Wrocław, 14 października 2022

Programowanie **w j**ęzyku **P**ython — Laboratorium 4

Funkcje - Definicja i wywołanie funkcji, przekazywanie argumentów, zwracanie wartości

prowadzący: K. Kluwak

Wyjątki

Obsługa wyjątków pozwala zabezpieczyć program przed niespodziewanymi/niepożądanymi awariami na wypadek m.in.

- wprowadzenia danych złego typu
- wykroczenia poza zakres sekwencji
- sytuacji dzielenia przez zero
- ... i wielu innych...

Przykład: `lista = [2, 5, 7]`

```
lista[3] - wyrzuci wyjątek
```

Sposób 1:

```
lista = [2, 7, 3]
lista[3]

# try:
#     lista[3]
# except:
#     print(''Poza zakresem listy'')
```

Tym sposobem obsłużymy wszystkie wyjątki, ale nie wiemy jakiego typu wyjątek się pojawił.

Sposób 2:

```
# try:
#    lista[3]
# except IndexError:
#    print("Poza zakresem listy")
```

Korzystając z tej konstrukcji, ustawiliśmy program na wyłapanie konkretnego typu wyjątku.

Sposób 3:

```
# try:
#     lista[3]
# except IndexError:
#     print("Poza zakresem listy")
# else:
#     print("jeżeli blok try się powiedzie")
# finally:
#     print("ta linia wykona się zawsze")
```

Listę dostępnych wyjątków w języku Python można sprawdzić na stronie: https://docs.python.org/3/library/exceptions.html

Zad 3.1 *Stwórz kalkulator, który będzie odporny na niewłaściwe dane wprowadzane przez użytkownika. Program powinien pobierać pierwszą liczbę, drugą liczbę, znak operacji (+ - * /), wyświetlać wynik. Jeżeli użytkownik poda niepoprawną wartość, program pyta go ponownie o wpisanie właściwej informacji.*

Funkcje

Funkcje umożliwiają wielokrotne wykorzystywanie raz napisanego już kodu i pomagają w porządkowaniu programu. Działanie funkcji można przyrównać do funkcji w matematyce, w której po dostarczeniu danych (argumenty) otrzymujemy pewien wynik (wartości).
Przykładami funkcji z których korzystaliśmy do tej pory były m. in. print() oraz len(). Na ich przykładzie możemy zauważyć, że funkcja:

- posiada nazwę,
- może przyjmować argumenty,
- może zwracać dane (wyniki operacji).

Składnia funkcji w języku Python jest następująca:

```
# def nazwa_funkcji(argumenty):
#     <instrukcje do wykonania>
#     return dane
```

Funkcja:

- funkcja może przyjmować argumenty wybranego typu, jeśli podamy ich nazwy w nawiasie (...), ale nie jest to wymagane,
- może wykonywać pewne operacje, łącznie z wyświetlaniem komunikatów w konsoli, zapisywaniem plików, tworzeniem wykresów czy wyświetlaniem obrazów,
- może zwracać określoną wartość (lub wiele wartości) wybranego typu, przy użyciu instrukcji „return", ale nie jest to wymagane.

Przykład funkcji przyjmującej argumenty „a" i „b" i zwracającej wartość „c":

```
# def nazwa_funkcji(a, b):
#     c = a + b
#     return c
```

Argumenty funkcji mogą posiadać początkowo ustalone wartości. Wówczas jeżeli użytkownik nie wprowadzi wartości argumentów do funkcji, funkcja skorzysta z wartości domyślnych.

```
# def nazwa_funkcji(a = wartosc_1, b = wartosc_2):
#     c = a + b
#     return c
```

Funkcja może zwracać więcej niż jedną wartość w miejscu return

```
# def nazwa_funkcji(a = wartosc_1, b = wartosc_2):
#     c = a + b
#     d = a - b
#     return c, d
```

Korzystanie z dokumentacji i dokumentowanie kodu

Język Python daje nam możliwość wygodnego, prostego i inticyjnego korzystania z dokumentacji funkcji i klas:

- dla funkcji: `help(nazwa_funkcji)`
- dla klas:`help(nazwa_klasy)`

Aby samodzielnie utworzyć dokumentację funkcji (lub klasy) należy w linii po jej nazwie (pod nagłówkiem) wprowadzić tekst opisu w postaci komentarza (można korzystać z komentarzy wielowierszowych ograniczając blok tekstu symbolami """...""" lub '''...''').

Przykład:

```
    def parzystosc(liczba):
        """
        Ta funkcja sprawdza czy liczba jest parzysta
        """
        if (liczba%2 == 0):
            return True
        else:
            return False

help(parzystosc)
```

Zad 3.2 *Napisz funkcję bez argumetu, która wypisze w konsoli komunikat "Witaj świecie!"*

Zad 3.3 *Napisz funkcję, która przyjmie od użytkownika liczbę całkowitą i zwróci informację, czy jest parzysta, czy nie.*

Zad 3.4 *Napisz funkcję, która przyjmie od użytkownika dwie liczby i zwróci wynik ich dodawania, odejmowania, mnożenia i dzielenia.*

Zad 3.5 *Napisz funkcję, która przyjmie listę liczb i zwróci sumę z tych liczb.*

Zad 3.6 *Napisz program z funkcją, która pobierze od użytkownika jego imię oraz liczbę naturalną n, a następnie wyświetli na ekranie podane imię n razy. Jeżeli użytkownik nie poda żadnego imienia ani liczby, niech domyślnie funkcja ustawia wartości „Adam" oraz 7.*

Zad 3.7 *Związek między temperaturą w skali Celsjusza, a temperaturą w skali Fahrenheita ma postać:*

- $C = (F - 32) * (5/9)$ *(przy przeliczaniu skali Fahrenheita na Celsjusza),*
- $F = (C * (9/5)) + 32$ *(przy przeliczanniu skali Celsjusza na skalę Fahrenheita). Napisz dwie funkcje do przeliczania temperatur między poszczególnymi skalami.*

Sporządź dla powyższej funkcji dokumentację.

