

Gniewomir Sarbicki



Python

Kurs dla nauczycieli i studentów

Wydanie II

Helion

Wydawnictwo
Naukowe
Helion

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich.

Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite / Olsztyn

Obarek, Pokoński, Pazdrijowski, Zaprucki

Grafika na okładce została wykorzystana za zgodą Shutterstock.com

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/pytkn2>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-8239-8

Copyright © Helion S.A. 2022

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Wstęp	8
1 Wprowadzenie	11
1.1 Interaktywna powłoka, interpreter skryptów, edytory i środowiska	11
1.2 Typy liczbowe	14
1.3 Typy sekwencyjne	16
1.3.1 Łańcuchy znaków i łańcuchy bajtów	17
1.3.2 Listy	20
1.3.3 Krotki	22
1.4 Instrukcje warunkowe	22
1.4.1 Trójargumentowy operator logiczny	23
1.4.2 Kwantyfikatory	23
1.5 Pętle for i while	23
1.6 Listy składane	25
1.7 Wyrażenia przypisania	26
1.8 Słowniki	27
1.9 Funkcje	29
1.9.1 Zmienna liczba argumentów	32
1.9.2 Dokumentacja funkcji	33
1.9.3 Adnotacje funkcji	33
1.9.4 Zmienne globalne w funkcjach	34
1.9.5 Funkcje anonimowe	35
1.10 Programowanie funkcyjne	35
1.11 Formatowanie łańcuchów	38
1.11.1 Formatowanie z użyciem operatora %	39
1.11.2 Formatowanie z użyciem metody format	40
1.11.3 f-łańcuchy	41

1.12	Importowanie modułów	42
1.13	Funkcje matematyczne i liczby pseudolosowe	44
1.14	Pobieranie argumentów ze standardowego wejścia	45
1.15	Pobieranie argumentów z linii poleceń. Tworzenie aplikacji konsolowych.	45
1.16	Obsługa wyjątków	46
1.17	Praca z plikami	47
1.18	Porównywanie wydajności rozwiązań	50
1.19	Data i czas	51
1.20	Serializacja*	52
1.21	Współpraca z systemem operacyjnym	53
1.22	Dostęp do zasobów WWW	55
2	Programowanie obiektowe	57
2.1	Klasy i instancje, atrybuty i metody	57
2.2	Konstruktor klasy	60
2.3	Dziedziczenie i przysłanianie	60
2.4	Przeciążanie operatorów	61
2.5	Wywoływanie wyjątków	67
3	Graficzny interfejs użytkownika	70
3.1	Pierwszy program w GTK	70
3.2	Umieszczanie w oknie jego obiektów składowych	72
3.3	Obsługa zdarzeń	75
3.4	Metody elementów okna	79
4	Wielowątkowość	82
4.1	Pierwszy program wielowątkowy	83
4.2	Blokady	84
4.3	Porównanie wydajności	86
4.4	Kolejki	89

5	Komunikacja sieciowa	91
5.1	Pierwszy program	92
5.2	Serwer wielowątkowy	95
5.3	Serwer dyskusyjny	96
5.4	Klient usługi TCP*	99
5.5	Serwer i klient UDP*	99
6	Obsługa baz danych	101
6.1	SQLite	101
6.1.1	Dostęp do bazy z linii poleceń	101
6.1.2	Polecenia SQL w SQLite	102
6.1.3	Moduł sqlite3	102
6.2	MySQL*	105
6.2.1	Dostęp do serwera z linii poleceń i tworzenie kont użytkowników	105
6.2.2	Polecenia SQL w MySQL	106
6.2.3	Moduł mysql.connector	107
6.3	Ćwiczenia	108
7	Skrypty CGI	110
7.1	Aplikacje WWW korzystające z bazy danych	114
8	Obliczenia numeryczne	129
8.1	Tablice jednowymiarowe	129
8.2	Wykresy funkcji jednej zmiennej	132
8.3	Tablice wielowymiarowe	137
8.4	Wykresy trójwymiarowe	139
8.5	Pola wektorowe	146
8.6	Wykresy animowane	147
8.7	Równania różniczkowe zwyczajne	149
8.8	Równania różniczkowe cząstkowe	157
9	Funkcje wyższych rzędów	164
9.1	Dekoratory funkcji	165
9.2	Atrybuty funkcji	171

9.3	Dekoratory jako klasy	172
9.4	Dekoratory klas	173
9.5	Menedżery kontekstu	179
10	Iteratory, generatory, koprocedury	182
10.1	Funkcje generatorów	185
10.2	Wyrażenia generatorów i odwzorowywanie generatorów	186
10.3	Działania na iteratorach	187
10.4	Menedżery kontekstu z generatorów	192
10.5	Koprocedury	192
10.6	Obsługa wyjątków w generatorze	194
10.7	Algorytm roju cząstek realizowany przez mikrowątki	197
10.8	Nieblokujące operacje wejścia-wyjścia	201
11	Programowanie asynchroniczne	204
11.1	Obiekty oczekiwalne	204
11.2	Współbieżność zadań	207
11.3	Asynchroniczne odpowiedniki obiektów i konstrukcji składniowych Pythona	214
11.4	Asynchroniczny serwer TCP	217
12	Zarządzanie atrybutami w klasach	220
12.1	Niskopoziomowe zarządzanie atrybutami	220
12.2	Właściwości	225
12.3	Deskryptory	228
12.4	Metody statyczne i metody klas	230
12.5	Atrybut <code>__slots__</code> klasy	233
13	Współbieżność wykorzystująca podprocesy	235
13.1	Operacje na tablicach NumPy	239
13.2	Puła podprocesów	242
13.3	Obiekt podprocesu	243
13.4	Komunikacja międzyprocesowa	244
13.5	Synchronizacja podprocesów	249

14 Rozwiązania	261
Rozdział 1.	261
Rozdział 2.	271
Rozdział 3.	286
Rozdział 4.	298
Rozdział 5.	299
Rozdział 6.	306
Rozdział 7.	311
Rozdział 8.	329
Rozdział 9.	345
Rozdział 10.	365
Rozdział 11.	393
Rozdział 12.	396
Rozdział 13.	401
Skorowidz	408

Rozdział 3.

Graficzny interfejs użytkownika

Do zbudowania graficznego interfejsu użytkownika (GUI) można użyć wielu bibliotek, z których najbardziej popularne to GTK, Qt, wx i pythonowy tkinter. Ze względu na prostotę pisania kodu wybierzemy bibliotekę GTK, dostarczaną przez pakiet `gi`. Pakiet ten nie jest częścią biblioteki standardowej i musimy go doinstalować.

3.1 Pierwszy program w GTK

Importujemy moduł `Gtk`:

```
1 from gi.repository import Gtk
```

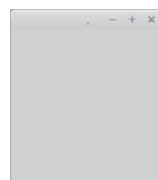
Moduł ten dostarcza nam obiekt `Window`, który reprezentuje główne okno programu. By obiekt z modułu `Gtk` stał się widoczny, wywołujemy jego metodę `show`.

