**Python**

1.lekcia:

súbory sa ukladajú do .py, spúšťajú sa cez terminál

print(„“Ahoj, „svet!“) – jednoducho vypíše hodnotu z argumentov

# - komentár

**Datové typy**

na získanie nejakého znaku zo stringu -> “ahojsvet“[3] -> j

int(„45“) -> 45

str(912) -> „912“

type(x) – vráti triedu x, či to je str alebo int,...

len(x) – vráti dĺžku x, pri str počet znakov, pri tuple alebo list vráti počet prvkov

**1. tuple** - N-tica, nedá sa zmeniť hodnota pola (1, „str“, 2)

**2. list**  - zoznam [‚ahoj‘, ‚svet‘, 123]

|  |  |
| --- | --- |
| List.append(item) | Prida item na koniec listu |
| List.insert(value[int], item) | Pridanie itemu na dany index |
| List.remove(item) | Odstranenie itemu |
| List[0] | Vypisanie prveho itemu |
| List[2] = newItem | Prenastavenie itemu na indexe 2 |

**Logické operátory**

**is / is not** – či dva odkazy na objekt odkazuju na ten istý objekt

**<, <=, ==, !=, =>, >**  - da sa retazit: 0 <= a < 10

**in / not in** – či sa nachádza v tuple alebo v liste

**and / or / not** – boolean operátory

**Riadenie toku programu**

|  |  |
| --- | --- |
| **if:**  if logický\_výraz1:  sada1  elif logický\_výraz2:  sada2  else:  iná\_sada | **while:**  while True:  if not x:  break;  print(); |
| **While:**  While logicky\_vyraz:  sada | **For**  For i in [‚Abc‘, ‚nazdar‘]/list/“ABCDEFGHIJKL“:  Print(i) |

**Ošetrovanie výnimiek**

try: \n i = int(input(‚zadajte celé číslo‘));

except ValueError as err: \n print(err)

**Aritmretrické operátory**

|  |  |
| --- | --- |
| + | Sčítanie |
| - | Odčítane alebo negacia |
| \* | Násobenie |
| / | Delenie ale vracia desatinné čísla |
| += | A = A + 5, pri stringoch += sa vytvorí nový objekt s tým istým odkazom ako starý objekt ale + ho nevytvorí, tak to funguje aj pri listoch a tuple pomocu += [‚nieco‘] |

**Vstup a výstup**

na výstup môžeme cez CMD *C:\code\Python\1.lekcia>cislo2.py > res.txt* napísať všetky print výstupy do txt súboru a pomocou *C:\code\Python\1.lekcia>cislo2.py < data.dat* môžeme zobrať input vstup, súbor musí byť .dat, vstup aj vystup sa robí: python subor.py < vstup.txt > vystup.txt

**Vlastné funkcie**

def nazov(arg, arg2):

#nieco rob

return hodnota#dobrovolne

**moduly**

**modul random**

random.randint(min, max), random.choice() – náhodne vyberá prvky

**modul sys**

sys.argv[0] – vracia názov súboru

2.lekcia

jedno podtržítko sa pouíva vo for in loop ked nas identifikator vobec nezaujima. napr

for \_ in [1,2,3,4,5]: print(‚ahoj‘)

niektoré operácie:

|  |  |  |  |
| --- | --- | --- | --- |
| Abs(x) | Absolutna hodnota | Oct(i) | I v osmičkovej |
| Divmod(x,y) | Podiel a zvyšok | Bin(i) | i v dvojkovej |
| Pow(x, y) | X \*\* Y | Hex(i) | I v hex |
| Pox (x, y, n) | (X \*\* Y) % N | Int(i) | Prevod na int(ValueError/TypeError) |
| Round(x, n) | X zaokruhlene na n číslic | Int(s, base) | Prevod na int do base<2;36>(ValueError)­ |

