Programowanie Python 1

(CP1S02005)

Politechnika Białostocka - Wydział Elektryczny

Cyfryzacja przemysłu, sem. II, studia stacjonarne I stopnia

Rok akademicki 2023/2024

Wykład nr 12 (29.05.2024)

dr inż. Jarosław Forenc

Plan wykładu nr 12

- Biblioteka standardowa Pythona
  - wbudowane funkcje
  - wbudowane stałe
  - wbudowane typy
  - wbudowane wyjątki
  - moduły (string, datetime, zoneinfo, calendar)

Python - biblioteka standardowa

- zbiór modułów i pakietów, które są dystrybuowane wraz z główną instalacją języka Python
- opis: https://docs.python.org/3/library/index.html (EN)
- biblioteka jest bardzo obszerna, zawiera wiele funkcji i modułów, które są przetestowane i wydajne
- biblioteka zawiera moduły:
  - wbudowane, napisane w języku C - dostęp do funkcjonalności systemowych, np. operacje na plikach
  - napisane w Pythonie - standardowe rozwiązania wielu problemów występujących podczas codziennego programowania
- w przypadku systemu Windows instalator Pythona zawiera całą bibliotekę standardową oraz wiele dodatkowych komponentów
- w systemach Linux/Unix, Python jest dostarczany jako zbiór pakietów, więc konieczne może być doinstalowanie niektórych komponentów

Python - biblioteka standardowa

- w skład biblioteki standardowej wchodzą:
  - wbudowane funkcje, wbudowane stałe
  - wbudowane typy, wbudowane wyjątki
  - moduły
- przykładowe przeznaczenie modułów:
  - usługi przetwarzania tekstu i danych binarnych, typy danych
  - moduły numeryczne i matematyczne
  - dostęp do plików i katalogów, kompresja i archiwizacja danych
  - usługi kryptograficzne, ogólne usługi systemu operacyjnego
  - sieci i komunikacja międzyprocesowa, obsługa danych internetowych
  - protokoły internetowe i wsparcie, usługi multimedialne
  - internacjonalizacja, narzędzia developerskie, importowanie modułów

Python - lista wbudowanych funkcji

A

abs()
aiter()
all()
anext()
any()
ascii()

B

bin()
bool()
breakpoint()
bytearray()
bytes()

C

callable()
chr()
classmethod()
compile()
complex()

D

delattr()
dict()
dir()
divmod()

E

enumerate()
eval()
exec()

F

filter()
float()
format()
frozenset()

G

getattr()
globals()

H

hasattr()
hash()
help()
hex()

I

id()
input()
int()
isinstance()
issubclass()
iter()

Python - lista wbudowanych funkcji

L

len()
list()
locals()

M

map()
max()
memoryview()
min()

N

next()

O

object()
oct()
open()
ord()

P

pow()
print()
property()

R

range()
repr()
reversed()
round()

S

set()
setattr()
slice()
sorted()
staticmethod()
str()
sum()
super()

T

tuple()
type()

V

vars()

Z

zip()

_

__import__()

Python - lista wbudowanych stałych

- True - reprezentuje wartość logiczną prawda (typ bool)
- False - reprezentuje wartość logiczną fałsz (typ bool)
- None - reprezentuje brak wartości lub wartość null, obiekt używany gdy np. domyślne argumenty nie są przekazywane do funkcji (typ NoneType)
- NotImplemented - specjalna wartość używana do wskazania, że metoda nie jest zaimplementowana dla określonych typów danych
- Ellipsis - reprezentowany przez ... (trzy kropki); używany głównie w specjalnych przypadkach, takich jak definiowanie zakresów lub wycinków w tablicach wielowymiarowych
- __debug__ - stała, która ma wartość True, jeśli Python działa w trybie debugowania
- copyright, credits, license - stałe informacyjne wyświetlające informacje o prawach autorskich, kredytach i licencji Pythona

Python - lista wbudowanych typów

- typy liczbowe:
  - int - liczby całkowite, dowolnie duże (bez limitu wielkości) , np. a = 23
  - float - liczby zmiennoprzecinkowe (rzeczywiste), jak double w C, np. b = 2.5
  - complex - liczby zespolone z częścią rzeczywistą i urojoną, np. z = 2 + 5j
- typy sekwencyjne:
  - list - lista, dynamiczna tablica elementów, które mogą być różnych typów, mutowalna - można zmieniać jej zawartość po utworzeniu, np. lst = [1, 2, 3]
  - tuple - krotka, niezmienna sekwencja elementów, po utworzeniu nie można zmieniać jej zawartości, np. kr = (1, 2, 3, 'x', 'y', 'z')
  - range - generuje sekwencję liczb całkowitych w sposób leniwy (liczby generowane są na bieżąco, a nie przechowywane w pamięci), np. z = range(100)
- typy tekstowe:
  - str - łańcuch znaków, czyli tekst; kodowanie znaków - Unicode, np. txt = "Witaj"