Aplikacje okienkowe cały czas oczekują na zdarzenie wywołane przez użytkownika (przesuwanie i skalowanie okna, naciśnięcie przycisków w oknie, przeciąganie i opuszczanie obiektów itp.). Żeby uruchomić obsługę zdarzeń dla okna, należy wystartować *główną pętlę zdarzeń* poleceniem `Gtk.main()`.

Poniższy program zwróci widoczne okienko programu:

```
1 from gi.repository import Gtk
2
3 o=Gtk.Window()
4 o.show()
5 Gtk.main()
```

→



Ponieważ okno będzie zawierało wiele innych obiektów `Gtk` jako swoje elementy składowe oraz kilka funkcji obsługujących te obiekty, będziemy cały kod obsługujący okno pakować w klasę. Powyższy kod przyjmie postać:

```
1 from gi.repository import Gtk
2
3
4 class Okno:
5
6     def __init__(self):
7         self.o=Gtk.Window()
8         self.o.show()
9
10
11 Okno()
12 Gtk.main()
```

Wszystkie metody klasy `Gtk.Window` zwróci polecenie `dir(Gtk.Window)`. Do użytku w dalszej części kursu wybierzmy kilka z nich:

- `set_title` — ustawia tytuł okna.
- `move` — ustawia pozycję okna od lewego górnego rogu ekranu wg krotki `(x,y)`.
- `resize` — ustawia rozmiar okna wg krotki `(x,y)`.
- `get_title` — pobiera tytuł okna.
- `get_position` — pobiera bieżącą pozycję okna liczoną od lewego górnego rogu ekranu.
- `get_size` — pobiera bieżący rozmiar okna do krotki.

Działanie powyższych metod ilustruje poniższy kod:

```
1 from gi.repository import Gtk
2
3
4 class Okno:
5
6     def __init__(self):
7         self.o=Gtk.Window()
8         self.o.show()
9
10
11 ok=Okno()
12
13 ok.o.set_title("Moje Okno")
```

```
14 ok.o.move(500,500)
15 ok.o.resize(400,400)
16
17 input()
18
19 print(ok.o.get_title())
20 print(ok.o.get_position())
21 print(ok.o.get_size())
22
23 Gtk.main()
```

Kod pod nazwą metody_okna.py w archiwum <https://ftp.helion.pl/przyklady/pytkn2.zip>.

Pozycja i rozmiar okna uaktualniają się po wystąpieniu jakiegoś zdarzenia, stąd oczekiwanie na naciśnięcie przycisku realizowane przez `input`.

3.2 Umieszczanie w oknie jego obiektów składowych

Spośród wielu elementów graficznych, które może zawierać okno aplikacji, wybierzmy cztery:

- **Button** — przycisk. Łańcuch, który stanie się podpisem przycisku, przekazujemy w argumencie nazwanym `label`.
- **TextView** — pole, w którym wyświetlamy tekst.
- **Entry** — jednolinijkowe pole wprowadzania tekstu.
- **Label** — Etykieta tekstowa z nieedytowalnym tekstem, przekazywanym w argumencie nazwanym `label`.

Obiekt dodajemy do okna, podając go jako argument do metody `add` okna. Żeby stał się widoczny, musi być wywołana jego metoda `show`:

```
1 from gi.repository import Gtk
2
3
4 class Okno:
5
6     def __init__(self):
7         self.o=Gtk.Window()
8         self.o.set_title('')
9         self.b=Gtk.Button(label='OK')
10        self.o.add(self.b)
```

```

1     self.o.show()
2     self.b.show()
3
4
5 ok=Okno()
6 Gtk.main()

```



Spróbujmy dodać kolejny obiekt. Otrzymamy informację, że okno może zawierać tylko jeden obiekt. Więcej obiektów uzyskujemy, pakując je w pudełkach pionowych: `Gtk.VBox` lub poziomych: `Gtk.HBox`. Mają one metodę `pack_start`, pozwalającą dodawać kolejne elementy do końca pudełka (od dołu dla `VBox` i od prawej dla `HBox`), oraz metodę `pack_end`, pozwalającą dodawać kolejne elementy do początku pudełka (od góry dla `VBox` i od lewej dla `HBox`).

Wywołując metody `pack_start` i `pack_end`, musimy podać dodatkowe argumenty:

- `expand` — jeżeli jest `False`, pakowany obiekt ma przydzieloną minimalną szerokość (dla `HBox`) lub wysokość (dla `VBox`). Dodatkowa przestrzeń jest dzielona po równo pomiędzy elementy, dla których `expand=True`.
- `fill` — czy element powinien wypełniać całą przydzieloną mu przestrzeń. Jeżeli `expand=False`, argument ten nie ma znaczenia.
- `padding` — szerokość dodatkowych marginesów bocznych (dla `HBox`) lub pionowych (dla `VBox`).

Jeżeli oba elementy w pudełku mają wartość `expand=True`, to otrzymujemy:

```

1 from gi.repository import Gtk
2
3
4 class Okno:
5
6     def __init__(self):
7         self.o=Gtk.Window()
8         self.o.set_title('')
9         self.o.resize(400,100)
10        self.h=Gtk.HBox()
11        self.b=Gtk.Button(label='1')
12        self.e=Gtk.Button(label='2')
13        self.h.pack_start(self.b,True,True,0)
14        self.h.pack_start(self.e,True,True,0)
15        self.o.add(self.h)
16        self.o.show()
17        self.b.show()
18        self.e.show()
19        self.h.show()

```

```

20
21 ok=Okno()
22 Gtk.main()

```



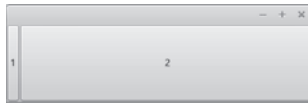
Kod pod nazwą `gtk_expand.py` w archiwum <https://ftp.helion.pl/przyklady/pytkn2.zip>.

Jeżeli dla jednego elementu `expand=False`, to otrzymujemy:

```

1 self.h.pack_start(self.b,False,True,0)
2 self.h.pack_start(self.e,True,True,0)

```

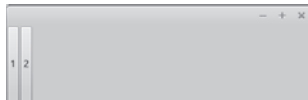


Jeżeli dla obu elementów `expand=False`, to otrzymujemy:

```

1 self.h.pack_start(self.b,False,True,0)
2 self.h.pack_start(self.e,False,True,0)

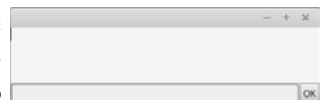
```



Obiekty pakowane z `expand=False` zachowają swoją oryginalną szerokość (wysokość dla `VBox`), a dodatkowa przestrzeń zostanie równo podzielona pomiędzy te obiekty, dla których `expand=True`.

Zamiast wywoływać metodę `show` na każdym elemencie okna, można wywołać jego metodę `show_all`.

Ćwiczenie 58. Stwórz okno pokazane na rysunku obok. Skalowanie pionowe nie powinno zmieniać wysokości dolnego paska, a skalowanie poziome nie powinno zmieniać szerokości przycisku.



3.3 Obsługa zdarzeń

Aplikacja okienkowa powinna oczekiwać na zdarzenia wywołane przez użytkownika i na nie reagować. W momencie zdarzenia (np. przesunięcia okna, naciśnięcia przycisku) obiekt emituje sygnał informujący o zdarzeniu. Sygnał ten możemy przechwycić dzięki metodzie `connect` tego obiektu, przekazując jej jako argumenty nazwę sygnału i funkcję obsługującą zdarzenie:

```
1 obiekt.connect('nazwa_sygnału',
    ↪ funkcja_obsługujaca_zdarzenie)
```

Poniżej zostaną opisane możliwe zdarzenia i sygnały, które są wtedy emitowane.