0b1101101101010110 – zapis cisla v binarnej

0o65432147 – zapis cisla v osmickovej

0xDECAB15872A – zapis cisla v sestanskovej

pomocou ord() a chr() sa prevadza medzi ascii a sustavami

pre pristup znakov z textu používame [], počítaju sa od 0, ak chceme ísť od zadu tak -1, ...(IndexError)

seq[začiatok]

seq[začiatok:koniec]

seq[začiatok:koniec:krok] – vypiše každe *krok* písmeno

„ „.join(list) – spoji prvky pola do textu, medzi ne da „ „

reverse(text) – otočenie text

index(text) – hlada text v str, vracia index alebo ValueError

find(text) – hlada text v str, vracia index alebo -1

capitalize() – prve pismenko velke

center(dlzka, znak) – pred a po string vlozi znak aby bola dlzka

count(znak) – pocita pocet znakov s texte

endswith(znak, zaciatok, koniec) – vracia True ak txt konci znakom alebo vyrez z txt

expandtabs(pocet) – nahradi tab s medzerami

isalnum() – ci je text len text a cisla(bez medzier)

isaplha() – ci je text len text

isdecimal() – ci je text len cislo v 10tkovej

isdigit() – ci je text cislicou

isidentifier() – ci je platny identifikator

islower() – ci su vsetky mozne znaky malymi pismenami

isnumeric() - ci je text cislicou

isprintable() – ci sa da text vypisat(vsetko okrem \n)

isspace() – ci je kazdy znak medzera

istitle() – ak je prve pismeno velkym

isupper() – ak je vsetko mozne velkym

ljust() – ako center len zarovna dolava

lower() – da na male znaky

partion(n) – text pred n, n ,text po n

replace(t,u,n) – nahradi t na n a to spravi aspon raz alebo n krat

split(t,n) – rozdeli text podla t alebo podla medzier maximalne nkrat

startswith(znak, zaciatok, koniec) - vracia True ak txt zacina znakom alebo vyrez z txt

strip(znak) – zbavi text znakom zlava aj zprava

swapcase() – prehodi velke a male pismena

title() – da prve velkycm

upper() – cely text velkym

zfill(dlzka) – dosadi pred text 0 aby text mal danu dlzku

format()

{0: [vypln] <>^ -+ # 0 [sirka] , .[presnost] typ}

vypln => hocico co sa vyplni do minimalnej sirky

<>^ => zarovanie dolava, doprava, na stred

-+ => vypisanie znamienka(- je dobrovolne, + je povinne)

# => prefix cisla, 0b 0x

0 => cislo sa vyplni nulami do minimalnej sirky

[sirka] => nastavenie minimalnej sirky

, => oddelenie milionov, stotisiscov, tisicok, stoviek,...

.[presnot] => aky dlhy ma byt desatinny rozvoj cisla

typ => pre int: b, c, d, n, o, x, X; pre float: e E f g G n %

3.lekcia

postupnost je typ v ktory podporuje in, len(), [] a je iterovatelny (bytearray, bytes, list, str, tuple)

**tuple**

list() – prevedenie tuple na zoznam

vytvaraju sa ()

x.count(y) – kolkokrat sa nachadza y v x (x je tuple)

x.index(y) – najlavejsi vyskyt y v x, inak ValueError

pouzit s operatormi + \* [] in not in

napr farby[-3:] je to iste ako farby[1:]

**list**

L.append(x) – pripoji x na koniec L.count(x) – spocita x

L.extend(m) – pripoji iteratovatelneobjekty na konec L.index(x,start,end) – index pozicia x

L.insert(i,x) – na index poziciu i da prvok x L.pop() – vrati a odstrani posledny prvok

L.pop(i) – vrati a odstrani prvok na indexe i L L.reverse() – obrati L na mieste

L.remove(x) – odtrani prve x, inak ValueError L.sort() – zoradi na mieste

[chr(x) for x in range(65,91)] – zoznamova komprehenzia

hashovatelne objekty = ich navratova hodnota je stale ista pocas zivota objektu (nemenia sa, nedaju sa menit), patri medze ne float, frozenset, int, str, tuple. Nepatri medzi nich dict, list a set.