Zad 3.8 *Trójmian kwadratowy wyraża się wzorem $y = ax^2 + bx + c$. Utwórz funkcję, która będzie przyjmować jako argumenty współczynniki trójmianu kwadratowego a, b, c zwracać będzie miejsca zerowe oraz współrzędne wierzchołka funkcji. Funkcja powinna obsługiwać sytuację braku rozwiązań rzeczywistych. Sporządź dla powyższej funkcji dokumentację.*

Zad 3.9 *Napisz funkcję, która jako argument przyjmie listę liczb, a zwróci ich średnią. Sporządź dla powyższej funkcji dokumentację.*

Zad 3.10 *Napisz funkcję, która jako argument przyjmie listę liczb, a zwróci odchylenie standardowe. Wzór na odchylenie standardowe: $\sigma = \sum_{i=1}^{n} \sqrt{\frac{(x_i - \bar{x})^2}{n}}$.*

Zad 3.11 *Napisz funkcję, która przyjmie jako argument dwie listy liczb rzeczywistych, a zwróci odpowiadający im współczynnik korelacji Pearsona. Wzór na współczynnik korelacji Pearsona: $r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})}}$.*

Zad 3.12 *Napisz funkcję, która przyjmie jako argument dwie listy liczb rzeczywistych, a zwróci odpowiadające im współczynniki regresji liniowej. Wzory na współczynniki regresji liniowej: $a = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$, $b = \bar{y} - a\bar{x}$.*

Zad 3.13 *Utwórz program (z wykorzystaniem funkcji) do obliczania pól figur płaskich, który:*

- *pozwoli użytkownikowi wybrać figurę, której pole chce obliczyć,*
- *pobierze z klawiatury odpowiednie dane,*
- *wyświetli właściwy wynik,*
- *pozwoli użytkownikowi podjąć decyzję, czy chce uruchomić program ponownie, czy nie.*

Program powinien zawierać funkcje do obliczania pól następujących figur: trójkąt, prostokąt, koło, trapez, kwadrat, trójkąt równoboczny.

Zad 3.14 *Utwórz program (z wykorzystaniem funkcji) do obliczania objętości brył przestrzennych, który:*

- *pozwoli użytkownikowi wybrać bryłę, której objętość chce obliczyć,*
- *pobierze z klawiatury odpowiednie dane,*
- *wyświetli właściwy wynik,*
- *pozwoli użytkownikowi podjąć decyzję, czy chce uruchomić program ponownie, czy nie.*

Program powinien zawierać funkcje do obliczania objętości minimum trzech brył - kuli, graniastosłupa, stożka.

Zad 3.15 *Korzystając z biblioteki turtle utwórz funkcję, która będzie rysować:*

- *trójkąt*
- *kwadrat*
- *prostokąt*
- *pięciokąt*
- *gwiazdę*

Argumentami powinna być pozycja obrazka, kolor linii oraz jej grubość i wielkość kształtu.

Funkcje anonimowe (wyrażenia lamda)

W języku Python mamy możliwość tworzenia funkcji anonimowych (nazywanych też wyrażeniami lambda), które nie są powiązane z identyfikatorem (czyli w pewnym sensie nie mają nazwy). Służą do tworzenia funkcjonalności, którym nie powinniśmy nadawać nazw i/lub są potrzebne na krótkotrwały użytek.

W języku Python wyrażenia lambda mają następującą składnię:

```
lambda <zestaw argumentów>:
```

Tak utworzone wyrażenie lambda wydaje się bezużyteczne, ponieważ nie widać w nim jak do utworzonego wyrażenia wstawić argumenty. W języku Python istnieje wiele sposobów wstawiania argumentów do wyrażenia lambda. My skupimy się na dwóch:

- wywołanie wyrażenia od razu w tworzącej instrukcji: (lambda ¡zestaw argumentów¿:)(argumenty)
- przypisanie wyrażenia lambda do zmiennej: wyrazenie = lambda ¡zestaw argumentów¿: , wyrazenie(argumenty)

Przykład:

```
zm1 = (lambda x, y: x+y)(7, 3)
print(zm1)

wyrazenie = lambda x, y: x+y
zm2 = wyrazenie(7, 3)
print(zm2)
```

Zad 3.16 *Utwórz wyrażenie lambda do potęgowania (wyrażenie przyjmuje dwie liczby - potęgowaną wartość i wykładnik).*

Zad 3.17 *Utwórz wyrażenie lambda do składania dwóch łańcuchów znaków (wyrażenie przyjmuje dwa napisy i zwraca jeden będący sumą argumentów).*

Zad 3.18 *Utwórz wyrażenie lambda zwiększające argument o liczbę 13.*

Zad 3.19 *Utwórz wyrażenie lambda przyjmujące dwa argumenty (liczby rzeczywiste) i wykonujące na nich kwadrat sumy.*

Zad 3.20 *Utwórz wyrażenie lambda przyjmujące dwa argumenty (liczby rzeczywiste) i wykonujące na nich kwadrat różnicy.*