PTC-SL-01 Wstęp do programowania w języku Python

PODSTAWY TECHNIKI CYFROWEJ

LABORATORIUM SYSTEMÓW STEROWANIA PRZEMYSŁOWEGO I AUTOMATYKI BUDYNKÓW

KATEDRA ENERGOELEKTRONIKI I AUTOMATYKI SYSTEMÓW PRZETWARZANIA ENERGII
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ AKADEMIA GÓRNICZO-HUTNICZA

Temat:

Wstęp do programowania w języku Python

Cel: programy zrealizowane w języku Python, używające konstrukcji programowych koniecznych do realizacji programu konwersji liczb (ćwiczenie PSL-SL-02)

Narzędzia: Interpreter języka Python, środowisko IDE PyCharm Edu

Wprowadzenie

Python jest językiem zdobywającym coraz większe obszary zastosowań, od systemów wbudowanych, poprzez aplikacje desktopowe, po aplikacje serwerowe. Część serwisów internetowych tworzona jest właśnie z użyciem tzw. framework-ów właśnie dla języka Python. Duża ilość bibliotek (obecnie ponad 67 tysięcy! https://pypi.python.org/pypi), m.in. do obsługi baz danych, grafiki 3D, komunikacji (zarówno protokołami internetowymi czy przemysłowymi), jak i do zastosowań naukowych w wielu dziedzinach (matematyka, fizyka, chemia, itd.), analizy danych, przyczynia się do jego szerokiego obszaru praktycznych zastosowań. Warto również wspomnieć, że aplikacje w języku Python można uruchamiać na wielu systemach operacyjnych, poczynając od Windows, Linux, poprzez OS X, Solaris, aż do mniej znanych jak Tru64, IRIX, AS/400, BeOS, MorphOS.

Język Python wymusza prawidłowy styl programowania i dlatego jest jak najbardziej właściwym językiem do nauki programowania od podstaw, jako pierwszy język programowania.

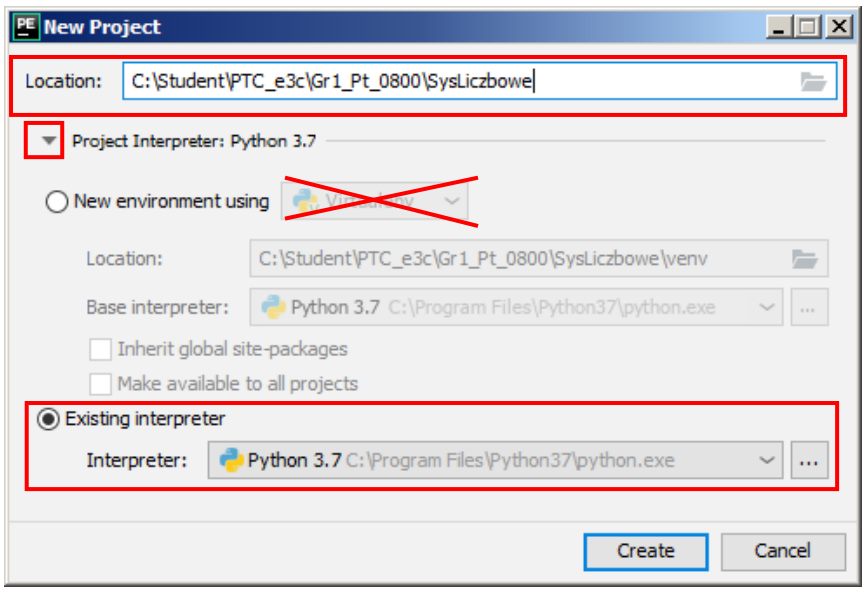
Jednym z elementów prawidłowego stylu programowania jest formatowanie kodu programu. W języku Python, każda instrukcja musi być zapisana w osobnej linii (w nielicznych wyjątkach dopuszcza się kilka instrukcji w jednym wierszu). Kolejną wyróżniającą język Python zasadą jest to, że dla poprawnego działania programu, istotne jest stosowanie prawidłowych **wcięć linii kodu**. Instrukcje warunków, pętli, zawartość bloku kodu objętego warunkiem lub pętlą "rozpoznają" po wcięciach o jeden poziom większych niż poziom wcięć danej pętli czy warunku.

W ćwiczeniu używany będzie interpreter języka Python, dostępny do pobrania ze strony www.python.org. Jako środowisko rozwojowe IDE używany będzie PyCharm w wersji edukacyjnej, PyCharm Educational Edition. Zawiera on m.in. samouczek języka Python, który w 50 krokach zapoznaje użytkownika z podstawami składni i funkcji języka Python. Przed rozpoczęciem zajęć, należy zapoznać się z tym samouczkiem. Jest on automatycznie uruchamiany po zainstalowaniu środowiska PyCharm Edu, a na komputerach laboratoryjnych można go uruchomić wybierając z okna startowego PyCharm Edu opcję "Introduction to Python".

