Technical University of Lodz
Institute of Electronics

Algorytmy i struktury danych

6. Funkcje

Łódź 2013

Ćwiczenie

- Edytuj kod programu
- Zapisz go w pliku hypot.py
- Uruchom skrypt

hypot.py

```
1  # hypot.py
2
3  from math import sqrt
4
5  def myhypot(x,y):
6      return sqrt(x ** 2 + y ** 2)
7
8  def main():
9      a = float(input("a: "))
10     b = float(input("b: "))
11     print "Hypotenous: ", myhypot(a,b)
12
13 main()
```

Ile funkcji jest wykorzystywanych w programie? Podaj nazwy.

Wykonywanie programu

BMI.py

```
1  # BMI.py
2
3  YourName = input( "Enter your name: " )
4  var = input ( "Enter your height [in cm]: " )
5  YourHeight = var / 100.0
6  YourMass = input ( "Enter the mass of your body [in kg]: " )
7  BMI = YourMass / ( YourHeight * YourHeight)
8  print ("\n")
9  print ("Hello " + YourName + "!")
10 print "Your Body Mass Index is %4.1f" % ( BMI )
11
```

hypot.py

hypot.py

```
1  # hypot.py
2
3  from math import sqrt
4
5  def myhypot(x,y):
6      return sqrt(x ** 2 + y ** 2)
7
8  def main():
9      a = float(input("a: "))
10     b = float(input("b: "))
11     print "Hypotenous: ", myhypot(a,b)
12
13 main()
```

- Linie kodu programu które nie są zawarte wewnątrz funkcji, są wykonywane bezpośrednio gdy program zostaje uruchomiony (BMI.py)

- W skrypcie hypot.py tylko wywołanie funkcji main() - linia 13 - jest na zewnątrz wszystkich definicji funkcji. Funkcja main() zostanie wykonana po uruchomieniu skryptu. Funkcja ta „steruje” wykonywaniem całego programu. Zazwyczaj definicja funkcji main() oraz jej wywołanie znajduje się na końcu skryptów.

Definicja funkcji

1. **Definicja funkcji** określa kod programu który zostanie wykonany, gdy funkcja zostanie **wywołana**
2. Składnia definicji

```
def   <funkcja>(<parametry>):
.     <ciało>
```

3. Na końcu linii definicji znajduje się **dwukropek**
4. Dwukropek sygnalizuje, że nastąpi blok kodu **rozpoczęty od wcięcia – ciało funkcji**
5. Wcięcie zawsze musi być tych samych rozmiarów (np. 4 spacje, 1 tabulator)
6. Ciało funkcji zawiera linie kodu, które są wykonywane gdy funkcja jest wywołana. Może zawierać dowolną liczbę linii kodu (w myhypot() jest to lina 6)
7. **Parametry funkcji** umożliwiają podanie dodatkowej informacji , aby funkcja mogła odpowiednio wykonywać zaplanowane operacje (w myhypot() są to x i y). Argumenty są opcjonalne (funkcja może nie posiadać żadnych argumentów).

Anatomia funkcji

Słowo kluczowe **def,** oznacza rozpoczęcie definicji funkcji.

Parametry funkcji oddzielone przecinkiem

Dwukropek (:), kończy linię definicji

```
def dodaj(arg0, arg1):
    """ Dokumentacja (opis) funkcji """

    wynik = arg0 + arg1
    return a
```

Wcięcie. Wskazuje zawartość (ciało) funkcji. Jest wymagane (konieczne), ponieważ jest częścią definicji

Opcjonalne wyrażenie **return** określa wartość (wartości) zwracaną przez funkcję. Gdy to wyrażenie jest pominięte funkcja zwraca wartość **None**.

Opcjonalny opis działania funkcji (docstring) . Wyświetlany po wpisaniu komendy help()

Funkcje

8. Zazwyczaj definicje funkcji umieszczane są w górnej części skryptów, zaraz po wyrażeniach **import**
9. Funkcja która została zdefiniowana **może być wywołana**, tzn. wykonana z odpowiednimi **argumentami** przekazanymi do funkcji jako jej parametry.
10. Wywołanie funkcji:

```
<funkcja>(<argumenty>)
```

Gdy w programie pojawi się to wyrażenie funkcja jest wykonywana(!)

11. Nazwy argumentów **nie muszą** być takie same jak nazwy parametrów funkcji (!)

Dokończ zdania:

Funkcja myhypot() jest wykonywana w linii nr ……

Argumenty przesyłane do funkcji to wartości zmiennych ….. i ……

Podczas wykonania funkcji wartość …. zostanie przypisana do **x**, a wartość …. do **y**.

Funkcje

12. Wyrażenie **return** może pojawić się w dowolnym miejscu funkcji. Po wystąpieniu słowa kluczowego **return** kolejne linie ciała funkcji nie są już wykonywane. Funkcja kończy swoje działanie i zwraca wartość **<wyrażenie>** jako wartość wywołania funkcji.

return <wyrażenie>

Gdy na wyjściu znajduje się samo słowo kluczowe **return** (beż żadnego wyrażenia) to funkcja zwraca wartość **None**.

13. Zmienne zdefiniowane wewnątrz funkcji (parametry funkcji **x** i **y** w myhypot()) są **zmiennymi lokalnymi**. Zakres ich użycia jest ograniczony **tylko** do wnętrza funkcji myhypot(). Po wykonaniu funkcji, zmienne te są kasowane.