Python - lista wbudowanych typów

- typy binarne:
  - bytes - sekwencja bajtów (wartości od 0 do 255); typ niemutowalny - po utworzeniu nie można zmienić jego zawartości, np. b = b'witaj'
  - bytearray - sekwencja bajtów; typ mutowalny - można zmienić zawartość, np. ba = bytearray(b'witaj')
  - memoryview - umożliwia efektywne operacje na dużych obiektach binarnych bez potrzeby ich kopiowania, np. mv = memoryview(b'witaj')
- typy zbiorów (zestawów):
  - set - modyfikowalny zbiór unikalnych elementów, np. s = {1, 2}
  - frozenset - niemodyfikowalny zbiór unikalnych elementów, np. fs = frozenset([1, 2, 3, 'x', 'y', 'z'])
- typy mapowania:
  - dict - słownik, kolekcja par klucz-wartość, klucze są unikalne, każdemu kluczowi przypisana jest jedna wartość, np. d = {'k1': 'w1', 'k2': 'w2'}

Python - lista wbudowanych typów

- typy logiczne:
  - bool - reprezentuje wartości logiczne True i False, podtyp typu int (dziedziczy wszystkie jego właściwości), np. b = True
- typy specjalne:
  - NoneType - reprezentowany tylko przez jeden obiekt None, oznacza brak wartości lub nieistnienie, np. n = None
- typy wbudowane obsługujące iterację:
  - enumerate - funkcja zwracająca iterator, który produkuje krotki zawierające indeks i element z sekwencji, np. e = enumerate(['x', 'y', 'z'])
  - reversed - funkcja zwracająca iterator odwracający kolejność elementów, np. r = reversed([6, 7, 8])
  - zip - funkcja zwracająca iterator, który łączy elementy z kilku iterowalnych obiektów. np. z = zip([6, 7, 8], ['x', 'y', 'z'])

Python - lista wbudowanych typów

- typy funkcyjne:
  - function - funkcja zdefiniowana za pomocą słowa kluczowego def lub lambda, np. def funkcja(x): return x ** 2
  - lambda - funkcja anonimowa, z dowolną liczbą argumentów, ale tylko jednym wyrażeniem, np. f = lambda x: x ** 2
- typy klas i obiektów:
  - class - klasy zdefiniowane za pomocą słowa kluczowego class, np. class MyClass: pass
  - object - podstawowa klasa bazowa dla wszystkich klas, np. o = object()
- typy wyjątków
  - BaseException - podstawowa klasa dla wszystkich wyjątków
  - Exception - podstawowa klasa dla większości wyjątków, używanych w typowych sytuacjach błędów programistycznych

Python - lista wbudowanych wyjątków

- wyjątki (ang. exceptions) są to specjalne obiekty, którymi posługuje się Python podczas zarządzania błędami, które mogą pojawić się w trakcie wykonywania programu

```
try:
    # kod, który może generować wyjątek
except ExceptionType:
    # obsługa wyjątku
finally:
    # kod, który zostanie wykonany zawsze
```

```
try:
    # kod, który może generować wyjątek
except ExceptionType:
    # obsługa wyjątku
else:
    # kod, który zostanie wykonany tylko wtedy,
    # gdy nie wystąpił żaden wyjątek
```

Python - lista wbudowanych wyjątków

- Exception - podstawowa klasa wszystkich wyjątków w Pythonie, wbudowane wyjątki są pochodnymi tej klasy.
- BaseException - podstawowa klasa wszystkich wyjątków, w tym takich, które nie są błędami (nie należy używać jej bezpośrednio)
- KeyboardInterrupt - występuje, gdy użytkownik przerywa wykonywanie programu za pomocą skrótu klawiaturowego (Ctrl+C)
- SystemExit - występuje, gdy program kończy działanie poprzez wywołanie funkcji sys.exit(), przechwycenie zapobiega zamknięciu programu
- StopIteration - wywoływany przez iteratory, aby wskazać koniec iteracji (automatycznie obsługiwany przez pętle for)
- AttributeError - wywoływany przy próbie dostępu do nieistniejącego atrybutu obiektu
- EOFError - wywoływany, gdy funkcje wczytujące dane z wejścia (np. input() lub read()) napotkają koniec pliku (EOF)