Zamknięcie okna — sygnał `'delete-event'` obiektu `Gtk.Window()`. Sygnał ten jest emitowany, gdy naciśniemy przycisk zamknięcia okna (lub zastosujemy skrót `Alt+F4`). Zauważmy, że gdy zamykamy okno, program dalej działa — nie nie przerywa pętli `Gtk.main()`. Żeby ją przerwać, należy wywołać funkcję `Gtk.main_quit`. ↪ `main_quit`. Połączmy ją w metodzie `connect` z sygnałem `'delete-event'`, dodając w konstruktorze linię:

```
1 self.o.connect("delete-event", Gtk.main_quit)
```

Teraz zamknięcie okna powoduje również zakończenie programu. Możemy zablokować domyślną obsługę tego zdarzenia (zamknięcia okna), jeżeli funkcja obsługi zdarzenia będzie zwracać 1:

```
1 self.o.connect("delete-event", lambda *args: 1)
```

(Ponieważ nie wiemy, ile argumentów jest przekazywanych do funkcji obsługi zdarzenia, tworzymy funkcję o zmiennej liczbie argumentów). Możemy też stworzyć złośliwy kod:

```
1 self.o.connect("delete-event", lambda *args: [Okno()] * 5)
```

Ćwiczenie 59. *Sprawdź, jakie argumenty otrzymuje funkcja obsługi zdarzenia `'delete-event'`.*

Z poziomu programu zamknąć okno możemy wywołując jego metodę `close`.

Naciśnięcie przycisku myszy — sygnał `'button-press-event'`. Sygnał jest emitowany przez elementy składowe okna w momencie naciśnięcia przycisku myszy na ich obszarze. Wypiszmy, jakie argumenty są przekazywane do funkcji obsługi zdarzenia:

```
1 from gi.repository import Gtk
```

```
2
```

```

3
4 class Okno:
5
6     def __init__(self):
7         self.o=Gtk.Window()
8         self.o.set_title('')
9         self.o.resize(400,100)
10        self.h=Gtk.HBox()
11        self.b=Gtk.Button(label='Pierwszy')
12        self.e=Gtk.Button(label='Drugi')
13        self.h.pack_start(self.b,True,True,0)
14        self.h.pack_start(self.e,True,True,0)
15        self.o.add(self.h)
16        self.o.show_all()
17        self.b.connect("button-press-event",self.b_press)
18        self.e.connect("button-press-event",self.b_press)
19
20    def b_press(*args):
21        print(args)
22
23
24 ok=Okno()
25 Gtk.main()

```



```

(<__main__.Okno object at 0x7f6d451898d0>, <Button object
  ↳ at 0x7f6d47ffb410 (GtkButton at 0x1845150)>, <void at
  ↳ 0x17d7c00>)

```

Kod pod nazwą `gtk_bpe.py` w archiwum
<https://ftp.helion.pl/przyklady/pytkn2.zip>.

Pierwszym argumentem jest `self` (pamiętamy, że jako pierwszy argument jest przekazywana instancja wywołująca metodę), drugim jest obiekt, który wyemitował zdarzenie. Trzecim jest obiekt zdarzenia. Żeby sprawdzić, jakie atrybuty ma zdarzenie, przeprowadźmy modyfikację: `print(args) → print(dir(↳ args[2]))`. Otrzymamy:

```

1 ['__class__', '__delattr__', '__dict__', '__doc__', '__eq__
  ↳ ', '__format__', '__ge__', '__getattr__', '__
  ↳ __gt__', '__gtype__', '__hash__', '__info__', '
  ↳ __init__', '__le__', '__lt__', '__module__', '__ne__
  ↳ ', '__new__', '__reduce__', '__reduce_ex__', '
  ↳ __repr__', '__setattr__', '__sizeof__', '__str__', '
  ↳ __subclasshook__', '__weakref__', '_get_angle', '
  ↳ _get_center', '_get_distance', 'axes', 'button', '
  ↳ copy', 'device', 'free', 'get', 'get_axis', '

```

```

↳ get_button', 'get_click_count', 'get_coords', '
↳ get_device', 'get_event_type', 'get_keycode', '
↳ get_keyval', 'get_root_coords', 'get_screen', '
↳ get_scroll_deltas', 'get_scroll_direction', '
↳ get_source_device', 'get_state', 'get_time', '
↳ get_window', 'handler_set', 'new', 'peek', 'put', '
↳ request_motions', 'send_event', 'set_device', '
↳ set_screen', 'set_source_device', 'state', 'time', '
↳ triggers_context_menu', 'type', 'window', 'x', '
↳ x_root', 'y', 'y_root']

```

W szczególności mamy dostęp do położenia kursora wewnątrz przycisku i na ekranie (`x`, `y`, `x_root`, `y_root`) oraz czasu wystąpienia zdarzenia (`time`).

Możemy każdemu elementowi okna przypisać odrębną funkcję obsługi zdarzenia, ale możemy też w jednej funkcji zróżnicować jej działanie w zależności od jej drugiego argumentu (jego wartością jest element, na którego obszarze nastąpiło zdarzenie).

Ćwiczenie 60. *Stwórz okno z dwoma przyciskami: „OK” oraz „Cancel”. Niech naciśnięcie przycisku spowoduje wypisanie jego nazwy w terminalu. Powinno to być zrealizowane za pomocą jednej funkcji obsługi zdarzenia dla obu przycisków.*

Ćwiczenie 61. *Dla najprostszego okna (bez obiektów składowych) dodaj następującą obsługę zdarzenia zamknięcia okna. Przy próbie zamknięcia okna powinno się pojawić okno dialogowe z etykietą „Czy na pewno chcesz zamknąć?” i dwoma przyciskami „OK” i „Cancel” poniżej. Naciśnięcie „OK” powinno kończyć działanie programu, a naciśnięcie „Cancel” tylko zamykać okno dialogowe.*

Zwolnienie przycisku myszy — sygnał `'button-release-event'` emitowany przez elementy okna.

Zmiana położenia lub rozmiaru okna — sygnał `'configure-event'` obiektu `Gtk.Window()`.

Sprawdźmy, jakie argumenty są przekazywane do funkcji:

```

(<__main__.Okno object at 0x7f57ca4368d0>, <Window object
↳ at 0x7f57cd263b40 (GtkWindow at 0x182e240)>, <void at
↳ 0x17f0db0>)

```

Pierwszym argumentem jest instancja, drugim obiekt, który wyemitował sygnał. Trzecim argumentem jest obiekt zdarzenia. Jeżeli wylistujemy go polece-

niem `dir`, zobaczymy atrybuty `x`, `y`, `width`, `height`, w których przechowywane są nowe położenie okna i jego nowy rozmiar.