Projekty w środowisku PyCharm Edu

Tworzenie programu w środowisku PyCharm wymaga utworzenia projektu, do którego będą dodawane pliki z kodem źródłowym programu. Okno powitalne umożliwia utworzenie nowego projektu (Create New Project) lub otwarcie istniejącego (Open).

Aby utworzyć projekt, należy kliknąć Create New Project, w następnym oknie wybrać katalog (C:\Student\PTC_e3c\[grupa]\SysLiczbowe, gdzie [grupa] to np. Gr1_Pt_0800) oraz wybrać interpreter wskazujący na zainstalowaną wersję Pythona i kliknąć Create. Przykładowe okno tworzenia nowego projektu pokazano na rysunku 1.



Rys. 1. Konfiguracja nowego projektu w środowisku PyCharm

Po chwili zostanie otwarte drzewo projektu, do którego należy dodać plik z kodem źródłowym Pythona. W tym celu z menu kontekstowego projektu należy wybrać opcję New, Python File i podać nazwę pliku (np. Program1.py).

Aby otworzyć istniejący projekt, w oknie startowym PyCharm Edu należy wybrać "Open" lub kliknąć na jednym z ostatnio otwieranych projektów wymienionych na liście po lewej stronie okna startowego.

Program ćwiczenia

Poniżej podano wytyczne do programów, które należy napisać i prześledzić w trybie wykonywania krokowego, zwracając szczególną uwagę na typy danych (int, str, long).

1. Napisać program: (zmienne, funkcja print, formatowanie tekstu, instrukcja for)
   - ustawiający w zmiennej nick wybraną przez siebie nazwę użytkownika,
   - wyświetlający w pętli for 3 razy napis „Twój nick to” oraz wartość zmiennej nick
   - zmodyfikować program aby wypisywał "Witaj <nick> w świecie Pythona"
2. Napisać program: (funkcja range, instrukcje for, if/else/elif)
   - ustawiający dwie wartości w zmiennych x1 i x2,
   - wyświetlający (w kolejnych wierszach) kolejno wszystkie liczby od x1 do x2 włącznie
   - program napisać tak aby uwzględniał dwa warianty: a) x1 <= x2, b) x1 > x2
   - zmodyfikować program tak, aby wypisywał wartość absolutną różnicy x1 i x2
3. Napisać program: (funkcje isdigit, int, len, range, instrukcje for, if, operator [])
   - ustawiający w zmiennej tekstWiek liczbę zapisaną jako łańcuch tekstowy
   - sprawdzający czy każdy znak zmiennej tekstWiek jest cyfrą dziesiętną
   - jeśli tak jest, to program ma przekształcić tekstWiek na liczbaWiek i wyświetlić napis „Do wieku emerytalnego pozostało” oraz wynik odejmowania 67 – liczbaWiek (lub 65 – liczbaWiek)
   - w przeciwnym wypadku program wyświetla napis "zmienna tekstWiek nie zawiera liczby dziesiętnej"
   - zmodyfikować program tak, aby zawierał zmienną czyKobieta (true/false) i wykonywał odpowiednie odejmowanie w zależności od płci użytkownika

4. Napisać program: (instrukcja while, operatory / i %)
   - ustawiający w zmiennych L1 i L2 dwie liczby całkowite,
   - wypisujący ich iloraz (L1/L2) oraz resztę z dzielenia L1/L2 (operator %)
   - po wypisaniu w/w informacji, program ma sprawdzić czy reszta z dzielenia wynosi 0, jeżeli nie, to zmniejsza o 1 zmienną L1 i wraca do poprzedniego punktu
   - zmodyfikować program tak, aby zamiast sprawdzać czy reszta z dzielenia wynosi 0, sprawdzał czy wynik dzielenia jest większy od 0 i jeżeli tak, ustawiał w zmiennej L1 wynik dzielenia i wracał do drugiego punktu
5. Napisać program: (funkcje ord, chr, len, operator [], definiowanie funkcji)
   - ustawiający w zmiennej dowolny tekst
   - wyświetlający w kolejnych wierszach kolejne znaki wprowadzonego tekstu wraz z ich kodami ASCII (np. A = 65)
   - przetestować program na dowolnym tekście oraz na liczbie (traktowanej jako tekst)
   - zmodyfikować program tak, aby wyświetlał kolejny znak oraz jego kod ASCII pomniejszony o kod ASCII znaku "0" (zero). Przetestować program na dowolnej liczbie traktowanej jako tekst – zauważyć iż kolejny znak (będący tekstem) równa się wynikowi odejmowania
   - przekształcić zasadniczą część programu w funkcję
   - zmodyfikować program tak, aby mógł być używany poprzez interfejs graficzny Tkinter, według podanego wzorca

W ramach przygotowania do ćwiczenia PSL-SL-02, zaleca się zapoznanie się z samo-uczkiem nt. tworzenia graficznego interfejsu użytkownika przy pomocy biblioteki Tkinter, którego stroną startową jest http://www.python-course.eu/python_tkinter.php

