Wrocław University of Science and Technology

Wstęp do programowania

INP003203L

Semestr zimowy 2020/2021

Poniedziałek, 7:30 - 9:00

sala wirtualna
– zajęcia online

Sylwia Majchrowska

sylwia.majchrowska@pwr.edu.pl

https://majsylw.netlify.app/teaching/

pokój 213, budynek L-1

Wrocław University of Science and Technology

Plan na dziś

1. Importowanie modułów
2. Wybrane funkcje z modułu **random**
3. Wybrane funkcje z modułu **math**
4. Tworzenie własnych modułów
5. Instalowanie i użytkowanie modułów spoza biblioteki standardowej

Wrocław University of Science and Technology

Funkcje

- samodzielne definiowanie funkcji

definicja funkcji suma()

```
def suma(a, b):
    s = a + b
    return s
```

dwukropek

nagłówek funkcji suma()

ciało funkcji suma()
(wydzielone przez 4 spacje lub 1 tabulator)

nazwa funkcji

argumenty

słowa kluczowe

zwracane wartości

wywołanie funkcji suma()

```
suma(2,3)
s1 = 4
s2 = 5
suma(s1, s2)
```

```
>>> type(suma)
<class 'function'>
```

```
>>> type(print)
<class 'builtin_function_or_method'>
```

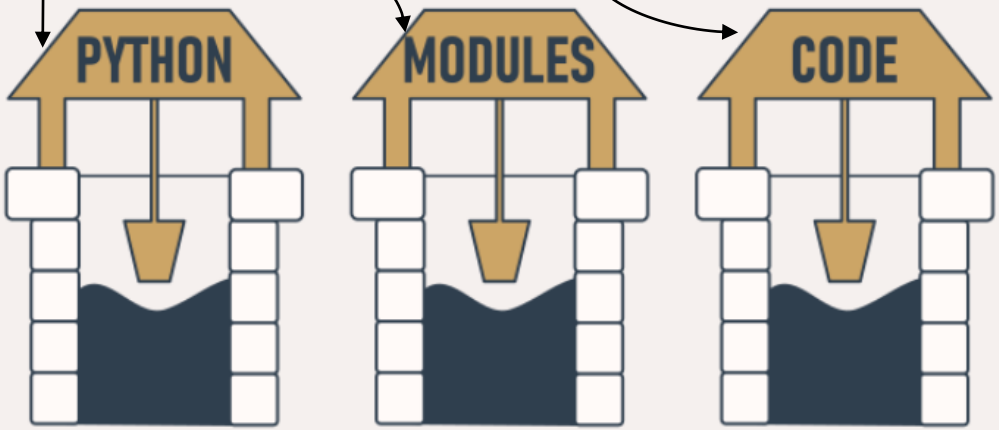
Funkcja to oddzielna część kodu komputerowego, która może:

- **wywołać jakiś efekt** (np. wysłać tekst do terminala, stworzyć plik, narysować obrazek, odtworzyć dźwięk itp.); jest to coś zupełnie niespotykanego w świecie matematyki;
- **obliczyć jakaś wartość** lub wartości (np. pierwiastek kwadratowy z wartości lub długość danego tekstu); to właśnie sprawia, że funkcje programistyczne są krewnymi pojęć matematycznych.

Wrocław University of Science and Technology

Skąd się biorą funkcje?

- funkcje wbudowane (ang. ***build-in functions***) - taka funkcja jest wartością dodaną otrzymywaną razem z interpreterem pythona i jego środowiskiem; nie musisz robić nic specjalnego (np. prosić kogokolwiek o cokolwiek), jeśli chcesz z niej skorzystać, np. funkcja print(), input(), type();
- mogą pochodzić z jednego lub więcej nazwanych modułów dodatkowych języka python; niektóre moduły są dostarczane z interpreterem, inne mogą wymagać oddzielnej instalacji - w każdym przypadku wszystkie muszą być jawnie połączone z Twoim kodem (**możesz także pisać własne moduły**);
- możesz napisać je samodzielnie, umieszczając w programie tyle funkcji, ile chcesz i potrzebujesz, aby kod był prostszy, bardziej przejrzysty i elegancki.