Ćwiczenie 62. *Stwórz puste okno, które w tytule będzie pokazywać swoje aktualne położenie.*

Ćwiczenie 63. *Stwórz puste okno, które w tytule będzie pokazywać swój aktualny rozmiar.*

Naciśnięcie klawisza — sygnał `'key-press-event'` obiektu `Gtk.Window()`. Jest on emitowany przez okno w momencie, gdy zostaje naciśnięty klawisz i okno jest wtedy aktywne. Znów obiekt zdarzenia jest ostatnim argumentem przekazany do funkcji obsługi zdarzenia. Wylistowując go, znajdziemy atrybuty o nazwach `hardware_keycode` i `keyval`. Przyjrzyjmy się ich wartościom przy przyciskaniu różnych klawiszy. Wartości tych można używać, by różnicować działanie funkcji w reakcji na różne klawisze.

Ćwiczenie 64. *Stwórz puste okno, które reaguje na klawisze:*

- *strzałka* — przesuwa okno o 10 pikseli we wskazywanym kierunku
- *Esc* — kończy program

Zwolnienie klawisza — sygnał `'key-release-event'` obiektu `Gtk.Window()`.

Ćwiczenie 65. *Rozbuduj program z poprzedniego ćwiczenia, by kombinacja `Shift+strzałka` zmieniała o 10 pikseli rozmiar okna w wybranym kierunku.*

Wejście kursora w obszar okna — sygnał `'enter_notify_event'`.

Ćwiczenie 66. *Stwórz okno uciekające przed kursorem — w funkcji obsługi zdarzenia `'enter_notify_event'` wykryj, z którego boku kursor wszedł na obszar okna, i przesuń okno w przeciwnym kierunku.*

Wyjście kursora z obszaru okna — sygnał `'leave_notify_event'`.

Pełna lista sygnałów GTK jest dostępna pod adresem: <http://lazka.github.io/pgi-docs/Gtk-3.0/classes/Widget.html#signals>.

3.4 Metody elementów okna

W tym podrozdziale poznamy metody pozwalające pobierać tekst z obiektów klasy `Gtk.Entry` oraz zmieniać zawartość obiektu klasy `Gtk.TextView`.

Spśród metod klasy `Gtk.Entry` interesujące dla nas będą dwie:

- `get_text` — pobiera bieżącą zawartość pola.
- `set_text` — ustawia zawartość na podany łańcuch.

W obiekcie klasy `Gtk.TextView` za pomocą metody `set_buffer` ustawiamy bufor tekstowy — obiekt klasy `TextBuffer` — i na nim wykonujemy wszystkie operacje. Dla nas istotna będzie tylko jego metoda `set_text` klasy `Gtk.TextBuffer`, która ustawia zawartość bufora na podany łańcuch.

Ponieważ element `Gtk.TextView` będzie nam służył tylko do wyświetlania wyników, będziemy go ustawiać jako nieedytowalny przy użyciu metody `set_editable` — wywołana z argumentem `False` sprawia, że element jest nieedytowalny (służy tylko do wypisywania tekstu).

Dodatkowo każdy element okna ma metodę `grab_focus` (bez argumentów), która pozwala elementowi przechwycić klawiaturę. Od wywołania tej metody naciśnięcia klawiszy będą kierowane do elementu, który ją wywołał.

Listę wszystkich metod powyższych obiektów uzyskamy, wykonując polecenia: `dir(Gtk.Button)`, `dir(Gtk.TextView)`, `dir(Gtk.TextBuffer)`, `dir(Gtk.Entry)`, lub znajdziemy ją wraz z opisami na stronach:
<https://lazka.github.io/pgi-docs/Gtk-3.0/classes/Button.html>,
<https://lazka.github.io/pgi-docs/Gtk-3.0/classes/TextView.html>,
<https://lazka.github.io/pgi-docs/Gtk-3.0/classes/TextBuffer.html>,
<https://lazka.github.io/pgi-docs/Gtk-3.0/classes/Entry.html>.

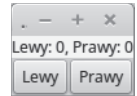
Ćwiczenie 67. *Dla rozwiązania ćwiczenia 58. dodaj obsługę zdarzenia naciśnięcia przycisku „OK”. Po naciśnięciu przycisku bieżąca zawartość pola `Gtk.Entry` powinna zostać dodana jako kolejna linia do tekstu w polu `Gtk.TextView`, a pole powinno zostać wyczyszczone. Pole `Gtk.TextView` powinno być nieedytowalne, a dostęp do klawiatury po uruchomieniu programu powinien należeć do pola `Gtk.Entry`.*

Ćwiczenie 68. *Rozbuduj rozwiązanie powyższego ćwiczenia o obsługę klawiszy. `Enter` powinien działać tak jak przycisk „OK”. `Esc` powinien kończyć program.*

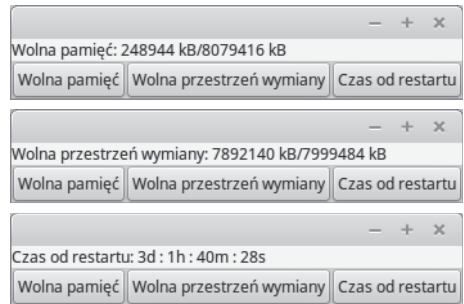
Ćwiczenie 69. *Spraw, by nie dało się przenieść kursora do nieedytowalnego pola `Gtk.TextView` oraz by po kliknięciu przycisku „OK” dostęp do klawiatury wracał do pola `Gtk.Entry`.*

Ćwiczenie 70. *Wykorzystując funkcję `stolica` z rozwiązania ćwiczenia 38., zmodyfikuj rozwiązanie poprzedniego ćwiczenia tak, by do tekstu w polu `Gtk.TextView` dodawana była wartość funkcji `stolica` dla argumentu pobranego z `Gtk.Entry`.*

Ćwiczenie 71. *Stwórz okno z nieedytowalnym polem tekstowym i przyciskami „Lewy”, „Prawy”. Okno powinno zawierać informację o tym, ile razy który przycisk został naciśnięty od początku działania programu, która powinna się uaktualniać przy każdym naciśnięciu przycisku — jak na rysunku obok. Spraw, by nie dało się w w oknie tekstowym umieścić kursora.*



Ćwiczenie 72. *Stwórz okno z nieedytowalnym polem tekstowym i przyciskami jak na rysunku obok. Spraw, by w oknie tekstowym nie dało się umieścić kursora. Okno powinno pokazywać, w zależności od naciśniętego przycisku: ilość wolnej pamięci na całkowitą ilość pamięci, ilość wolnej przestrzeni wymiany na całkowity rozmiar przestrzeni wymiany lub czas od restartu w formacie dni:godziny:minuty:sekundy. Dane dotyczące pamięci można znaleźć w pliku `/proc/meminfo`. Czas od restartu (w sekundach) to pierwsza liczba w pliku `/proc/uptime`.*



Dodawanie obrazów do okna*. Obiektem reprezentującym obraz jest `Gtk.Image`. Możemy go bezpośrednio dodać do obiektu okna. Inicjalizujemy go plikiem z obrazem za pomocą jego metody `set_from_file` lub za pomocą metody `set_from_pixbuf` obiektem `Pixbuf` z modułu `GdkPixbuf`, który pozwala na wykonywanie operacji na obrazie.

