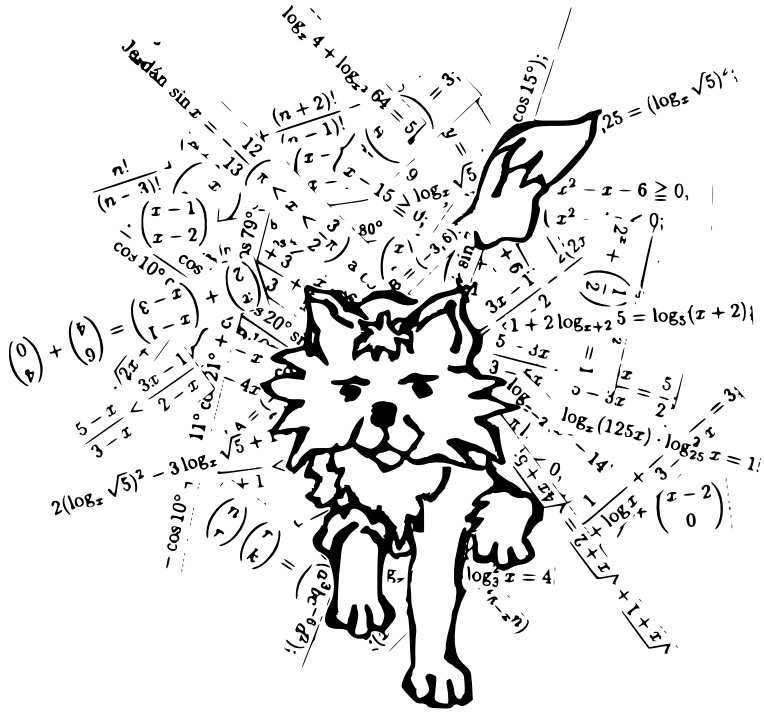
Studentský matematicko-fyzikální časopis

M&M

Ročník XVII / Číslo 1



Milá kamarádko, milý kamaráde!

Zajímáš se o matematiku, fyziku či informatiku? Baví tě zamýšlet se nad zajímavými problémy a úlohami? Chceš se naučit napsat vědecký článek? Chceš jet dvakrát za rok na týdenní soustředění a poznat tam nové kamarády? Pak je tu pro tebe časopis M&M a stejnojmenný korespondenční seminář.

Časopis M&M je určen pro studenty středních škol. Během školního roku řešitelé dostávají zdarma čísla časopisu se zadáním úloh a témat k přemýšlení. Svá řešení posílají do redakce, kde jejich příspěvky opravíme, obodujeme a zašleme zpět. Nejzajímavější z nich uveřejníme.

Jak začít řešit? Přečti si následující úlohy a témata. Zkus se nad nimi zamyslet a pošli nám svá řešení a nápady. Více informací o tom, jak se zapojit najdeš na straně 14.

Termín odeslání první série: 18. 10. 2010
(27. 9. 2010 pro účast na soustředění)

Zadání úloh

Úloha 1.1 – Pokažená hra (4b)

Rád vzpomínám na své klukovské roky, kdy jsme si u nás za domem hrávali na malém čtvercovém plácku. Nejvíce mi v paměti utkvělo jedno pozdní sobotní odpoledne. Sešli jsme se tehdy v plném počtu šestnácti kluků. Není také divu, když jsme se odplatou měli utkat s Podkováky za to, že nám zničili jednu z našich her.

Jako první se k něčemu odvážil nejstarší Mirek, který stál spolu s Jiříkem uprostřed. Udělal dva kroky dopředu a z plna hrdla se rozkřičel na Podkováky, ať táhnou domů. Podkováci si to samozřejmě nenechali líbit a z řad Podkováků vyrazil Lojza, jenž taktéž provedl dva kroky vpřed. Postavil se tak, že stál šikmo od Mirka a hned mu to začal oplácet. A tak se do klukovské šarvátky začali postupně přidávat další a další - náš Jiřík, který stál naproti Lojzovi, popošel o dva kroky, takže měl Mirka po své pravé ruce. Ondra od Podokováků se posunul do takové pozice, aby stál v jedné přímce s Lojzou a Mirkem ...

A o co tehdy vlastně šlo? Jak se dá u klukovských svárů očekávat, byl důvod zcela malicherný.

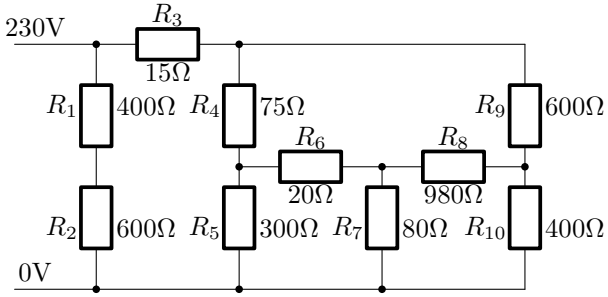
Na náš plácek jsme tehdy rozmístili papírky s čísly 1, 2, 3, ... 64 do mřížky 8×8 tím způsobem, že v prvním řádku byla postupně čísla 1 až 8, v druhém 9 až 16 atd. Papírky na plácku zůstaly přes noc a Podkováci nám před některá čísla napsali mínus, a to tak zvláštně, že v každém řádku a v každém sloupečku byla právě čtyři záporná čísla. A proč to udělali? Prý chtěli vědět, jaký potom bude celkový součet všech takto upravených čísel naší hry. Kolik jim mohlo vyjít?

Úloha 1.2 – Osvětlení (4b)

Situace na plácku se postupně vyostřovala. Do souboje, který zatím probíhal jen formou nadávek, se přidal i náš Karel. Nejdříve nesměle přešlapoval na místě, ale pak udělal úkrok doprava a dvěma mohutnými skoky se objevil nedaleko Jiříka. Když nadávky přestávaly stačit, Lojza nevydržel a vrhnul se na Mirka. Mezi rvoucí se kluky skočil i Karel, ale bylo již příliš pozdě - Mirek odcházel s brekem a rozbitým nosem, což Karel Lojzovi velmi rychle a tvrdě oplatil. Nato vyrazil Tomáš od Podkováků a podobně jako prve Karel, ukročil stranou a pak učinil dva skoky. Tomáš i Karel se nyní dívali takřka z očí do očí připraveni jeden na druhého ve vteřině skočit.