Korzyści z dzielenia programu na funkcje:

- Czytelniejszy kod
- Wielokrotne wykorzystanie kodu
- Lepsze testowanie (łatwiej testować podzadanie umieszczone w osobnej funkcji)
- Szybsze tworzenie oprogramowania
- Łatwiejsza praca w zespołach

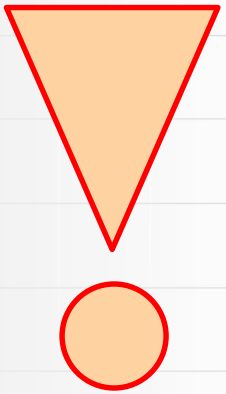
Wrocław University of Science and Technology

Moduły

Moduł to plik (najczęściej na nasze potrzeby będziemy korzystać z plików pythonowych czyli z rozszerzeniem *.py) zawierający definicje zmiennych i funkcji, będący pakietem dodatkowych narzędzi, które mogą zostać użyte po jego jawnym dodaniu do naszego kodu (zaimportowaniu).

Moduł to oddzielny zestaw instrukcji, który może przybrać formę:

- modułu standardowego, który pomaga w organizacji biblioteki standardowej i którego nie trzeba dodatkowo zainstalować, przykładowo: funkcje matematyczne przechowywane są razem w module **math**, funkcje obsługujące „losowość” razem w module **random** lub graficzne w module *turtle* (zadanie świąteczne),
- samodzielnie zdefiniowanego modułu, który jest oddzielnym plikiem przygotowywanym samodzielnie.
- modułu dodatkowego, którego przed użytkowaniem należy zainstalować np. korzystając z instrukcji pip lub conda, jeśli pracujemy w środowisku Anacondy.



Pamiętaj, że aby używać funkcji i zmiennych zdefiniowanych wewnątrz modułu musimy taki moduł najpierw zaimportować.

Wrocław University of Science and Technology

Importowanie modułów

- na przykładzie modułu math

Aby użyć zawartości modułu, należy go *zaimportować*. Dotyczy to również modułów ze standardowej biblioteki Pythona, od czego właśnie zaczniemy.

Instrukcja import pozwala na załączenie modułu

```
import math              # importujemy całą zawartość modułu math

print(math.sqrt(4))      # wyświetlamy wyliczony pierwiastek z 4 czyli 2
print(math.pow(2,3))     # wyświetlamy 2**3 czyli 8
```

kropka

Aby skorzystać z funkcji z zaimportowanego modułu wpisujemy <nazwa_zaimportowanego_modułu>.<nazwa_funkcji>

(1)

```
from math import sqrt
print(sqrt(4))
print(floor(2,3)) # ERROR
print(pow(2,3))# działa bo
# jest też taka funkcja wbudowana
```

(2)

```
from math import sqrt, pow
print(sqrt(4))
print(pow(2,3))
```

(3)

```
from math import *
print(pow(2,3))
```

Z modulu math importujemy tylko funkcję sqrt, dlatego w tym przypadku nie musimy wpisywać math.sqrt() – wystarczy samo wpisanie nazwy funkcji sqrt oraz podanie jej argumentów. Wywołanie linijki z funkcją floor skutkuje błędem gdyż, nie pobraliśmy jej do naszego programu (1).
Inaczej sytuacja wygląda w poniższym przypadku – nastąpił tu import dwóch funkcji pow i sqrt (2).
A jeszcze inaczej gdy pobieramy z danego modułu wszystko (3).

Wrocław University of Science and Technology

Importowanie modułów

- jak to działa?

Najpierw importujemy moduł (np. math) używając wyrażenia import <nazwa_modułu> (np. import math), czyli po prostu mówimy Pythonowi, że chcemy go używać. Moduł math zawiera funkcje i stałe matematyczne (więcej pod linkiem: https://docs.python.org/3.8/library/math.html).
Podczas wykonywania wyrażenia import math Python szuka pliku lub katalogu o odpowiedniej nazwie (m.in. zaczynającej się od math). W tym przypadku to moduł wbudowany, więc Python wie gdzie go znaleźć. Możliwości:

- katalog, w którym znajduje się wywoływany skrypt główny (lub bieżący katalog operacyjny),
- PYTHONPATH (lista katalogów w których Python poszukuje modułów – możesz ją podejrzeć w interpreterze poprzez wywołanie sys.path() po zaimportowaniu modułu sys),
- domyślne miejsce dla systemu – biblioteki wbudowane w interpreter (opcja dla biblioteki ***math***). https://stackoverflow.com/questions/18857355/where-are-math-py-and-sys-py

Aby operacja się powiodła, moduł musi być w jednym z podanych powyżej miejsc. Gdy moduł zostaje znaleziony, jego treść zostaje wykonana, a sam moduł staje się dostępny pod swoją nazwą. Gdy moduł nie zostanie odnaleziony pojawi się błąd (np. *No module named math*).

Inicjalizacja każdego modułu następuje tylko raz, podczas jego pierwszego użycia w danym programie.