Obiekt `Pixbuf` tworzymy na podstawie obrazu z pliku za pomocą jego metody `new_from_file`. Możemy go skalować, korzystając z jego metody `scale_simple`, której argumentami są: nowa szerokość, nowa wysokość oraz metoda interpolacji (będziemy wybierać wolną, ale dającą najlepsze rezultaty `GdkPixbuf.InterpType.HYPER`). Z obiektu `Pixbuf` możemy też wyciąć fragment za pomocą jego metody `new_subpixbuf`, której argumentami są cztery

liczby całkowite: współrzędna x lewego górnego rogu wycinka, współrzędna y lewego górnego rogu wycinka, jego szerokość oraz wysokość.

Poniższy program stworzy okno z obrazem, który powstał na skutek przeskalowania obrazu z pliku `test.png` w katalogu roboczym do rozmiaru 500×500 , a następnie wycięcia z niego środkowego fragmentu o rozmiarach 300×300 :

```
1 from gi.repository import Gtk, GdkPixbuf
2
3
4 class Okno:
5
6     def __init__(self):
7         self.o=Gtk.Window()
8         pixbuf = GdkPixbuf.Pixbuf.new_from_file("./test.png")
9         pixbuf = pixbuf.scale_simple(500,500,GdkPixbuf.
10             ↪ InterpType.HYPER)
11         pixbuf = pixbuf.new_subpixbuf(100,100,300,300)
12         image = Gtk.Image()
13         image.set_from_pixbuf(pixbuf)
14         self.o.add(image)
15         self.o.show_all()
16
17 Okno()
18 Gtk.main()
```

Kod pod nazwą `gtk_image.py` w archiwum <https://ftp.helion.pl/przyklady/pytkn2.zip>.

Ćwiczenie 73. *Stwórz okno, w którym będzie wyświetlony fragment tapety pulpitu bezpośrednio pod nim. Zakładamy, że znamy zarówno ścieżkę do pliku tapety, jak i wymiary ekranu oraz że obraz z pliku jest skalowany do rozmiarów ekranu.*

Skorowidz

- #, 18
- #!, 46
- #%% (znak początku i końca komórki w Spyder), 13
- * (operator mnożenia), 15
- * (operator rozpakowania sekwencji), 32
- * (w definicji funkcji), 30
- ** (operator potęgowania), 15
- ** (operator rozpakowania słownika), 32
- **=, 15
- *=, 15
- +=, 15
- =, 15
- >, 33
- / (w definicji funkcji), 30
- //, 15
- //=, 15
- /=, 15
- :=, 27
- @, 166
- @classmethod, 231
- @property, 227
- @staticmethod, 231
- @vectorize (dekorator z pakietu `numpy`), 237
- % (formatowanie łańcuchów), 39
- % (reszta z dzielenia), 15
- %=, 15
- %d, 39
- %e, 39
- %f, 39
- %s, 39
- `%matplotlib` (dyrektywa IPython), 144, 148
- &, 262
- <<, 262
- [:], 16
- [], 16
- \n, 18
- \t, 18
- \v, 18
- ode (klasa z modułu `scipy.integrate`), 155
- |=, 28
- l, 28
- _, 11, 45
- `__add__`, 62
- `__aenter__`, 216
- `__aexit__`, 216
- `__aiter__`, 215
- `__anext__`, 215
- `__annotations__`, 34
- `__await__`, 205
- `__call__`, 64
- `__class__`, 177
- `__contains__`, 113
- `__delattr__`, 224
- `__delete__` (metoda deskryptora), 228
- `__dict__`, 176
- `__doc__`, 33, 59
- `__enter__` (metoda menedżera kontekstu), 179
- `__eq__`, 67
- `__exit__` (metoda menedżera kontekstu), 179

- __floordiv__, 62
- __ge__, 67
- __get__ (metoda deskryptora), 228
- __getattr__, 221
- __getattribute__, 222
- __getitem__, 65
- __gt__, 67
- __init__, 60
- __iter__, 182
- __le__, 67
- __lt__, 67
- '__main__', 42
- __mod__, 62
- __mul__, 62
- __name__, 42
- __name__ (atrybut funkcji), 166
- __ne__, 67
- __neg__, 68
- __next__, 182
- __pow__, 62
- __radd__, 63
- __rfloordiv__, 63
- __rmod__, 63
- __rmul__, 63
- __rpow__, 63
- __rsub__, 63
- __rtruediv__, 63
- __set__ (metoda deskryptora), 228
- __setattr__, 223
- __slots__, 233
- __str__, 61
- __sub__, 62
- __truediv__, 62
- _pickle (moduł), 52

- abs, 131
- abs (funkcja wbudowana), 16
- accept (metoda klasy socket), 92
- access (funkcja z modułu os), 53
- acos (funkcja z modułu math), 44
- add (metoda klasy Gtk.Window), 72
- AF_INET (stała z modułu socket), 91
- AF_INET6 (stała z modułu socket), 91
- AF_UNIX (stała z modułu socket), 91
- all, 23
- all_tasks (funkcja z modułu asyncio), 212
- and, 23

- animation (moduł z pakietu matplotlib), 147
- annotation (atrybut klasy Parameter), 170
- any, 23
- Apache (serwer HTTP), 110
- append (metoda typu list), 20
- arange (funkcja z pakietu numpy), 130
- args (atrybut wyjątku), 196
- argument nazwany (przekazany przez klucz), 30
- argument opcjonalny (o domyślnej wartości), 29
- argument pozycyjny, 30
- argv (zmienna z modułu sys), 45
- Array (klasa z modułu multiprocessing), 248
- as, 42
- asin (funkcja z modułu math), 44
- async def, 205
- async for, 215
- async with, 216
- asynchroniczny generator, 215
- asynchroniczny iterator, 215
- asynchroniczny menedżer kontekstu, 216
- asyncio (moduł), 205
- atan (funkcja z modułu math), 44
- atan2 (funkcja z modułu math), 224
- atomowość instrukcji, 85
- atrybut generowany dynamicznie, 220
- atrybuty prywatne, 224
- attr (moduł), 178
- attrib (funkcja z modułu attr), 179
- Attribute (klasa z modułu attr), 179
- attrs (dekorator z modułu attr), 178
- auto_scale_xyz (metoda typu Axes3D), 145
- await, 205
- Axes3D (typ z modułu mpl_toolkits.mplot3d), 142
- axis (funkcja z modułu pyplot), 134

- biblioteka standardowa, 43
- bind (metoda klasy socket), 92
- blok instrukcji, 22
- bpython, 12
- branie wycinków, 16