Skrótowy opis wybranych konstrukcji języka Python

Konwencja oznaczeń:
[nazwa w nawiasach kwadratowych] – należy podać opisany fragment programu, tzn.:
zamiast [nazwa_zmiennej] należy podać nazwę zmiennej (bez nawiasów kwadratowych!),
zamiast [wartość] należy podać np. 2.5,
zamiast [tekst] należy podać np. ”Podaj liczbę”
POGRUBIONE – słowo kluczowe lub inny element języka Python

# komentarz 1-linijkowy **'''** komentarz wielolinijkowy **'''**	wstawianie komentarze w pliku źródłowym
[nazwa_zmiennej] **=** [wartość]	przypisanie do zminnej podanej wartości np.: x = 5 s = ”tekst”
print([wyrażenie])	wypisuje podane wyrażenie (tekst lub wartość zmiennej). Wiele wyrażeń może zostać podanych, rozdzielonych przecinkami, pomiędzy wyrażeniami zostaną wówczas wypisane znaki spacji
print([sformatowany tekst] % (argumenty, ...))	wypisuje podany tekst, zawierające znaki formatowania zgodne ze standardem printf (%s, %d, %c, itp.), podstawiając pod kolejne znaki formatowania wartości podanych argumentów
for [zm.] **in** range([w.pocz], [w.końc])**:** [instrukcje]	wykonuje [instrukcje] ze zmienną [zm.] ustawioną na wartości od [w.pocz] do [w.końc.], wyłączając [w.końc], np. for i in range(1, 4): print i wypisze liczby 1, 2, 3 w kolejnych wierszach
for [zm.] **in range**([w.pocz], [w.końc], [krok])**:** [instrukcje]	wykonuje [instrukcje] ze zmienną [zm.] ustawioną na wartości od [w.pocz] do [w.końc.] z krokiem [krok], bez [w.końc] np. for i in range(3, 0, -1): print i wypisze liczby 3, 2, 1 w kolejnych wierszach
ord([znak tekstowy]**)**	zwraca wartość kodu ASCII podanego znaku np. ord(”A”) zwróci 65
chr([wartość]**)**	zwraca znak (tekstowy) odpowiadający podanemu kodowi ASCII. np. chr(65) zwróci ”A”
int([tekst]**)** **str(**[liczba]**)**	konwertuje liczbę dziesiętną z tekstu na liczbę oraz z liczby na tekst
long([tekst lub zm. tekstowa] **)**	zwraca liczbę typu long, będącą interpretacją podanego tekstu jako liczby dziesiętnej
len([tekst lub zm.tekst.]**)** **len(**[tablica]**)**	zwraca ilość znaków podanego tekstu lub zwraca ilość elementów tablicy

if [warunek]**:** [instrukcje]	jeśli warunek będzie spełniony, wykona [instrukcje]

if [warunek1]**:**
  [instrukcje_1]
elif [warunek2]**:**
  [instrukcje_2]
else:
  [instrukcje_3]
	jeśli warunek będzie spełniony, wykona [instrukcje_1], w przeciwnym razie sprawdza [warunek2] – jeśli będzie spełniony, wykona [instrukcje_2], a jeśli nie będzie spełniony, wykona [instrukcje_3].
	elif [warunek] jest opcjonalny i może się powtarzać wielokrotnie po if

while [warunek]**:** [instrukcje]	wykonuje [instrukcje] wielokrotnie (w pętli), dopóki [warunek] jest spełniony. Jeśli jedną z instrukcji będzie break, spowoduje to zakończenie wykonywania pętli (wyjście z pętli). [warunek] może również być warunkiem zawsze spełnionym – wyrażeniem **true**
[zmienna tekstowa]**[**pozycja**]**	zwraca znak zmiennej tekstowej z podanej pozycji (licząc od zera) np. print(”Ala ma kota”[4]) wypisze ”m”
a + b, a – b **a * b, a ** b**	operacje dodawania, odejmowania, mnożenia, potęgowania np. 2*3 = 6, 2**3 = 8 (2^3)
[liczba1] + [liczba2] [tekst1] + [tekst2]	operator + dla liczb wykonuje ich dodawanie, a dla łańcuchów tekstowych – łączenie, np. 2 + 3 zwróci 5 ”2” + ”3” zwróci ”23” ”aż” + ”3!” zwróci ”aż3!”
[liczba1] **/** [liczba2]	zwraca wynik dzielenia liczba1/liczba2. Jeżeli dzielone liczby są typu całkowitoliczbowego, wynik również będzie całkowitoliczbowy, np. 5 / 3 zwróci 1
[liczba1] **%** [liczba2]	zwraca resztę z dzielenia liczba1/liczba2, np. 5 % 3 zwróci 2
[tekst lub zm. tekstowa].**isdigit**()	sprawdza czy podany tekst zawiera jedynie cyfry (może być zinterpretowany jako liczba dziesiętna)