Ćwiczenia 6.1 – 6.5

Testowanie funkcji

1. Testy wszystkich ważnych typów i struktur danych parametrów wejściowych : wartości dodatnie, ujemne, liczby całkowite, zmiennoprzecinkowe…

2. Sprawdzanie przewidywalnych przypadków

3. Sprawdzenie wartości granicznych (wartości minimalne, maksymalne)

Ćwiczenie 6.6 – 6.7

Odczytywanie błędów

```
C:\Users\dell\Desktop\funkcje.py in <module>()
     48     #3## bledy
     49     dodaj61('ala',6)
---> 50 main()

C:\Users\dell\Desktop\funkcje.py in main()
     47     dodaj61([1,2,3], [6,7,8])
     48     #3## bledy
---> 49     dodaj61('ala',6)
     50 main()

C:\Users\dell\Desktop\funkcje.py in dodaj61(arg0, arg1)
      3
      4 def dodaj61(arg0, arg1):
----> 5     suma = arg0 + arg1
      6     print 'Suma dwoch cyfr wynosi:', suma
      7

TypeError: cannot concatenate 'str' and 'int' objects

In [4]:
```

Ćwiczenie 6.7 – 6.8

suma = arg0 + arg1

Type error: cannot concatenate 'str' and 'int' objects

The "type" of the error that occurred

Short message about why it occurred

Dokumentacja funkcji

```
def dodaj(arg0, arg1):
    """ Dokumentacja (opis) funkcji """

    wynik = arg0 + arg1
    return a
```

Opcjonalny opis działania funkcji (docstring) . Wyświetlany po wpisaniu komendy help()

Ćwiczenie:

Rozwiń opis funkcji aby zawierał co najmniej 4 linie.

Możesz dodać swoje imię i nazwisko, datę,….

Dokumentacja funkcji

Aby wyświetlić dokumentację danej funkcji **dodaj()** wpisz poniższe komendy. Czym się różnią otrzymane wyniki?

- dodaj**?**
- dodaj**??**
- help(dodaj)
- dodaj**(**

Wyświetl dokumentację następujących funkcji:

- range()
- len()
- type()
- squeeze()
- arange()

Domyślne parametry funkcji

Definicja funkcji:

```
def dodaj(arg0, arg1 = 7):
    suma = arg0 + arg1
    print 'Suma dwoch cyfr wynosi:', suma
```

Wywołanie funkcji:

```
# arg0=6 arg1=7
dodaj (6)
```

Definicja funkcji:

```
def dodaj(arg0=5, arg1 = 7):
    suma = arg0 + arg1
    print 'Suma dwoch cyfr wynosi:', suma
```

Wywołanie funkcji:

```
# arg0=5 arg1=7
dodaj ()
```

Wywołanie funkcji:

```
# arg0=9 arg1=8
dodaj (arg1= 8, arg0 = 9)
```

Ćwiczenia

6.1 Napisz funkcję **dodaj()** która dodaje dwie liczby **(arg0** i **arg1**), a następnie wypisuje wynik za pomocą polecenia **print**(). Funkcja nie zawiera wyrażenia **return**. Sprawdź jaką wartość zwraca funkcja.

6.2 Zmodyfikuj funkcję **dodaj()** aby nie wypisywała informacji o sumie liczb, ale żeby zwracała sumę liczb.

6.3 Zmodyfikuj funkcję **dodaj()** aby zwracała sumę liczb. Funkcja nie powinna posiadać żadnych dodatkowych zmiennych lokalnych (**return <wyrażenie>)**

6.4 Zmodyfikuj funkcję aby obliczała sumę i różnicę podanych liczb.

6.5 Sprawdź wartość zmiennych **arg0** i **arg1** po wyjściu z funkcji **dodaj()** np. w funkcji **main()**

Ćwiczenia

6.6 Dla funkcji **dodaj()** zaproponuj przykłady możliwych wartości testowych

6.7 Dla jakich argumentów wejściowych funkcja **dodaj()** nie działa?

6.8 Zaobserwuj informacje o błędach dla następujących przypadków:

- wypisanie niezdefiniowanej zmiennej: **print k**
- definiuj krotka listę **liczby=[1,3,5]** a następnie odczytaj wartość nieistniejącego indeksu np. 10**: liczby[10]**

Ćwiczenia

6.9 Napisz funkcję która dodaje liczbę całkowitą do listy. Lista posiada stałą długość np. 5 elementów, wszystkie elementy są typu int.

6.10 Napisz funkcję która dodaje liczbę całkowitą do listy. Lista posiada dowolną długość którą trzeba określić w algorytmie, wszystkie elementy są typu int.

6.11 Napisz funkcję która w zależności od wyboru użytkownika dodaje/odejmuje/mnoży/dzieli wszystkie liczby w liście przez podaną liczbę.

6.12 Napisz funkcję która liczy średnią arytmetyczną z elementów listy

6.13 Napisz funkcję sprawdza czy w liście znajduje się dana liczba (zwraca True lub False)

6.14 Rozwiń funkcję z 6.13 aby dodatkowo zwracała indeks tego elementu.

Literatura

Brian Heinold, Introduction to Programming Using Python, Mount St. Mary's University, 2012 (http://faculty.msmary.edu/heinold/python.html).

Brad Dayley, Python Phrasebook: Essential Code and Commands, SAMS Publishing, 2007 (dostępne też tłumaczenie: B. Dayley, Python. Rozmówki, Helion, 2007).

Mark J. Johnson, A Concise Introduction to Programming in Python, CRC Press, 2012.

11/18/2013 1:06 AM