- break, 24
- Button (klasa z modułu Gtk), 72
- 'button-press-event' (sygnał emitowany przez elementy okna), 75
- 'button-release-event' (sygnał emitowany przez elementy okna), 77
- bytes (typ wbudowany), 16
- cancel (metoda klasy Task), 210
- CancelledError (wyjątek z modułu asyncio), 210
- cgi, 110
- cgi (moduł), 112
- chain (funkcja z modułu itertools), 189
- chdir (funkcja z modułu os), 53
- check_output (fun. z modułu subprocess), 54
- chmod (funkcja z modułu os), 53
- chown (funkcja z modułu os), 53
- chr (funkcja wbudowana), 19
- class, 58
- close (metoda generatora), 194
- close (metoda klasy Server), 218
- close (metoda klasy socket), 92
- close (metoda obiektu Gtk.Window), 75
- close (metoda pętli zdarzeń z modułu asyncio), 207
- close (metoda typu sqlite3.Connection), 103
- close (metoda typu file), 48
- colorbar (funkcja z modułu pyplot), 141
- combinations (funkcja z modułu itertools), 188
- combinations_with_replacement (funkcja z modułu itertools), 188
- commit (metoda typu sqlite3.Connection), 103
- complex (typ wbudowany), 15
- CONCAT (funkcja MySQL), 106
- concatenate (funkcja z modułu numpy), 239
- 'configure-event' (sygnał Gtk.Window), 77
- conjugate (metoda typu complex), 16
- connect (funkcja z modułu mysql.connector), 107
- connect (funkcja z modułu sqlite3), 102
- connect (metoda klas modułu Gtk), 75
- connect (metoda klasy socket), 99
- Connection (klasa z modułu multiprocessing), 247
- contextlib (moduł), 192
- contextmanager (dekorator z modułu contextlib), 192
- continue, 24
- contour (funkcja z modułu pyplot), 139
- contour (metoda typu Axes3D), 144
- contourf (funkcja z modułu pyplot), 140
- cos (funkcja z modułu math), 44
- count (funkcja z modułu itertools), 187
- count (metoda typu sekwencyjnego), 17
- CREATE DATABASE (polecenie MySQL), 106
- CREATE TABLE (polecenie MySQL), 106
- CREATE TABLE (polecenie SQLite), 102
- CREATE USER (polecenie MySQL), 106
- create_task (funkcja z modułu asyncio), 210
- create_task (metoda pętli zdarzeń z modułu asyncio), 207
- current_task (metoda klasy Task), 212
- cursor (metoda typu sqlite3.Connection), 103
- cycle (funkcja z modułu itertools), 187
- date (typ z modułu datetime), 51
- datetime (moduł), 51
- datetime (typ z modułu datetime), 51
- deadlock, 86

- decode (metoda typu str), 17
- def, 29
- default (atrybut klasy Parameter), 170
- dekorator funkcji, 165
- del, 28
- DELETE FROM (polecenie SQL), 102
- 'delete-event' (sygnał
Gtk.Window), 75
- deleter (atrybut deskryptora), 227
- deskryptor, 228
- dict, 27
- dir, 59
- divmod, 262
- dokumentacja funkcji, 33
- dokumentacja klasy, 59
- domknięcie funkcji, 165
- drain (metoda koprocedury klasy
StreamWriter), 217
- DROP TABLE (polecenie SQL), 102
- dropwhile (funkcja z modułu
itertools), 189
- dump (funkcja z modułu cPickle), 52
- dumps (funkcja z modułu cPickle), 52
- dziedziczenie, 61

- e (stała z modułu math), 44
- elif, 22
- else (dla instrukcji if), 22
- else (dla instrukcji try), 47
- else (dla pętli), 24
- empty (metoda klasy
multiprocessing.Queue),
244
- empty (metoda klasy queue.Queue),
89
- encode (metoda typu str), 17
- endswith (metoda typu str), 19
- 'enter_notify_event' (sygnał
Gtk.Window), 78
- Entry (klasa z modułu Gtk), 72
- environ (zmienna z modułu os), 53
- except, 46
- Exception (klasa nadrzędna
wyjątków), 196
- execute (metoda typu
sqlite3.Cursor), 103
- exp (funkcja z modułu math), 44

- fetchall (metoda typu
sqlite3.Cursor), 103
- fetchone (metoda typu
sqlite3.Cursor), 103
- FieldStorage (klasa z modułu cgi),
112
- figure (funkcja z modułu pyplot),
142
- file (typ wbudowany), 48
- filter, 36
- filterfalse (funkcja z modułu
itertools), 188
- finally, 47
- float (typ wbudowany), 15
- float128 (typ z pakietu numpy), 130
- float64 (typ z pakietu numpy), 130
- for, 23
- format, 40
- from, 42
- FuncAnimation (klasa z modułu
animation), 147
- funkcja generatora, 185
- functools, 37, 166
- funkcja, 29
- funkcja fabryki, 165
- funkcja generatora asynchronicznego,
215
- funkcja koprocedury, 205

- gather (funkcja z modułu asyncio),
208
- GdkPixbuf (moduł z pakietu gi), 80
- geany, 12
- generator, 185
- generator asynchroniczny, 215
- GeneratorExit, 194
- get (funkcja z modułu requests), 55
- get (metoda klasy
multiprocessing.Queue),
244
- get (metoda klasy queue.Queue), 89
- get (metoda typu dict), 27
- get_event_loop (funkcja z modułu
asyncio), 207
- get_position (metoda klasy
Gtk.Window), 71
- get_size (metoda klasy Gtk.Window),
71

- get_text (metoda klasy `Gtk.Entry`), 79
 get_title (metoda klasy `Gtk.Window()`), 71
 get_traced_memory (funkcja z modułu `tracemalloc`), 233
 getcwd (funkcja z modułu `os`), 53
 getenv (funkcja z modułu `os`), 53
 getfirst (metoda klasy `cgi.FieldStorage`), 112
 getlist (metoda klasy `cgi.FieldStorage`), 112
 gi (pakiet modułów), 70
 global, 35
 gniazdo sieciowe, 91
 grab_focus (metoda elementów okna), 79
 GRANT ALL ON ... TO (polecenie MySQL), 106
 Gtk (moduł z pakietu `gi`), 70

 hasattr, 177
 HBox (klasa z modułu `Gtk`), 73
 hstack (funkcja z modułu `numpy`), 239
 HTML (klasa z modułu `IPython.display`), 148

 if, 22
 imag (atrybut typu `complex`), 16
 Image (klasa z modułu `Gtk`), 80
 import, 42
 imshow (funkcja z modułu `matplotlib`), 141
 in (członkostwo w typie sekwencyjnym), 17
 in (członkostwo w słowniku), 29
 indeksowanie, 16
 index (metoda typu sekwencyjnego), 17
 input, 45
 insert (metoda typu `list`), 20
 INSERT INTO (polecenie SQL), 102
 inspect (moduł), 170, 205
 instancja, 58
 int (typ wbudowany), 15
 integrate (metoda klasy `ode`), 156
 integrate (moduł z pakietu `scipy`), 150

 interaktywna powłoka, 11
 ipython, 12
 IPython.display (moduł), 148
 isalnum (metoda typu `str`), 19
 isalpha (metoda typu `str`), 19
 iscoroutine (funkcja z modułu `inspect`), 205
 isdigit (metoda typu `str`), 19
 isinstance, 61
 islice (funkcja z modułu `itertools`), 188
 isnan (funkcja z pakietu `numpy`), 162
 iter, 184
 iter_lines (metoda obiektu `Response`), 55
 iterator, 182
 iterator asynchroniczny, 215
 itertools (moduł), 187

 join (metoda klasy `Process`), 243
 join (metoda klasy `Thread`), 84
 join (metoda typu `str`), 19
 jupyter notebook, 12

 keep_blank_values (argument konstruktora obiektu `FieldStorage`), 113
 'key-press-event' (sygnał `Gtk.Window`), 78
 'key-release-event' (sygnał `Gtk.Window`), 78
 keys (metoda typu `dict`), 28
 kind (atrybut klasy `Parameter`), 170
 klasa, 58
 kodowanie znaków, 18
 komentarze, 18
 konkatencja sekwencji, 17
 konstruktor, 60
 koprocedura, 205
 krotka, 16
 kwantyfikatory, 23