Dodatkowo przy każdym imporcie możemy lokalnie zmienić nazwę modułu lub funkcji.

```
import math as m # stosujemy aliasowanie
print(m.sqrt(2)) # zamiast math wpisujemy m
                 # gdyż powyżej nadalismy modułowi math taką nazwę
```

Wrocław University of Science and Technology

Importowanie modułów

- pliki pośrednie .pyc

Importowanie modułu jest dość czasochłonną operacją. Z tego powodu Python tworzy pliki z rozszerzeniem *.pyc, które są pośrednią formą przetwarzania programu. W sytuacji gdy będziesz importował dany moduł kolejny raz jego wykonanie będzie o wiele szybsze, gdyż część pracy potrzebnej do zaimportowania została już wykonana. Moduły importowane są raz.

Uwaga!
Pliki .pyc są zazwyczaj tworzone w tym samym folderze, co odpowiednie pliki *.py (często towarzyszy temu utworzenie całego folderu `__*pycache*__`).

Każdy moduł posiada zmienną zawierającą jego nazwę (zazwyczaj). Najczęściej używa się tej zmiennej wtedy, gdy chcemy się dowiedzieć, czy moduł został zaimportowany, czy uruchomiony jako osobny program (wykorzystanie atrybutu __*name*__).

```
if __name__ == '__main__':
    print('This program is being run by itself')
else:
    print('I am being imported from another module')
```

Wrocław University of Science and Technology

Moduły standardowe

- na przykładzie math i random

https://docs.python.org/3/library/index.html

Dwa moduły matematyczne i numeryczne:

- math — funkcje matematyczne dla liczb rzeczywistych, np. sin, cos, exp…
- random — generowanie liczb pseudolosowych, np. random, uniform, seed…

```
>>> import math
# wyświetl nazwy atrybutów w module
>>> dir(math)
```

math	random
pi, e - znane i często używane stałe liczbowe **sin(x), tan(x), cos(x), atan(x), acos(x), asin(x)** - funkcje trygonometryczne z kąta x w radianach	**seed()** - ustawianie startowej wartości ‚zalążka' dla generatora liczb pseudolosowych
sqrt(x) - pierwiastek kwadratowy z liczby **pow(x, n)** - potęga x^n	**randrange(start, stop[, step])** - losowa liczba całkowita z kolekcji **range(start, stop[, step])**
log(x) - logarytm naturalny z liczby x **log10(x)** - logarytm dzisiętny z liczby x	**random()** - losowa liczba rzeczywista z przedziału [0.0, 1.0)
exp(x) - exponenta e^x	**uniform()** - losowa liczba rzeczywista z przedziału [a, b] lub [b, a]
floor(x) - podłoga z liczby x **ceil(x)** - sufit z liczby x	**randint(a, b)** - losowa liczba całkowita z przedziału [a, b]

Wrocław University of Science and Technology

Własne skrypty jako moduły

Tworzenie własnych modułów jest proste - każdy program w języku Python jest także modułem.

Moduł moj_modul.py

```
def powitanie():
    print('Mój moduł moj_modul')

__version__ = '0.1'
```

Skrypt główny main.py

```
import moj_modul

moj_modul.powitanie()
print('Version', moj_modul.__version__)
```

Pamiętaj, że moduł powinien być umieszczony w tym samym katalogu co program, który z niego korzysta, lub też w jednym z katalogów wpisanych w *sys.path*.

Uwaga: Pamiętaj aby unikać importowania za pomocą gwiazdki tzn. from mymodule import *.

Zen Pythona

Jednym z głównych zasad w Pythonie jest "Wyrażone wprost jest lepsze niż domniemane.". Uruchom import this aby dowiedzieć się więcej.

Wrocław University of Science and Technology

Paczki (ang. packages)

Hierarchia tworzonych bibliotek może być bardziej skomplikowana. Zmienne zazwyczaj znajdują się w funkcjach. Funkcje oraz zmienne globalne — w modułach. A moduły? Poszczególne moduły mogą być zapakowane w paczki.

