**8. PRG – Proměnné, datové typy, operace**

Proměnná:

* Proměnná je nějaká dočasná entita, kterou můžeme v proběhu programu měnit a pracovat s ní, není statická; v paměti PC vydrží do ukončení programu
* V názvu proměnné se mohou používat je znaky ENG abecedy, podtržítka, pomlčky a číslice (ale název proměnné nesmí číslem začínat)
* Velikost proměnné je omezena pouze velikostí paměti PC, neexistuje pevně daný limit velikosti

Operátor přiřazení:

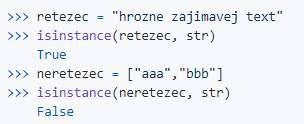
* jedná se o =
* na levé musí být název proměnné a na pravé hodnota, jiná proměnná, číslo, výraz, funkce
  + x = 5
  + x += 5 => x = x + 5
  + x -= 3 => x = x - 3
  + x \*= 3 => x = x \* 3
  + stejně platí pro všechny ostatní operátory, slouží pouze ke zkrácení zápisu

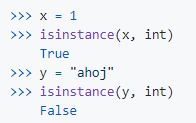
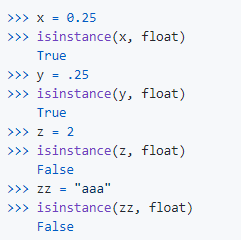
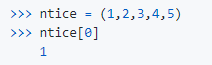
Datové typy:

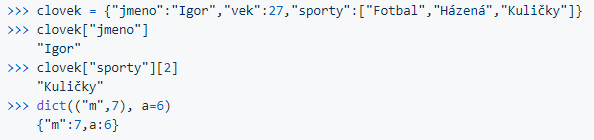
* textové: str
* numerické: int, float, komplex
* sekvenční: list, tuple, range
* mapovací: dict
* sety: set, frozenset
* booleovské: bool
* binární: bytes, bytearray, memoryview

Nejvýznamnější datové typy:

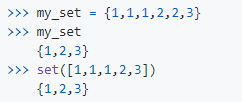
* **Řetězec (string, str)** – sled znaků, definuje se pomocí jednoduchých nebo dvojitých uvozovek



* **Celé číslo (integerer, int)** – celé číslo, jehož velikost je limitována pouze pamětí PC, definuje se jako obyčejné číslo
* **Desetinné číslo (float)** – uchovává v paměti des. číslo, není ale neomezeně přesné, nepíše se čárka ale tečka
  + čísla začínající 0 se dají psát i bez 0 (0.25 je stejné jako .25)
  + definuje se stejně jako int, jen s desetinnou tečkou
* **Seznam (list**) – jeden z nejvyužívanějších datových typů v Pythonu
  + definuje se pomocí hranatých závorek [] a prvky se v něm oddělují čárkou
  + můžeme je též tvořit za pomocí funkcí 'list()' a 'range()' jen pozor na funkci range, která vrací vlastní datový typ, který je třeba ještě prohnat funkcí list
  + list může obsahovat prakticky jákekoliv datové typy. Obshahuje li seznam více seznamů, tak hovoříme o více rozměrnémseznamu
  + k jednotlivým hodnotám seznamu můžeme přistupovat za pomocí indexu
  + list lze rozšířit a mazat z něj
  + rozšiřuje se pomocí metody append
  + mazat lze více způsoby
* **N-tice (tuple)** – velmi podobný listu, jediný rozdíl je že se do něj nedá zapisovat ani mazat -> je rychlejší; definuje se klasickými závorkami (), prvky se oddělují čárkou
* **Slovník (dictionary, dict)** - hodnoty jsou seřazeny do párů: klíč ('key') a hodnota ('value')
  + slovníky definujeme složenými závorkami '{ }', párové hodnoty dělíme dvojtečkou ':' a páry mezi sebou čárkou ','
  + lze je též definovat funkcí 'dict()'
  + hodnoty získáváme pomocí klíče (slovník[klíč] : hodnota )
  + zadáme-li neexistující klíč, dostaneme key error





* **Bool** – pravda/lež, ano/ne, 0/1
* **Set** – seznam, ve kterém se každá hodnota vyskytne pouze jednou
  + definuje se složenými závorkami {}, ale jen s jednou hodnotou, nebo pomocí funkce set()

Převodní funkce:

* **int()** – převede na celé číslo, když nemá co převést vyhodí error
* **float()** – převede na desetinné číslo
* **str()** – převede na string (řetězec)
* **ord()** – převede znak na ASCII hodnotu
* **chr()** – převede ASCII hodnotu na znak

Vstup a výstup:

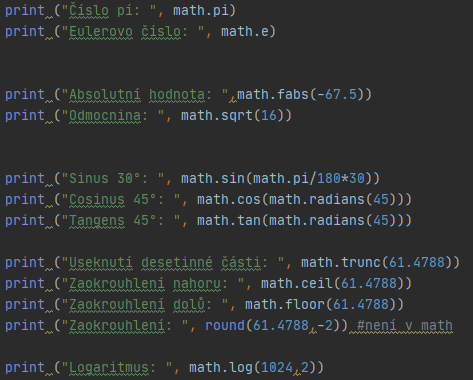
* *input()* – příkaz pro vstup, program čeká na vstup od uživatele, vrací string
* *print()* – vypíše řetězec, číslo, return funkce... co konzole, každý print je jeden řádek, pomocí *end* tomu lze zamezit, že se nastaví místo konce řádku jiný znak

Aritmetické operátory:

* Sčítání: x + y
* Odčítání: x – y
* Násobení: x \* y
* Dělení: x / y
* Modulo (zbytek z celočíselného dělení): x % y
* Umocnění: x \*\* y
* Celočíselné dělení: x // y

Import, použití knihoven:

* příkaz import má 3 podoby
  + *import random* – základní, v tomto případě se na knihovnu odkazujeme *random.randint()*
  + *from random import randint, from random import \** - buď můžeme importovat pouze část knihovny, nebo celou, v obou případech se odkazujeme pouze *randint()*, v tom případě u \* je problém, pokud by ve dvou a více importovaných knihovnách byla stejná funkce, bude způsobovat kolize, použije se ta funkce z knihovny, která se importovala později
  + *import random as rn* – v tomto případě se na knihovnu random odkazujeme *rn.randint()*
* **Knihovna math** – math má spoustu funkcionalit, obsahuje prakticky všechny matematické operace
  + goniometrické funkce – úhel se jim zadává v radiánech a musí se převádět (to se určitě zeptá)

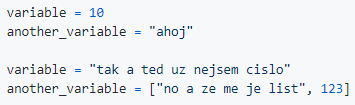
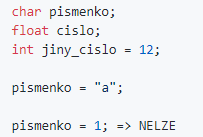


Komentáře: pouze #, ale používáme i trojité uvozovky, to v tom případě není komentář ale řetězec víceřádkový

Ostatní operace a operátory:

* operátory se využívají k provedení operací na proměnných, nebo hodnotách
* python dělí operátory do několika skupin:
* Porovnávací operátory:
  + Rovnost: x == y
  + Nerovnost: x != y
  + Větší než: x > y
  + Menší než: x < y
  + Větší nebo rovno: x >= y
  + Menší nebo rovno: x <= y
* Logické operátory:
  + and – vrací True, pokud jsou obě tvrzení pravdivá: x < 5 and x < 10
  + or – vrací True, pokud je alespoň jedno tvrzení pravdivé: x < 5 or x < 10
  + not – obrací logiku tvrzení (False na True a naopak): not(x < 5 and x < 10)
* Identitní operátory:
  + in - Vrátí True, pokud se hodnota nachází v sekvenci: x in y
  + not in - Vrátí True, pokud se hodnota nenachází v sekvenci: x not in y
* Bitové operátory:
  + & - AND - logické 'a'
  + | - OR - logické 'nebo'
  + ^ - XOR
  + ~ - NOT
  + << - Zero fill left shift
  + >> - Signed right shift

Nějakej ten bonus co Stejsky nechce ale je cool kdyžtak:

* V programování existují 2 základní typy deklarování proměnných:
* **Dynamický typový systém**
  + Ve své hluboké podstatě proměnná nemá žádný typ (int, string,...)
  + Proměnné se velmi často nemusí deklarovat a mohou svůj typ měnit v průběhu programu
* **Statický typový systém**
  + proměnné jsou zde striktně definované a mají pevný datový typ, který se nemění.
  + Díky tomu běží program rychleji, protože může jen jednou vyhodnotit, že nějaká proměnná má nějaký datový typ a do konce s tím může počítat. U dynamického typového systému to tak nejde.
* Proměnné v pythonu se řadí do dynamického typového systému
* V kontextu proměnných stojí za zmínku rozdíl mezi kompilací a interpretací:
  + **Kompilace**: celý kód se zároveň přeloží do strojového kódu, zkompilovaná verze se spouští => nutný statický typový systém, protože kompilátor si nedokáže uhlídat, jestli se nějaké proměnné mění a jak. Potřebuje to mít pevně dané.
  + **Interpretace**: intepret postupně překládá a spouští program => možnost dynamického typového systému, protože interpreter si s tím poradí.
* Dynamický typový systém (Python):
* Statický typový systém (C++):