 Label (klasa z modułu `Gtk`), 72
 lambda, 35
 laplasjan, 157
 'leave_notify_event' (sygnał `Gtk.Window`), 78

- legend (funkcja z modułu pyplot), 133
- len(s), 17
- liczby zespolone, 15
- LifoQueue (klasa z modułu Queue), 90
- LIKE (operator w SQL), 102
- linspace (funkcja z pakietu numpy), 130
- list (typ wbudowany), 16
- lista, 16
- lista składana, 25
- listdir (funkcja z modułu os), 53
- listen (metoda klasy socket), 92
- load (funkcja z modułu cPickle), 52
- loads (funkcja z modułu cPickle), 52
- Lock (funkcja z modułu threading), 85
- log (funkcja z modułu math), 44
- log10 (funkcja z modułu math), 44
- logical_and (funkcja pakietu numpy), 131
- logical_not (funkcja pakietu numpy), 131
- logical_or (funkcja pakietu numpy), 131
- lstrip (metoda typu str), 19
- łańcuch znaków, 16
- łańcuch bajtów, 16
- main (funkcja z modułu Gtk), 70
- main_quit(funkcja z modułu Gtk), 75
- map, 35
- map (metoda obiektu Pool), 242
- mappingproxy, 176
- math (moduł), 44
- matplotlib (pakiet modułów), 132
- max, 17
- max (funkcja z modułu numpy), 160
- menedżer kontekstu, 179
- menedżer kontekstu asynchroniczny, 216
- meshgrid (funkcja z pakietu numpy), 138
- metaklasa, 175
- metoda klasy, 231
- metoda statyczna, 231
- mikrowątki, 201
- min, 17
- min (funkcja z modułu numpy), 160
- mkdir (funkcja z modułu os), 53
- mode (atrybut zmiennej plikowej), 48
- moduł, 42
- modyfikowalny w miejscu, 21
- move (metoda klasy Gtk.Window), 71
- multiprocessing (moduł), 242
- mysql.connector (moduł), 107
- mysqlclient (moduł), 107
- MySQLdb (moduł), 107
- name (atrybut klasy Parameter), 170
- name (atrybut zmiennej plikowej), 48
- name (funkcja z modułu os), 53
- NaN (stała w pakiecie numpy), 134
- nanmax (funkcja z pakietu numpy), 142
- nanmin (funkcja z pakietu numpy), 142
- ndarray (typ z pakietu numpy), 129
- new_from_file (metoda obiektu Pixbuf), 80
- new_subpixbuf (metoda obiektu Pixbuf), 80
- next (funkcja wbudowana), 184
- np (skrót od numpy), 129
- numpy (pakiet modułów), 129
- numpy.random (moduł), 199
- odeint (funkcja z modułu scipy.integrate), 150
- ones (funkcja z pakietu numpy), 130, 138
- open, 47
- operator członkostwa, 17, 29
- operatory logiczne, 23
- operatory porównania, 23
- or, 23
- ord (funkcja wbudowana), 19
- ORDER BY ... (polecenie SQL), 102
- os (moduł), 53
- pętla zdarzeń, 206, 207
- pack_end (metoda klas Gtk.VBox i Gtk.HBox), 73
- pack_start (metoda klas Gtk.VBox i Gtk.HBox), 73
- pakiety modułów, 43
- Parameter (klasa z modułu inspect), 170

- parameters (atrybut obiektu `Signature`), 170
- pass, 24
- permutations (funkcja z modułu `itertools`), 187
- pi (stała z modułu `math`), 44
- Pipe (funkcja z modułu `multiprocessing`), 246
- Pixbuf (obiekt z modułu `GdkPixbuf`), 80
- plot (funkcja z modułu `pyplot`), 132
- plot_surface (metoda typu `Axes3D`), 142
- plt (skrót od `matplotlib.pyplot`), 132
- pochodna numeryczna druga, 157
- pochodna numeryczna pierwsza, 157
- polar (funkcja z modułu `pyplot`), 134
- Pool (funkcja z modułu `multiprocessing`), 242
- pop (metoda typu `list`), 20
- pop (metoda typu `dict`), 28
- Popen, 54
- powłoka, 11
- powtarzanie sekwencji, 16
- print, 12
- PriorityQueue (klasa z modułu `Queue`), 90
- problem płytkiej kopii, 21
- Process (klasa z modułu `multiprocessing`), 243
- product (funkcja z modułu `itertools`), 187
- property, 225
- przeciążanie, 62
- przysyłanie, 61
- put (metoda klasy `multiprocessing.Queue`), 244
- put (metoda klasy `queue.Queue`), 89
- pycharm, 12
- PyMySQL (moduł), 107
- pyplot (moduł z pakietu `matplotlib`), 132
- pyswarm (moduł), 201
- query string, 112
- Queue (klasa z modułu `multiprocessing`), 244
- Queue (klasa z modułu `queue`), 89
- queue (moduł), 89
- quiver (funkcja z modułu `pyplot`), 146
- raise, 67
- rand (funkcja z modułu `numpy.random`), 199
- randint (funkcja z modułu `random`), 44, 108
- randn (funkcja z modułu `numpy.random`), 199
- random (funkcja z modułu `random`), 44
- random (moduł), 44, 108
- range, 183
- range (typ wbudowany), 23
- read (metoda klasy `StreamReader`), 217
- read (metoda typu `file`), 48
- readline (metoda typu `file`), 48
- readlines (metoda typu `file`), 48
- real (atrybut typu `complex`), 16
- recv (metoda klasy `socket`), 92
- recv (metoda obiektu `multiprocessing.Connection`), 247
- recvfrom (metoda klasy `socket`), 92
- reduce (funkcja z modułu `functools`), 37
- remove (funkcja z modułu `os`), 53
- remove (metoda typu `list`), 20
- rename (funkcja z modułu `os`), 53
- repeat (funkcja z modułu `itertools`), 187
- replace (metoda typu `str`), 19
- requests (moduł), 55
- resize (metoda klasy `Gtk.Window`), 71
- Response (obiekt z modułu `requests`), 55
- reszta z dzielenia, 15
- return, 29
- rmdir (funkcja z modułu `os`), 53
- rollback (metoda typu `sqlite3.Connection`), 103
- rsplit (metoda typu `str`), 19
- rstrip (metoda typu `str`), 19