Python - lista wbudowanych wyjątków

- ArithmeticError - klasa podstawowa dla wyjątków związanych z błędami arytmetycznymi:
  - ZeroDivisionError - występuje przy próbie dzielenia liczby przez zero
  - OverflowError - występuje, gdy wynik operacji arytmetycznej jest zbyt duży, aby można go było reprezentować
  - FloatingPointError - występuje przy błędach związanych z operacjami zmiennoprzecinkowymi
- ImportError - występuje, gdy nie uda się zaimportować modułu
  - ModuleNotFoundError - występuje, gdy moduł nie może zostać odnaleziony
- IndexError - występuje, gdy indeks sekwencji (np. listy) jest poza zakresem
- KeyError - występuje, gdy klucz nie zostanie odnaleziony w słowniku
- MemoryError - występuje, gdy nie ma wystarczającej ilości pamięci, aby zakończyć operację

Python - lista wbudowanych wyjątków

- NameError - występuje, gdy zmienna lub symbol nie zostały zdefiniowane
- OSError - podstawowa klasa wyjątków związanych z błędami systemowymi
  - FileNotFoundError - plik lub katalog nie mogą zostać odnalezione
  - PermissionError - brak uprawnień do wykonania operacji
  - IsADirectoryError - operacja wymaga pliku, ale obiekt jest katalogiem
  - NotADirectoryError - operacja wymaga katalogu, ale obiekt nie jest katalogiem
- TypeError - występuje, gdy operacja lub funkcja jest wykonywana na obiekcie niewłaściwego typu
- ValueError - występuje, gdy funkcja otrzymuje argument o poprawnym typie, ale o niewłaściwej wartości

Python - moduł string

- moduł string zawiera różne funkcje i stałe, przydatne podczas wykonywania operacji na łańcuchach znaków
- lista stałych:
  - string.ascii_letters - zawiera wszystkie litery alfabetu (małe i duże)

```
import string
print(string.ascii_letters)
```

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
```

  - string.ascii_lowercase - zawiera wszystkie małe litery alfabetu
  - string.ascii_uppercase - zawiera wszystkie wielkie litery alfabetu
  - string.digits - zawiera wszystkie cyfry dziesiętne (0-9)
  - string.hexdigits - zawiera wszystkie cyfry szesnastkowe (0-9, a-f, A-F)
  - string.octdigits - zawiera wszystkie cyfry ósemkowe (0-7)

Python - moduł string

- lista stałych:
  - string.punctuation - zawiera wszystkie znaki interpunkcyjne
  - string.printable - zawiera wszystkie znaki, które są uznawane za "drukowalne", czyli litery, cyfry, znaki interpunkcyjne i białe znaki
  - string.whitespace - zawiera białe znaki (spacje, tabulatory, nowe linie itp.)

```
import string
txt = """W lutym 2024 r. powstalo 11 567 sztuk nowych
instalacji pv. Ich laczna moc wyniosla 232,50 MW.
Najwieksza grupe stanowia mikroinstalacje: 11 478
sztuk o lacznej mocy 100,33 MW."""

cyfry = [zn for zn in txt if zn in string.digits]
print(f"Liczba cyfr w tekście: {len(cyfry)}")
```

```
Liczba cyfr w tekście: 24
```

Python - moduł datetime

- moduł datetime jest używany do manipulowania datami i czasem, zawiera wiele funkcji i klas, które ułatwiają operacje na datach, takie jak tworzenie, porównywanie, formatowanie i wykonywanie operacji arytmetycznych
- klasy znajdujące się w module:
  - datetime - reprezentuje datę i czas jako pojedynczy obiekt
  - date - reprezentuje tylko datę (bez czasu)
  - time - reprezentuje tylko czas (bez daty)
  - timedelta - reprezentuje różnicę pomiędzy dwoma datami lub czasami
- klasy umożliwiają wykonywanie operacji arytmetycznych na datach i czasach, takich jak dodawanie i odejmowanie
- obiekty klas mogą być porównywane ze sobą za pomocą operatorów porównania, takich jak <, <=, >, >=, ==, !=
- obiekty posiadają metody umożliwiające dostęp do składowych daty i czasu, takich jak rok, miesiąc, dzień, godzina, minuta, sekunda

Python - moduł datetime