Večer mírně pokročil, na plácek se snášela tma a nad hlavami kluků se začaly rozsvěcet žárovky.

Nad pláckem byla natažena síť nejrůznějších světýlek, která měla různý elektrický odpor. Jednou jsme je pro zábavu proměřili a zakreslili do následujícího schématu[1]. Když jsem se po letech na toto schéma opět díval, všiml jsem si jedné velice zvláštně zapojené žárovky. Pokud by se totiž vymontovala, nic by se v okolním zapojení nezměnilo – tedy nezměnil by se proud protékající ostatními žárovkami. Poznáte, o jakou žárovku šlo?



Úloha 1.3 – Podkovácká čtvrť (3b)

Tehdy jsem se poprvé do hry zapojil i já. Udělal jsem dva kroky a postavil se za Jiříka. Natáhl jsem svůj prak a mířil přes hlavu Karla. Útok od Podkováků však přišel odjinud. Na Jiříka se vyřítila Julča, jediná slečna mezi Podkováky. Na holku se uměla rvát překvapivě dobře. A tak Jiřík s roztrženou košilí a boulí na nose utíkal domů. To mi připomnělo, jak vlastně rivalita mezi námi a Podkováky začala.

Způsobil to Jiřík, který se jednou zatoulal do jejich městské čtvrti. Podkovácká čtvrť byla tvořena čtvercovou sítí náměstíček $n \times n$, která byla spojena pravoúhlou sítí ulic (vnitřní náměstíčko bylo ulicemi propojeno právě se čtyřmi vedlejšími, krajní s okolními třemi a rohové se dvěma náměstíčky). Kolem čtvrti byly vysoké ploty továren a hlavní silnice, kudy se nedalo rychle utéct.

[1] Ve schématu jsou žárovky znázorněny značkami pro rezistory a označeny jako $R_1, \ldots, R_{10}$.

Vidím to jako dnes. Jednotliví Podkováci byli tehdy velmi dobře rozmístěni. Jiřík ještě stihl doběhnout na náměstíčko, které mu přišlo bezpečné. Pozorně se zaposlouchal aby uhodl, kde se jeho protivníci nacházejí. Pak hon začal. Jiřík se rozeběhl na jedno z vedlejších náměstíček. Jen co tam doběhl, všimlo si ho nějaké dítě a běželo všem větším klukům říci, kam se Jiřík přesunul. Každý z Podkováků buď přeběhl na některé ze sousedních náměstíček, nebo vyčkával na místě. Když dusot Podkováků ustal, Jiřík se opět zaposlouchal a pak běžel zas na jedno ze sousedních náměstíček nebo vyčkával. A tak to probíhalo stále dokola.

Jiříka tehdy pořádně prohnali. Pokud by se Jiřík ve kteroukoliv chvíli ocitl na jednom z náměstíček spolu s některým z Podkováků, dostal by výprask. Podkováci ale naštěstí nejsou zas tak vytrvalí běžci a pokud by Jiřík Podkovákům unikal dostatečně dlouho[2], přestalo by je to bavit.

Zjistěte, kolik Podkováků je potřeba k polapení Jiříka a kolik ještě nestačí. Oba počty podpořte nějakým důkazem, například strategií k Podkováků, která Jiříka vždy chytí, nebo Jiříkovou strategií, se kterou bude l Podkovákům unikat libovolně dlouho. Strategie musí zahrnovat počáteční rozestavení i vhodný tah pro každou situaci.

Část bodů dostanete i za neoptimální počty, ale věžte, že Podkováků stačí mnohem méně než n. Je lépe dokázat slabší výsledek, než uhodnout optimum.

Úloha 1.4 – Šachová úloha (2b)

Čím dál více se šeřilo a souboj s Podkováky začínal být dramatičtější. Potom, co Julča poslala domů Jiříka, se do středu dění vyřítil Pepa. Podobně jako Karel, nejdříve ukročil stranou, a pak dvěma kroky přeskočil před sebou stojícího Frantu. Postavil se tak, že mu stačil jeden skok a Julča by se dostala do jeho spárů. Při pohledu na vysokého Pepu se Julča raději vrátila na své místo, čekajíc, co se bude dít. To ocenila naše Lucka, jež udělala krok k Pepovi a poplácala jej po zádech.

Z Pepy měli Podkováci respekt. Dokonce takový respekt, že Tomáš raději vyklidil pole a šel se postavit před Julču, aby ji mohl chránit.

Čím si Pepa zasloužil takový respekt u Podkováků? Jednou se mu totiž podařilo Podkováky přechytračit.

Pepa se s Podkováky vsadil, že jim zadá šachovou úlohu, kterou nevyřeší. Zeptal se jich, kolik nejvíce dam můžou postavit na šachovnici tak, aby právě dvě políčka zůstala dámami neohrožena. Dámy se navzájem ohrožovat můžou. Podkováci na to tehdy vůbec nepřišli. A co vy, víte jak na to?

Úloha 1.5 – A jak to bylo dál? (2b)

Jak si myslíte, že příběh skončil? Zkuste nám poslat své dokončení příběhu.

[2] Podkováci určitě neproběhnou více než 2^n ulic.

Zadání témat

Co to je téma?

