrovat na všechny platformy s možností kompilovat a spouštět programy v jazyce "C" (tuto vlastnost mají prakticky všechny platformy, což Pythonu otevírá mnoho možností rozšíření).

Proto se implementace PSF často označuje jako CPython. Jedná se o nejvlivnější Python mezi všemi Pythony na světě.

Cython

Dalším členem rodiny Pythonů je Cython.

Cython je jedním z možných řešení nejbolavějšího rysu jazyka Python - nedostatečné efektivity. V Pythonu lze snadno nakódovat rozsáhlé a složité matematické výpočty (mnohem snadněji než v "céčku" nebo jiném tradičním jazyce), ale provedení výsledného kódu může být časově velmi náročné.

Jak lze tyto dva rozpory sladit? Jedním z řešení je psát své matematické myšlenky v jazyce Python, a když jste si naprosto jisti, že váš kód je správný a dává platné výsledky, můžete jej přeložit do jazyka "C". Je jisté, že "C" poběží mnohem rychleji než čistý Python.

Právě k tomu je Cython určen - k automatickému překladu kódu v jazyce Python (čistého a přehledného, ale ne příliš svižného) do kódu v jazyce "C" (složitého a upovídaného, ale svižného).

1.1.3.3 Existuje více než jeden Python: Jython, PyPy a RPython.

Další verze jazyka Python se nazývá Jython.

"J" znamená "Java". Představte si Python napsaný v jazyce Java místo v jazyce C. To je užitečné například v případě, že vyvíjíte rozsáhlé a složité systémy napsané výhradně v jazyce Java a chcete do nich přidat trochu flexibility Pythonu. Tradiční CPython může být obtížné do takového prostředí integrovat, protože C a Java žijí ve zcela odlišných světech a nesdílejí mnoho společných myšlenek.

Jython může efektivněji komunikovat se stávající infrastrukturou Javy. Proto jej některé projekty považují za použitelný a potřebný.

Poznámka:

současná implementace Jythonu se řídí standardy Pythonu 2. Jython odpovídající Pythonu 3 zatím neexistuje.

PyPy a Rpython

Podívejte se na logo níže. Je to rebus. Dokážete ho vyřešit?

Je to logo PyPy - Python v Pythonu. Jinými slovy představuje prostředí Pythonu napsané v jazyce podobném Pythonu s názvem RPython (Restricted Python). Jedná se vlastně o podmnožinu jazyka Python.

Zdrojový kód PyPy se nespouští interpretačním způsobem, ale překládá se do programovacího jazyka C a poté se spouští samostatně.

To je užitečné, protože pokud chcete otestovat nějakou novou funkci, která může být (ale nemusí) zavedena do běžné implementace jazyka Python, je snazší ji ověřit pomocí PyPy než pomocí CPythonu. Proto je PyPy spíše nástrojem pro lidi, kteří Python vyvíjejí, než pro ostatní uživatele.

To samozřejmě neznamená, že by PyPy bylo méně důležité nebo méně seriózní než CPython.

Kromě toho je PyPy kompatibilní s jazykem Python 3.

Různých Pythonů je na světě mnohem více. Pokud budete hledat, najdete je, ale tento kurz se zaměří na CPython.

Odkazy:

Cisco Programming Essentials in Python

Root.cz

ITNetwork.cz

Internet