**mnoziny set**

množina je menitelny, neindexovany datovy typ. Odkazuje na hashovatelne objekty

da sa vytvorit cez a = set(‘ahoj’) alebo {‘ahoj’}, nie vsak {} lebo toto vytvori dict

obsahuju vzdy jedinecne prvky

set(‘pecan’) | set(‘pie’) == {‘p’, ‘e’, ‘c’, ‘a’, ‘n’, ‘i’ } Zjednotenie

set(‘pecan’) & set(‘pie’) == {‘p’, ‘e’, ‘c’, ‘a’, ‘n’, ‘i’ } Prienik

set(‘pecan’) - set(‘pie’) == {‘p’, ‘e’, ‘c’, ‘a’, ‘n’, ‘i’ } Rozdiel

set(‘pecan’) ^ set(‘pie’) == {‘p’, ‘e’, ‘c’, ‘a’, ‘n’, ‘i’ } Symetricky rozdiel

s.add(x) – prida do mnoziny x ak tam nie je s.clear() – vycisti ju

s.copy() – vracia melku kopiu s.difference(t) – ako rozdiel

s.differnce\_update(t) – odstrati vsetky prvky t z s s.discard(x) – odtrani x

s.intersection(t) – vrati mnozinu so spolocnymi prvkami s.intersection\_update(t) – prienik

s.isdidjoint(t) – True ak nemaju spolocny prvok s.issubset(t) – s = t alebo s podmnozina t

s.issuperset(t) – s = t alebo s nadmnozina t s.pop() – odstrani nahodny prvok

s.remove(x) – odstrani x z s, inak KeyError s.symetric\_difference(t) – s ^ t

s.symetric\_difference\_update(t) – symetricky s.union(t) – zjednotenie

s.update(t) – z pride do s

**frozenset**

zmrazene mnoziny, neda sa ich menit

**slovniky**

d.clear() – odstrani prvky d.copy() – melka kopia

d.fromkeys(s, v) – z d kluc ma postupnsot s a prvky v d.get(k) – vrati hodnotu npopa k

d.get(k, v) – vrati hodnotu na k, inak v d.items() – pohlad tvor. dvojicami kluc hodnota

d.keys() – vrati pohlad klucov d.pop(k) – vrati a odstrani prvok k

d.pop(k, v) – vrati a odstrani prvok k, inak v d.popitem() – vrati a odstrani dvojicu

d.setdefault(k, v) – vrati k alebo nastavi k na v d.update(a) – to co je v a da do d

d.values() – vsetky hodnoty z d

4.lekcia

zjednodusenie podmienky

width = 100 + (10 if margin else 0)

pokial je margin True tak pride 10, inak 0

print("{0} subor{1}".format((count if count != 0 else 'ziadny'), ("y" if count in range(2,5) else ("" if count in range(0,2) else "ov")))) = vyuzitie pre vypis poctu suborov

po cycle while alebo for In moze ist blok else. Ten sa vykona ked ked skonci cyklus. nevykona sa ked cyklus konci break alebo return. Pri while sa blok else spusti ak podmienka nie je splnena

Vynimky:

try:

sada

except skupina:

sada

else:

sa prevedie pri normlanom dokonceni try, pri vynimke nie

finally:

sa prevedie vzdy

pokial chceme vyskocit z programu tak sa toto pouziva taky hack(lek4/raise\_vlastny.py) – vytvorim si vlastnu triedu, jej argument je Exception a jej telo tvori prikaz pass. Potom ked chces niekde v strede programu program skoncit tak staci cely program dat do try bloku a vyvlovat chybu - raise Nazov(). Za try dat except Nazov() a mam poriesene

pokial chceme dat do argumentu nejaku nticu alebo list, s tym ze i-ty argument je i-ty prvok, tak nemusime vsetko zapisovat ale iba napisat fcia(\*list) a vo funckii s tym pracovat ako s n-ticou

pokial to chceme spravit s dict {} tak sa pred argument napise \*\* a funcuje to ako jedna \*

lambda funkcie: area = lambda b, h: 0.5 \* b \* h

pokial chcem vyvolat spadnutie programu pri testovani ze nejaky argument sa mi nepaci tak napisem assert logicky\_vyraz, co\_robit

assert result, „Nulovy argument“