Specialitou našeho semináře jsou témata. Vlastními silami v nich prozkoumáš fyzikální zákonitosti, objevíš matematické vztahy nebo napíšeš program. Každé z nich začíná úvodní úlohou, která je formulována poměrně široce a má obvykle několik částí. Zadání by mělo být především námětem k přemýšlení. Můžeš nám poslat jak řešení některé části úvodní úlohy, tak řešení dalších problémů, které si v rámci tématu sám vymyslíš. Pokud se nám bude tvůj článek líbit, uveřejníme jej v některém z dalších čísel. Článek k tématu můžeš zaslat kdykoli během roku. Počet tvých článků k jednomu tématu není nijak omezen – své úvahy můžeš dále rozvíjet, doplňovat, případně poopravit nebo úplně vyvrátit. Můžeš též reagovat na články svých kolegů nebo využít jejich výsledky ve svém dalším řešení.

Za kvalitní otištěný článek lze získat až 20 bodů – hodnotíme nejen správnost, ale i dobrý nápad a snahu téma rozvinout. Důležitá je i forma tvého vědeckého článku.

Do redakce můžeš poslat i vlastní námět na nové téma týkající se matematiky, fyziky nebo informatiky. Pokud se nám bude zdát zajímavý, uveřejníme ho při nejbližší vhodné příležitosti a tebe bodově ohodnotíme.

Téma 1 – Rekurentní posloupnosti

Posloupnost[3] můžeme zadat mnoha způsoby. V tomto tématu budeme zkoumat především situaci, kdy je posloupnost zadaná rekurentně, tedy n–tý člen je definován pomocí několika předchozích členů. Například posloupnost 1, 2, 3, ... můžeme zadat pomocí podmínky $a_1 = 1$ a rekurentního vztahu $a_{n+1} = a_n + 1$.

Jednou z nejznámějších takových posloupností je Fibonacciho posloupnost[4], která je definovaná pomocí rekurentní podmínky $F_{n+2} = F_{n+1} + Fn$ a počátečních hodnot $F_0 = 0$ a $F_1 = 1$. Její členy tedy vypadají 0, 1, 1, 2, 3, 5, 8, 13, 21, ... Touto posloupností se zabývala celá řada matematiků a proto už je prozkoumaná opravdu dobře. Je znám například explicitní vzorec pro n-tý člen a spousta zajímavých (mnohdy až neuvěřitelných) identit, například:

$$\sum_{i=0}^{n} F_i = F_{n+2} - 1\,, \sum_{i=0}^{n} {F_i}^2 = F_n F_{n+1}\,,$$

[3] Pokud nevíš, co je to posloupnost, tak si pod tímto pojmem můžeš představit uspořádanou řadu čísel. Případně se podívej na `http://cs.wikipedia.org/wiki/Posloupnost`.

[4] Více informací o Fibonacciho posloupnosti nalezneš například na internetu na adresách: `http://cube.mysteria.cz/fibonacci.ppt`, `http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/`, `http://en.wikipedia.org/wiki/Fibonacci_number` a na adrese `http://mathworld.wolfram.coms/FibonacciNumber.html`.

$$\sum_{1=0}^{\infty} \frac{F_i}{10^{i+1}} = \frac{1}{89},$$

$$F_{3n} = F_{n+1}^3 + F_n^3 - F_{n-1}^3$$

a mnohé další.

Vaším úkolem v tomto tématu bude zkoumat různé rekurentní posloupnosti a hledat všelijaké jejich vlastnosti. Zkuste se například zamyslet nad posloupností definovanou vztahem $A_{n+2} = 2A_{n+1} + 2An$ s počátečními hodnotami $A_0 = 0$ a $A_1 = 1$. Dokážete nějak shora a zdola odhadnout n-tý člen? A co pro něj odvodit přímo přesný vzorec? Případně nebude pro členy této posloupnosti platit nějaký zajímavý vztah? Fantazii se meze nekladou. A našla by se nějaká souvislost s Fibonacciho posloupností? Pokud chcete nějakou zajímavější posloupnost, můžete se pokusit na stejné otázky odpovědět třeba pro $B_{n+3} = 4B_n - 8B_{n+1} + 5B_{n+2}$ s prvními členy $B_0 = 0$, $B_1 = 1$ a $B_2 = 2$. Bude tato posloupnost vůbec rostoucí?

Případně se můžete pokusit o opačný postup. Zkuste najít například takovou rekurentní posloupnost, jejíž všechny členy budou mezi 0 a 1. A zjistit o ní co nejvíc. Nebo zkuste najít posloupnost, která bude mít konečný součet všech svých členů a každý třetí člen bude záporný. Existuje posloupnost pouze s kladnými členy s konečným součtem?

Pokud narazíte na libovolnou pěknou rekurentní posloupnost, tak se o ni nebojte s ostatními podělit.

Téma 2 – Mapování

Mapy mohou mít mnoho podob. Z některých se dozvíme, jak se dostat z místa A do místa B, jiné nám popisují, kde v České republice je jaké geologické podloží. Vzhledem k tomu, že bydlíme na různých místech naší země, bylo by zajímavé, kdybychom se pokusili provést měření některých fyzikálních veličin a sestavit pro ně mapy.

Co má smysl měřit? Veličiny, které se v rámci ČR mění. Rozhodně nemá smysl měřit fundamentální fyzikální konstanty, jako je rychlost světla, ta by, jak všichni věříme `:-)`, měla být všude stejná. Někdo by možná mohl navrhnout vytvořit mapu místního tíhového zrychlení, která opravdu existuje. Ovšem naše vybavení by jen stěží umožnilo provést měření, které by mělo menší chybu, než je hledaný rozdíl.

Nabízí se proto provést meteorologická měření. Tato měření závisí nejen na poloze, ale době, kdy bylo měření provedeno. Proto je potřeba, abychom měření synchronizovali. Mohli bychom začít tím nejjednodušším, což je měření teploty. První hromadné měření vyhlašujeme na **středu 22. září 2010 ve 20:00 hodin**. V tento čas zkuste vystrčit ven teploměr a naměřit hodnotu. Tu pak zadejte do webového formuláře na našich stránkách. Podrobnosti naleznete na `http://mam.mff.cuni.cz/mapovani`[5].