```
import datetime

dowolna_data = datetime.datetime(2024, 5, 1, 12, 0, 0)
biezaca_data = datetime.datetime.now()

print("Dowolna data:")
print("Rok:", dowolna_data.year)
print("Miesiąc:", dowolna_data.month)
print("Dzień:", dowolna_data.day)

print("\nTeraz jest:")
print("Godzina:", biezaca_data.hour)
print("Minuta:", biezaca_data.minute)
print("Sekunda:", biezaca_data.second)
```

```
Dowolna data:
Rok: 2024
Miesiąc: 5
Dzień: 1

Teraz jest:
Godzina: 10
Minuta: 14
Sekunda: 20
```

Python - moduł zoneinfo

- klasa zoneinfo umożliwia obsługę stref czasowych (dostępna od wersji 3.9)
- pozwala tworzyć obiekty reprezentujące konkretne strefy czasowe, takie jak "Europe/Warsaw", "America/New_York", "Asia/Tokyo", itp.
- obiekty zoneinfo zawierają informacje o przesunięciach czasowych, historii zmian czasu letniego/zimowego i innych ustawieniach specyficznych dla danej strefy czasowej
- klasa zoneinfo umożliwia wykonywanie operacji na strefach czasowych, takich jak konwersja daty i czasu między różnymi strefami czasowymi, określanie czy dana data i czas należy do określonej strefy czasowej, itp.
- obiekty zoneinfo automatycznie uwzględniają zmiany czasu letniego/zimowego

Python - moduł calendar

- moduł calendar umożliwia generowanie kalendarzy oraz wykonywanie operacji związanych z datami
- pozwala na tworzenie kalendarzy dla różnych lat i miesięcy, dostęp do informacji o dniach tygodnia, daty, a także wykonywanie różnych operacji arytmetycznych na datach
- moduł calendar zawiera funkcje do generowania kalendarzy dla określonego roku i miesiąca, takie jak calendar.month() i calendar.calendar()
- klasa TextCalendar umożliwia generowanie kalendarzy w postaci tekstu, można dostosować sposób wyświetlania kalendarza, takie jak wybór pierwszego dnia tygodnia i formatowanie tekstu
- moduł zawiera funkcje pomocnicze, np. calendar.weekday() - zwraca dzień tygodnia dla określonej daty, calendar.monthrange() - zwraca pierwszy dzień tygodnia oraz liczbę dni w danym miesiącu
- moduł calendar zawiera również stałe, takie jak calendar.MONDAY, calendar.TUESDAY, itd., które reprezentują dni tygodnia

Python - moduł calendar

```
import calendar

print("Kalendarz miesiąca stycznia 2024:")
print(calendar.month(2024, 1))
```

```
Kalendarz miesiąca stycznia 2024:
    January 2024
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

Python - moduł calendar

```
import calendar

print("Numer dnia tygodnia dla 29 maja 2024 roku:")
print(calendar.weekday(2024, 5, 29))

print("Początkowy dzień tygodnia i liczba dni w styczniu
2024 roku:")
print(calendar.monthrange(2024, 1))

print("Stałe reprezentujące dni tygodnia:")
print("Poniedziałek:", calendar.MONDAY)
print("Wtorek:", calendar.TUESDAY)
```

```
Numer dnia tygodnia dla 29 maja 2024 roku:
2
Początkowy dzień tygodnia i liczba dni w styczniu 2024 roku:
(calendar.MONDAY, 31)
Stałe reprezentujące dni tygodnia:
Poniedziałek: 0
Wtorek: 1
```

Python - moduł calendar

```
import calendar

print(calendar.calendar(2024))
```

```
                                  2024

      January                   February                  March
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7                1  2  3  4                   1  2  3
 8  9 10 11 12 13 14       5  6  7  8  9 10 11       4  5  6  7  8  9 10
15 16 17 18 19 20 21      12 13 14 15 16 17 18      11 12 13 14 15 16 17
22 23 24 25 26 27 28      19 20 21 22 23 24 25      18 19 20 21 22 23 24
29 30 31                  26 27 28 29               25 26 27 28 29 30 31

      April                     May                       June
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7             1  2  3  4  5                1  2
 8  9 10 11 12 13 14       6  7  8  9 10 11 12       3  4  5  6  7  8  9
15 16 17 18 19 20 21      13 14 15 16 17 18 19      10 11 12 13 14 15 16
22 23 24 25 26 27 28      20 21 22 23 24 25 26      17 18 19 20 21 22 23
29 30                     27 28 29 30 31            24 25 26 27 28 29 30
```

Koniec wykładu nr 12

Dziękuję za uwagę!