Paczki to katalogi z modułami oraz ze specjalnym plikiem __init__.py (może być pusty lub mieć specjalnie sformułowaną strukturę), który informuje Pythona, że ten katalog jest przeznaczony właśnie do przechowywania modułów.

```
- <katalog>/
   - <podkatalog_1>/
      - __init__.py
      - modul_podkatalogu_1.py
   - <podkatalog_2>/
      - __init__.py
      - modul_podkatalogu_2.py
      - <podpodkatalog>/
         - __init__.py
         - modul_podpodkatalogu.py
```

nazwa paczki – głównego modułu

import <katalog>

specjalnym plik __init__.py

nazwa podkatalogu

import <katalog>. <podkatalog_1>

import <katalog>. <podkatalog_2>

nazwa podpodkatalogu

import <katalog>. <podkatalog_1>.<podpodkatalog>

import <katalog>. <podkatalog_1>.<podpodkatalog>.modul_podkatalogu

Wrocław University of Science and Technology

Instalacja modułów dodatkowych

Instalacja za pomocą pip (“PIP Installs Packages” lub “Preferred Installer Program”)

```
# instalacja modułu zewnetrzengo
pip install <nazwa_modułu>
# np. instalacja konkretnej wersji
pip install pandas==1.2.0

# szukanie informacji
# o zainstalowanym module
pip show <nazwa_modułu>

# szukanie informacji
# o module z bazy
pip search "nazwa_modułu"


# usuwanie modułu
pip uninstall <nazwa_modułu>
```

Instalacja za pomcą menadżera pakietów Conda

```
# instalacja modułu zewnetrzengo
conda install <nazwa_modułu>
# np. instalacja konkretnej wersji
conda install pandas=1.2.0

# szukanie informacji
# o zainstalowanym module
conda list <nazwa_modułu>

# szukanie informacji
# o module z bazy
conda search "nazwa_modułu"


# usuwanie modułu
conda uninstall <nazwa_modułu>
```

Pliki binarne dla systemu Windows.

Popularne paczki do analizy danych

Wrocław University of Science and Technology

Rzeczy do zapamiętania!

1. Moduł to plik (najczęściej pythonowy, ale nie tylko) zawierający definicje zmiennych i funkcji, będący pakietem dodatkowych narzędzi. Moduły mogą:
   - należeć do biblioteki standardowej,
   - być samodzielnie napisane przez użytkownika lub,
   - zostać dodatkowo zainstalowane.
2. Aby móc skorzystać z dowolnego modułu należy go wcześniej zaimportować używając instrukcji import. Importować możemy:
   - cały moduł (*import moduł*),
   - jedynie wybrane funkcje (*from moduł import funkcja1, funkcja2*),
   - wszystkie funkcje (*from moduł import* *),
   - a nawet aliasować je lokalnie (*import moduł as m*).
3. Biblioteka standardowa podzielona jest na szereg modułów w celu lepszej organizacji. Jednymi ze znanych modułów są:
   1. moduł **math**: funkcje matematyczne,
   2. moduł **random**: funkcje obsługujące „losowość”.
4. Warto własny kod dzielić na mniejsze moduły, aby łatwiej nim zarządzać – każdy skrypt pythonowy jest modułem.

Wrocław University of Science and Technology

Spojrzenie na kalendarz

		PAŹDZIERNIK					LISTOPAD					GRUDZIEŃ				STYCZEŃ				LUTY			
PN	28	5	12	19	26	2	9 K1	16	23	30 K2	7	14	21 K3	28	4	11	18	25 T1 Kolokwium	1 T2 Pn N	8	15	22
WT	29	6	13	20	27	3	10	17	24	1	8	15	22 Śr P	29	5	12	19	26	2	9	16	23
ŚR	30	7	14	21	28	4	11	18	25	2	9	16	23	30	6	13	20	27	3	10	17	24
CZ	1	8	15	22	29	5	12	19	26	3	10	17	24	31	7	14	21	28	4	11	18	25
PT	2 Pt N	9 Pt P	16	23	30	6	13 Śr P	20	27	4	11	18	25	1	8	15	22	29	5	12	19	26
SO	3	10	17	24	31	7	14	21	28	5	12	19	26	2	9	16	23	30	6	13	20	27
N	4	11	18	25	1	8	15	22	29	6	13	20	27	3	10	17	24	31	7	14	21	28
P - PARZYSTY N - NIEPARZYSTY	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N

Kolokwium = część pisemna (**90 minut na 3 zadania** w terminie **21-22.01.21**) + część ustna (ok. **10 minut rozmowy** na temat rozwiązanych zadań - **25.01.21r.**)

Poprawa = część pisemna (dla osób poprawiających na ocenę wyższą niż 3 lub poprawiających zaliczenie cząstkowe - do ustalenia 25.01.21r.) + część ustna (ok. 10 minut rozmowy - 1.02.21r.)

Poniedziałek, 7:30 - 9:00
sala wirtualna

Konsultacje:
Poniedziałki, 19:30 - 20:30
Środy, 19:30 – 20:30