[5] Stránka bude zprovozněna během prázdnin.

Měření se pokuste provádět tak, aby bylo co nejméně ovlivněno okolními vlivy. Pokud teploměr položíte na rozpálený parapet okna, tak budete měřit spíše teplotu parapetu, než teplotu okolního vzduchu. Nejlepších výsledků měření docílíte někde v přírodě, kde kolem sebe nemáte sálající asfaltové cesty atd. Při měření teploměrem nedržte teploměr v ruce.

Jaké další veličiny můžeme měřit? Můžeme měřit tlak, srážky, intenzitu slunečního svitu, rychlost větru, směr větru, ale třeba také veličiny, které se nemění s časem, jako je nadmořská výška vašeho bydliště. Měření těchto veličin je ovšem složitější, proto zkuste navrhnout postup experimentů s pomocí běžně dostupných věcí, který by mohli ostatní řešitelé provést. Jak asi tušíte, ne všichni mají doma tlakoměr...

Co s naměřenými daty? Ve fyzice se někdy stane, že u některých závislostí neznáme přesný vztah, obvykle u materiálových konstant. Vzoreček je příliš složitý, nebo ani není možné jej analyticky vyjádřit. Pak je nutné naměřená data fitovat[6] nějakým vhodným předpisem, třeba polynomem. Bude se vám k tomu hodit program Gnuplot. Návod k němu naleznete v čísle 4. XVI. ročníku. Fitovat funkci dvou proměnných (zeměpisné šířky a výšky) není snadné. Proto můžete začít s tím, že si vezmete nějaký řez ČR, půjdete třeba po poledníku, např. budete zkoumat závislost teploty na zeměpisné šířce nebo na nadmořské výšce, nebo na něčem co vám připadá zajímavé, klidně na hustotě osídlení.

Také je dobré se zamyslet nad tím, jak data zobrazovat. Obvykle se velikost veličiny znázorňuje barevně (podívejte se na různé mapy na `http://www.chmi.cz`). Ovšem vzhledem k relativně řídké síti měřících bodů bychom měli mapu s několika puntíky. Proto je vhodné si ji rozřezat na mřížku. Vzniknou nám tak políčka, která budou mít hodnotu, která se získá zprůměrováním naměřených veličin. Jak velká políčka bychom měli zvolit?

[6] Fitování je proces, kdy hledáte koeficienty nějaké funkce, např. $y = ax^3 + bx^2 + cx + d$, tak aby funkce co nejlépe seděla s naměřenými daty.

Seriál o Pythonu I. díl

Po loňské sérii užitečných programů vám letos v seriálu představíme programovací jazyk Python[7]. Krom toho, že je velmi jednoduchý a přímočarý, má spoustu dalších výhod: univerzalitu (používá se jak na programování webových stránek, aplikací, her, vědeckých výpočtů a databází, tak pro jednoduché skriptíky či jako kalkulačka), rozšířenost (používá ho hodně firem a organizací včetně NASA, je běžnou součástí Linuxu) a multiplatformnost (rozjedete ho na Linuxu, Windows i MacOSu). Dobrou zprávou je taky to, že je „free“ (zdarma včetně zdrojových kódů).

Seriál se zpočátku zaměří na úvod do Pythonu, ale později ukážeme i jeho jednoduché a užitečné aplikace. Seriál se snažíme psát tak, aby byl pochopitelný i pro neprogramátory, ale samozřejmě hodně pomůže, pokud máte představu, co je to program, nebo jste duší matfyzáci `;-)`

Seriál je velmi zhuštěný, a proto se nenechte odradit, pokud nebudete něčemu rozumět ani napodruhé. Snažili jsme se být co nejúplnější, co se vlastností jazyka týče. Některé části (zvláště ke konci) jsou pro začátek zbytečně podrobné, ale připadalo nám o něco rozumnější říci toho raději více a vysvětlit i okrajové případy, než vás na ně nechat narážet.

Pokud vám něco nebude jasné, napište nám a my vám zkusíme poradit.

Jak se k Pythonu dostat

Popisujeme dnes rozšířenější verzi Pythonu 2.x, verze 3.x se liší v několika drobnostech. Pokud používáte Linux, máte Python 2.x takřka určitě už nainstalovaný, stačí ho spustit z terminálu příkazem `python`. Ve Windows si ho budete muset nainstalovat. Podrobnosti a doporučení pro Windows i Linux najdete na adrese `http://mam.mff.cuni.cz/python/`.

Python jako kalkulačka

Když spustíte samotný Python, bude hned reagovat na každý váš příkaz. První, co si můžete vyzkoušet, je napsat aritmetický výraz jako například `(1+2*3) * 6`, a po stisknutí Enter vám Python rovnou zobrazí výsledek. Aritmetické operátory jsou `+`, `-`, `*`, `/`, `**` (mocnění), `%` (zbytek po dělení) a mají běžnou prioritu, ale platí známé „jsi-li na pochybách, závorkuj“. *Na umístění a množství mezer ve výrazech nezáleží.*

Python rozlišuje celá (typ `int`) a reálná (typ `float`) čísla. Reálná čísla se vyznačují tím, že mají desetinou tečku. Python počítá celočíselně, dokud to jde, ale jakmile by měl provést operaci s různými typy, přejde na ten obecnější (`float`). To se projeví například při dělení, které je, pokud možno, celočíselné: `9/2` vyjde `4`, ale `9.0/2` vyjde `4.5`.

Velká a malá reálná čísla lze psát s (desítkovým) exponentem: `3.5e-2 == 0.035` a `0.2e3 == 200`. Python umí i čísla komplexní (typ `complex`, např.

[7] česky hroznýš

`1+2j`). Celá čísla mohou být libovolně velká (zkuste `2**1024`), reálná jdou až do 10^{308} a přesná jsou na asi 15 cifer.

Už v základní knihovně jsou např. funkce `abs(x)` (absolutní hodnota), `min(a,b,...)` a `max(a,b,...)`, `round(x, poč.míst)` (zaokrouhluje na daný počet des. míst a vrací `float`) a další v modulu `math`.

Jména typů jsou zároveň funkce převádějící na daný typ, např. `int(2**5.4) == 42`. `int(x)` zaokrouhluje vždy dolů.

Modul je v podstatě sada funkcí, otevřete ji příkazem `import` *modul*, např. `import math`. Funkce z modulu se pak používají se jménem modulu, např. `math.sin(x)`[8] či `math.log(x, základ)`.

Mnoho funkcí má buď volitelný počet parametrů (`min` a `max`) nebo mají některé jejich parametry přednastavené hodnoty, pokud je neuvedete (`round(4.2) == 4.0` a `math.log(math.e) == 1.0`). Pozor na to, že Python rozlišuje velikosti písmen jak u jmen funkcí, tak u proměnných.

Proměnné a přiřazení

Během běhu programu si můžete výsledky ukládat do proměnných běžným přiřazením tvaru "proměnná = výraz", které v Pythonu funguje jako definice a přepisuje předchozí hodnotu (nejde tedy o řešení rovnic). Jednotlivé příkazy (mezi které přiřazení patří) se oddělují koncem řádku, případně středníkem.

Jméno proměnné se musí skládat z písmen, číslic a podtržítka `'_'` a nesmí začínat číslem. Též se je potřeba vyhnout klíčovým slovům Pythonu *(keywords)*, jako `and, or, not, while, for, if, else, as, with, break, def` a několika dalším.

Například můžete napsat: `a=1; b=0` (nový řádek) `a=42`. Proměnná `a` pak bude mít hodnotu 42, vypsat jí můžete napsáním `a` nebo jí můžete použít v libovolném složitějším výrazu (`a*3600*24`).

Co se ale stane, pokud napíšete `a=0; b=a; a=42`? V Pythonu jsou proměnné pouhá pojmenování *objektů*, `a` a `b` proto chvíli budou jména pro objekt `0`, ale nebudou k sobě mít žádný další vztah a po třetím přiřazení se `b` nezmění (protože objekt `0` nikdo nezměnil).

Pokud dvě proměnné `a` a `b` obsahují na *tentýž měnitelný* objekt (jako například *seznam* níže), pak se změna tohoto objektu (např. zkrácení seznamu) projeví v obou proměnných.

Např. po `a=[1,2]; b=a; a[0]=0;` bude `b==[0,2]`. Po `a=[1,2]; b=a; a=[4,5]` bude stále `b==[1,2]`, protože přiřazení do `a` nezmění seznam uvnitř, ale nastaví `a` jako jméno nového objektu (seznamu).

Dobrá představa je vidět (i složitější) data jen jako bezejmenné struktury v paměti a proměnné jako šipky ukazující na ně. Pokud přiřazením namíříme šipku jinam, samotná data v paměti se tím nezmění. Pokud objekty (např. seznam) obsahují jiné objekty (prvky seznamu), i toto si lze představit jako pouhé „šipky“ z „políček“ seznamu na objekty prvků. Příklad je uveden u popisu seznamů níže.

[8] Pracuje s úhly v radiánech.

Jakožto pouhá jména nemají proměnné v Pythonu (na rozdíl od Pascalu a dalších) daný pevný typ, ten mají až data v nich uložená. Můžete tedy napsat `a=1; a="Tralala!"`.

Další typy dat

Nemusíte je zvládnout všechny napoprvé, existuje jich ve skutečnosti mnohem více, ale pro začátek budou stačit.

Nic neboli typ `NoneType` má jen jedinou možnou hodnotu (`None`) a vyjadřuje nulovou informaci. Můžete jej používat pro nedefinováno či chyba (ačkoliv ty se řešívají jinak).

Pravda a lež, neboli Boolovský typ `bool` má dvě hodnoty: `True` (pravda) a `False` (nepravda). Používá se při rozhodování, vrací ho operátory `==` (rovnost, nezaměňovat s `=` pro přiřazení), `!=` (nerovnost), `>`, `<`, `>=`, `<=` (řetězce, seznamy a podobné se porovnávají lexikograficky[9]) a logické operátory `and`, `or` a `not`.

Řetězec (typ `str`) je v Pythonu libovolný (i prázdný) text. Píše se do uvozovek (`"MaM"`) či jednoduchých uvozovek (`' je super!'`). Rozdíl je jen v tom, že v jednoduchých uvozovkách nemůžete přímo dvojité uvozovky a naopak. Pokud chcete do řetězce vložit nějaký speciální znak, je potřeba napsat jej nebo jeho kód za zpětné lomítko. Můžete napsat třeba: `\" \' \\ \n \t` pro uvozovku, apostrof, zpětné lomítko, zlom řádku (nový řádek) a tabulátor. Python nerozlišuje mezi znakem a jednopísmenným řetězcem.

Řetězce lze spojovat sčítáním: `"MaM"+' je super!'`. Jednotlivé znaky řetězce získáte indexováním: `"abc"[1]` je `"b"`, protože *Python indexuje od 0.* Výraz `"abcdef"[2:5]` má hodnotu `'cde'`, intervaly se uvádějí od (včetně) – do (nevčetně). Je možné i vynechat jeden z indexů (`'abc'[1:] == 'bc'`) nebo použít záporné číslo (počítá od konce: `"abc"[-1] == "c"`, `'abcde'[1:-2] == "bc"`). Pokud indexujete jeden znak mimo rozsah, je to chyba, ale v případě intervalu dostanete průnik (`"MaM"[24:] == ""`).

Převádět mezi řetězci a čísly (či jinými typy) můžete pomocí `str(42)` a např. `int("42")`. Pozor, `42` a `"42"` jsou naprosto odlišné hodnoty!

Seznam (či pole, typ `list`) je podobný řetězci, ale neskládá se ze znaků, ale z libovolných dat. Píše se do hranatých závorek, např. `[1,2,3]` je seznam tří prvků, ale i `["MaM", [42, 42], [] ]` je tříprvkový seznam – jeho druhý prvek je dvouprvkový seznam a poslední je prázdný seznam. V Pythonu můžete typy prvků v seznamu libovolně kombinovat, hloubka vnoření dat je omezena jen pamětí a zdravým rozumem.

Seznamy se indexují stejně jako řetězce (`[1,2,3][2] == [1,2,3][-1]`) a platí pro ně i táž pravidla intervalového indexování, ale zatímco běžné indexování zde vrací prvky, intervalové vrací seznamy.

Prvky seznamů lze (na rozdíl od řetězců) měnit přiřazením: po `a=[1,2,3]; a[1]=42` bude `a == [1,42,3]`. Pro úplnost bychom měli ještě dodat, že jednotlivé prvky seznamu (a dalších měnitelných struktur) jsou opět jen „šipky“

[9] Zleva po složkách – první s první, v případě rovnosti druhé, ...

ukazující na další data, takže např. po vytvoření seznamu seznamů: `a=[ [1,2], [3,4] ]; b=a[1]; a[1]=42` bude `a == [[1,2], 42]`, ale stále `b == [3, 4]`.

Do seznamu jménem `a` lze přidat prvek příkazem `a.append(x)` (na konec) nebo `a.insert(x, pozice)`. Setřídí ho `a.sort()`, obrátí `a.reverse()` a index prvního daného prvku vrátí `pozice=a.index(prvek)`[10]. Zápis též viz *Objekty a metody* níže.

Společné vlastnosti a funkce.

Délku řetězců a seznamů vrací funkce `len(x)`. Zda se prvek nachází v seznamu či zda se podřetězec vyskytuje v textu zjišťuje operátor `in`, např. `" Ma" in 'Res MaM!' == True`. Pokud vás zajímá typ hodnoty výrazu či proměnné, řekne vám ho funkce `type(x)`.

Skoro všechny typy lze použít místo `bool` v podmínkách či logických výrazech. `None`, `0` a `0.0`, `""`, `[]` a `()` se vyhodnotí jako `False`, vše ostatní jako `True`. Pozor na to, že už např. seznam `[()]` je jednoprvkový a vyhodnotí se jako pravda.

Objekty a metody

Když v seriálu zmiňujeme *objekty*, myslíme tím především kus dat paměti, se kterým se dá pracovat jako s celkem, tedy číslo, pole, řetězec a pod. Každý takový objekt je ale zároveň objekt ve smyslu objektového programování. K podrobnostem a vytváření vlastních objektů se dostaneme v pokračování.

K objektu mohou být přiřazeny funkce (tzv. *metody*), které pracují s daným objektem, aniž by bylo třeba předávat jej jako parametr. Například přidávání na konec seznamu `a` se provádí pomocí `a.append(x)` místo `append(a,x)`.

Významově (sémanticky) to na první pohled není velký rozdíl, ale pomáhá to programátorovi určit, se kterým objektem se bude pracovat (se seznamem, přidávaný prvek se tu nemění), zároveň to umožňuje definovat metodu jménem `append` pro různé typy různě, zatímco `len` je nadefinována najednou a musí vždy zjišťovat, jaký typ parametru dostala.

Objektový pohled na všechna data včetně těch nejzákladnějších (čísla) umožňuje pracovat se všemi objekty podobně a není tedy potřeba např. rozlišovat pole čísel a pole řetězců. U složitějších struktur se více hodí si představovat objekty jako „černé skříňky“ s pákami (metody), kterými je lze ovládat, aniž by bylo třeba se neopatrně a složitě hrabat uvnitř.

Krutá pravda: *Vše v Pythonu je objekt*, včetně funkcí, modulů a typů samotných. Ale nenechte si tím zamotat hlavu, přináší to spíše široké možnosti, než komplikace. Většinu času můžete tento fakt ignorovat a používat objekty pouze na úrovni `a.append(x)`.

Některé objekty mají krom metod ještě atributy (např. všechna (nejen komplexní) čísla mají `a.real` a `a.imag` pro reálnou a imaginární složku). Ty se chovají spíš jako proměnné (i když ne vždy se do nich dá zapisovat) a je možné přímo číst jejich hodnotu (bez volání, tedy např. `(1.0).imag == 0.0`).

[10] Funguje též pro hledání znaku či podřetězce v řetězci.

Část z metod a atributů začíná a končí dvěma podtržítky. Tyto jsou „interní“ a přímo se většinou nepoužívají, někdy jsou to interní názvy pro běžné operace (např. `[42].__getitem__(0)==42`). Můžete zkusit např. `(-1).__abs__() == 1` nebo `len.__name__ == 'len'`.

Pro seznam všech metod objektu použijte funkci `dir(objekt)`. Mnoho objektů a funkcí má krátký popis v `x.__doc__` (anglicky). V `ipython`-u lze použít doplňování tabulátorem a `x?` pro tento popis.

Zdrojové soubory

Větší programy v Pythonu je možné psát do souborů s příponou `.py` a pak je spouštět nebo načítat jako moduly. Soubory v Pythonu není potřeba nijak překládat.

Soubor s pythoním zdrojovým kódem je obyčejný textový. Python zpracuje takový soubor tak, jako byste ho „příkaz po příkazu“ napsali v interaktivním módu. Soubory se hodí hlavně pro definování funkcí a objektů – ty lze sice definovat i v interaktivním módu, ale je to nešikovné.

Vlastní soubor spustíte z příkazové řádky příkazem `python soubor.py`

Pokud chcete spustit soubor v rámci interaktivního módu, můžete použít funkci `execfile("soubor.py")`[11]. Program za běhu uvidí proměnné, které již byly nastaveny, a nechá po sobě modifikace i nové proměnné a definice.

Vytvořit vlastní modul je celkem snadné, budeme se tomu věnovat příště.

Ještě užitečné maličkosti: Pokud v souboru použijete znak `#` (mimo řetězec), Python bude jej a zbytek řádku považovat za komentář a ignorovat. Funkce `print(x)` vypíše hodnotu x a přejde na nový řádek. Funkce `input(zprava)` vypíše zprávu, počká na vstup z klávesnice a ten pak vyhodnotí jako výraz a vrátí výsledek – hodí se např. pro načítání čísel ze vstupu.

Definice funkce

Funkce se v Pythonu definuje takto:

```
def mojefce(x,y): # jméno funkce a parametrů
    a = x+y       # a další příkazy, všechny stejně odsazené
    return a+1    # co funkce vrátí
```

Všechny řádky funkce musí být stejně odsazené – Python podle toho pozná, kde funkce končí. Na velikosti odsazení nezáleží, ale doporučují se 4 mezery.

Funkce je v podstatě pojmenovaný kousek programu s parametry. Po jejím zavolání se provede jako podprogram, ovšem nemění „vnější“ proměnné a může vytvářet vlastní proměnné (zde `a`), které pak nebudou vidět „vně“. Funkce při volání (např. `z=mojefce(1+2, len([]))`) neuvidí jako parametry výrazy, ale přímo jejich hodnoty, tedy zde například bude `x=3; y=0`.

Funkce by měla vidět okolní svět jen skrze předané parametry. Parametry funkcí jsou (stejně jako vytvořené proměnné) vidět jen uvnitř funkce a různé

[11] Můžete předat i absolutní či relativní cestu k souboru.

funkce mohou mít stejná jména parametrů. Na vysvětlení a podrobnosti se můžete těšit příště.

Funkce může používat jiné funkce a dokonce i sebe sama, ale pozor na zacyklení! Příkaz `return` funkci okamžitě ukončí, i kdyby za ním byly další příkazy. Pokud `return` nepoužijete, funkce vrátí `None`.

Úloha 1.6 – Seriál o pythonu I. díl (6b)

Pro řešení úloh zkuste použít jen to, co jsme v seriálu zatím popsali.

I (1b): Popište, co dělá program a jakou hodnotu bude mít `v`. Nepoužívejte při tom Python (vyzkoušet, co dělají použité funkce, samozřejmě můžete).

```
s="(10!)=3 628 800"; a=s[s.index(' ')-1]; b=s[s.index(a)+2:]
    a=int(a); b=b[b.index(" ")+1:]; b=int(b); v=b/2**a
```

II (1b): Najděte nejmenší nezáporné hodnoty `a` a `b` typu `int`, aby platilo:

```
max(str(a), str(b)) != str(max(a,b))
```

III (2b): Napište funkci `roztret(x)`, která seznam či řetězec rozdělí co nejlépe na třetiny a vrátí trojici (seznam) těchto třetin. Třetiny se nesmí překrývat a dohromady musí dát celý vstup. Jejich délky musí být nerostoucí.

IV(2b): Napište funkci `cifra(x,k)`, která vrátí `k`-tou cifru zleva desítkového zápisu `x`. Předpokládejte celé $\texttt{k} \geq 1$, `x` může být celé i reálné (i záporné). Vyhněte se převodu na řetězec.

Co to je M&M a jak začít řešit

M&M je korespondenční seminář a zároveň studentský časopis zaměřený na matematiku, fyziku a informatiku. Pokud se rozhodneš zapojit, budeme ti v průběhu roku posílat poštou (samozřejmě zdarma) nová čísla se zadáním a řešením úloh a témat. Zároveň je budeme zveřejňovat na našich webových stránkách. Budeš pak mít zhruba měsíc na přemýšlení a v termínu, který je uveden na začátku každého čísla, pošleš svoje řešení na adresu redakce. S dalším vydáním časopisu ti přijdou tvé příspěvky zpět opravené a obodované.

Úlohy

V každém čísle otiskujeme zadání několika úloh. Nejsou to obyčejné příklady z hodin matematiky a fyziky. Některé vyžadují hlubší zamyšlení, v jiných musíš odhalit logický trik, v dalších si trochu započítáš. Bodové hodnocení úlohy, zpravidla 1–5 bodů, je uvedeno vedle jejího názvu. Za elegantní nebo zajímavé řešení však můžeš dostat bodovou prémii.

Pokud řešením úloh jedné série dosáhneš určité bodové hranice (několik pětin celkového uvedeného počtu bodů za úlohy), dostaneš bonus s ohledem na to, v kolikátém jsi ročníku na čtyřletém gymnáziu (pokud jsi v nižším ročníku, řadíme tě mezi prváky), a to podle následující tabulky:

ročník	1/5 b	2/5 b	3/5 b	4/5 b	5/5 b
1.	+1 b	+2 b	+3 b	+4 b	+5 b
2.	0	+1 b	+2 b	+3 b	+4 b
3.	0	0	+1 b	+2 b	+3 b
4.	0	0	0	+1 b	+2 b

Ocenění

V průběhu roku tvoje body sčítáme a v každém čísle otiskujeme aktuální žebříček řešitelů. Jakmile dosáhneš určité bodové hranice (sčítají se i body z předchozích ročníků M&M), získáš seminární titul, který bude uveden u každého tvého článku a ve výsledkové listině. Už za 10 bodů získáš titul Bc.^{MM} (čili borec), za 20 budeš Mgr.^{MM} (machr), pokud dosáhneš na hranici 50 bodů, stane se z tebe Dr.^{MM} (dříč), při stovce bodů získáš titul Doc.^{MM} (dokonalý) a při 200 bodech už budeš Prof.^{MM} (profík). Výzvou pro tebe může být získání titulu Akad.^{MM} (abnormální kandidát) za 500 bodů – této mety ještě nikdo nedosáhl.

Abys poznal(a) své kolegyně a kolegy ze semináře a také nás, organizátory, vybíráme dvakrát do roka 20–30 nejpilnějších řešitelů, které zveme na jarní a podzimní soustředění. Pro ty úplně nejlepší jsou navíc na konci ročníku připraveny odměny.

Soustředění

Jarní a podzimní týdenní soustředění je odměnou pro nejlepší řešitele, tj. takové, kteří se umístí přibližně do 30. místa ve výsledkové listině. Jestli budeš mít možnost zúčastnit se soustředění, záleží především na tvé píli a snaze při řešení

úloh a témat během celého roku. Na soustředění se během přednášek dozvíš mnoho nových zajímavých věcí z matematiky, fyziky, informatiky, astronomie i dalších oborů a také si zahraješ celou řadu více i méně tradičních, matfyzáckých i ryze nematfyzáckých her. Protože soustředění je za odměnu, hradíš si kromě dopravy pouze minimální část nákladů. Takže pilně řešit se rozhodně vyplatí!

Podzimní soustředění

I letos pro vás chystáme podzimní soustředění. Pozveme na něj deset nejlepších řešitelů z minulého roku. Zbytek (tj. asi patnáct řešitelů) doplníme z těch, kteří nám pošlou dobrá řešení alespoň některých úloh a témat z tohoto čísla nejpozději do 27. září, přičemž pět nejlepších nováčků (řešitelů, kteří ještě nebyli na žádném soustředění) dostane přednost.

Soustředění se bude konat v době od **16. do 24. října nedaleko Modravy v NP Šumava**. Na soustředění budeme vybírat účastnický poplatek do 500 Kč. Ubytování a stravování bude zajištěno, na vás je jen dopravit se na místo. Podrobnější informace rozešleme spolu s pozvánkou.

Ke svému řešení prosím připiš, jestli na soustředění jet chceš nebo nechceš. Ušetříš nám trochu starostí s pozváním správného počtu řešitelů.

Soutěžní pokyny

Do řešení M&M se můžeš zapojit kdykoli během školního roku. Nemusíš řešit všechny úlohy, vyber si především to, co tě baví. Má smysl posílat i náznak řešení. Nepiš jen výsledky, důležitější než čísla jsou myšlenkové postupy, kterými ses ubíral(a). Řešení každé úlohy nebo témátka napiš na *zvláštní papír* a nezapomeň se *podepsat*!

Svá řešení můžeš posílat i elektronicky e-mailem na adresu redakce uvedenou také na konci každého čísla. Pro elektronická řešení platí podobné pokyny jako pro řešení papírová.

Ušetříš nám mnoho práce, pokud řešení jednotlivých úloh odešleš v jednom e-mailu, každou úlohu vždy jako jeden soubor nebo archív více souborů. Všechny soubory, které nám pošleš, by měly obsahovat tvé jméno a označení úlohy či tématu.

Jako formát si prosím vyber jeden z následujících: Postscript, PDF, TEX, OpenDocument (*např. program OpenOffice.org*), MS Word Document či čistý text. Pokud posíláš čistý text či TEX, uveď též, jaké používáš kódování češtiny. Rozhodně není vhodné posílat řešení jako obrázek (např. fotografie ručně psaného řešení). Z formátů archívů je nejvhodnější použít zip či tar a gzip.

Pokud posíláš řešení tématu či konferenční příspěvek ze soustředění, pošli nám, prosím, i co nejvíce zdrojových kódů (např. TEXspolu s PDF či zdrojové kódy programů). Ušetříš nám tím mnoho zbytečné práce při přepisování do TEXu – stejně jako tím, že nám místo papírového řešení pošleš elektronické (i kdyby to byl jen čistý text).

Pokud chceš dostat potvrzení, že jsme tvé řešení v pořádku dostali, napiš to do textu e-mailu.

S prvním řešením nám prosím pošli jméno, adresu pro korespondenci (kam ti budeme posílat časopis a opravená řešení), adresu školy, ročník a rok, kdy budeš maturovat, a to i v případě, že nám budeš řešení zasílat elektronicky! Pokud přidáš i e-mail a telefonní číslo, budeme rádi.

Internet

Na adrese `http://mam.mff.cuni.cz` se můžeš dozvědět další informace o M&M, nahlédnout do archivu minulých ročníků nebo si prohlédnout fotky z minulých soustředění. Najdeš zde také návod, jak se přihlásit do naší e-mailové konference. Díky ní tě můžeme snadno a rychle informovat o vydání dalšího čísla tvého oblíbeného časopisu nebo pozvat na víkendové setkání řešitelů. Ty můžeš konferenci využít pro komunikaci s kamarády ze semináře.

Organizátoři

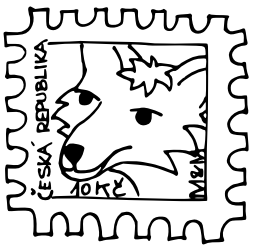
My organizátoři jsme většinou studenti různých oborů Matematicko-fyzikální fakulty Univerzity Karlovy v Praze, často bývalí řešitelé semináře. Během roku pro tebe vymýšlíme úlohy, opravujeme řešení a připravujeme soustředění. Těšíme se, že se na tom dalším setkáme třeba právě s tebou.

Anita, Bětka, Carlos, Eliška, Flavius, Honza, Hroch, Irigi, Jeffer, Klár(k)a, Kuba, Marble, Mára, Pepa, (R)adim, Terka, Tereza, Tomáš, Zuzka a Riki.

Adresa redakce:

M&M, OVVP, UK MFF
Ke Karlovu 3
121 16 Praha 2

Telefon: +420 221 911 235
E-mail: `MaM@atrey.karlin.mff.cuni.cz`
WWW: `http://mam.mff.cuni.cz`



Časopis M&M je zastřešen Oddělením pro vnější vztahy a propagaci Univerzity Karlovy, Matematicko-fyzikální fakulty a vydáván za podpory středočeské pobočky Jednoty českých matematiků a fyziků.