- run (funkcja z modułu `asyncio`), 206
- run_in_executor (metoda pętli zdarzeń), 213
- słownik statyczny, 176
- słownik uporządkowany, 170
- scale_simple (metoda obiektu `Pixbuf`), 80
- scipy (pakiet modułów), 150
- select (funkcja z modułu `select`), 201
- select (moduł), 201
- SELECT ... FROM (polecenie SQL), 102
- self, 59
- Semaphore (funkcja z modułu `threading`), 88
- send (metoda generatora), 193
- send (metoda obiektu `multiprocessing.Connection`), 247
- sendall (metoda klasy `socket`), 92
- sendto (metoda klasy `socket`), 92
- serve_forever (metoda koprocedury klasy `Server`), 218
- Server (klasa z modułu `asyncio`), 217
- set_buffer (metoda obiektu `Gtk.TextView`), 79
- set_editable (metoda klasy `Gtk.TextBuffer`), 79
- set_from_file (metoda klasy `Gtk.Image`), 80
- set_from_pixbuf (metoda klasy `Gtk.Image`), 80
- set_initial_value (metoda klasy `ode`), 155
- set_text (metoda klasy `Gtk.Entry`), 79
- set_text (metoda klasy `Gtk.TextBuffer`), 79
- set_title (metoda klasy `Gtk.Window`), 71
- set_xdata (funkcja uaktualniająca dane na wykresie), 148
- set_xlim (funkcja ustalająca zakres osi *x* wykresu), 153
- set_ydata (funkcja uaktualniająca dane na wykresie), 148
- set_ylim (funkcja ustalająca zakres osi *y* wykresu), 153
- setenv (funkcja z modułu `os`), 53
- setsockopt (metoda klasy `socket`), 92
- setter (atrybut deskryptora), 227
- shape (atrybut typu `numpy.array`), 236
- shebang, 46
- show (funkcja z modułu `pyplot`), 133
- show (metoda klas z modułu `Gtk`), 70
- show_all (metoda klasy `Gtk.Window`), 74
- signature (funkcja z modułu `inspect`), 170
- Signature (klasa z modułu `inspect`), 170
- sin (funkcja z modułu `math`), 44
- skrypt, 12
- skrypt cgi, 110
- sleep (funkcja z modułu `asyncio`), 205
- sleep (funkcja z modułu `time`), 82
- słownik, 27
- słabe typowanie, 15
- SOCK_DGRAM (stała z modułu `socket`), 91
- SOCK_STREAM (stała z modułu `socket`), 91
- socket (klasa z modułu `socket`), 91
- socket (moduł), 91
- sort (metoda typu `list`), 37
- sorted, 36
- split (metoda typu `str`), 19
- Spyder, 13
- SQL injection, 105
- sqlite3 (moduł), 102
- starmap (funkcja z modułu `itertools`), 189
- start (funkcja z modułu `tracemalloc`), 233
- start (metoda klasy `Process`), 243
- start (metoda klasy `Thread`), 83
- start_server (funkcja z modułu `asyncio`), 217
- startswith (metoda typu `str`), 19
- stop (funkcja z modułu

- tracemalloc), 233
- StopAsyncIteration, 215
- StopIteration, 182
- str (typ wbudowany), 16
- StreamReader (klasa z modułu asyncio), 217
- StreamWriter (klasa z modułu asyncio), 217
- strip (metoda typu str), 19
- subplots (funkcja z modułu pyplot), 148
- subprocess (moduł), 54
- sum, 20
- super, 66
- sys (moduł), 45

- tablica asocjacyjna, 27
- takewhile (funkcja z modułu itertools), 189
- tan (funkcja z modułu math), 44
- Task (klasa z modułu asyncio), 207
- tee (funkcja z modułu itertools), 189
- text (atrybut obiektu Response), 55
- TextBuffer (klasa z modułu Gtk), 79
- TextView (klasa z modułu Gtk), 72
- Thread (moduł), 83
- thread-safe, 84
- threading (klasa z modułu threading), 83
- throw (metoda generatora), 197
- time (funkcja z modułu time), 50
- time (moduł), 50
- time (typ z modułu datetime), 51
- to_html5_video (metoda klasy FuncAnimation), 148
- trójargumentowy operator logiczny, 23
- tracemalloc (moduł), 233
- try, 46
- tryb binarny, 48
- tryb tekstowy, 48
- tuple (typ wbudowany), 16
- typ mapujący, 27
- type, 14, 175

- uname (funkcja z modułu os), 53
- update (metoda typu dict), 28
- UPDATE ... SET (polecenie SQL), 102
- USE (polecenie MySQL), 106

- Value (klasa z modułu multiprocessing), 248
- values (metoda typu dict), 29
- VBox (klasa z modułu Gtk), 73
- vstack (funkcja z modułu numpy), 239

- wątek demona, 84
- wątek roboczy, 89
- WHERE (słowo kluczowe SQL), 102
- while, 24
- widok, 28
- Window (klasa z modułu Gtk), 70
- with, 49
- wraps (dekorator z modułu functools), 166
- write (metoda klasy StreamWriter), 217
- write (metoda typu file), 48
- writelines (metoda typu file), 48
- wycinki tablic, 139
- wyrażenie generatora, 186
- wyrażenie przypisania, 26

- yield, 185
- yield from, 186

- zakleszczenie, 86
- zeros (funkcja z pakietu numpy), 130, 138
- zip, 20
- znak ucieczki, 18

PROGRAM PARTNERSKI

— GRUPY HELION —

- 
1. ZAREJESTRUJ SIĘ
 2. PREZENTUJ KSIĄŻKI
 3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion

Python – prosto, szybko, skutecznie

- Konstrukcje języka Python
- Sposoby ich użycia
- Zadania z rozwiązaniami

Python to nowoczesny, potężny i uniwersalny język programowania, który zdobył dużą popularność zarówno wśród zawodowców z branży IT, jak i w szkołach i na uczelniach – jako doskonałe narzędzie do nauki programowania na różnych poziomach. Duże możliwości, prostota i zwartość składni, czytelność kodu, wszechstronność zastosowań i wsparcie wielu paradygmatów sprawiają, że co najmniej podstawowa znajomość Pythona stanowi jedno z najczęstszych wymagań wobec poszukujących pracy programistów i inżynierów, nawet jeśli na co dzień mają oni używać zupełnie innych technologii.

Ta książka pomoże nauczycielom, uczniom, studentom i wszystkim zainteresowanym poznanie Pythona opanować podstawy tego języka i rozpocząć stosowanie go w praktyce. Krok po kroku uczy czytać i pisać kod, przedstawiając zarówno konstrukcje i funkcje języka, jak i możliwości ich praktycznego użycia do rozwiązywania typowych problemów programistycznych. Programowanie interfejsów graficznych, programowanie wielowątkowe, programowanie sieciowe, tworzenie stron WWW, obliczenia numeryczne – w tym wszystkim świetnie sprawdzi się Python. I wszystko to można znaleźć w tej książce!

- Pisanie i uruchamianie programów w Pythonie
- Podstawowe typy danych i ich zastosowanie
- Instrukcje warunkowe, pętle i funkcje
- Obsługa wyjątków i używanie plików
- Programowanie obiektowe w Pythonie
- Graficzny interfejs użytkownika
- Wielowątkowość, sieci, bazy danych i strony WWW
- Funkcje wyższych rzędów
- Iteratory, generatory, koprocedury
- Programowanie asynchroniczne i współbieżne

Programowanie w Pythonie w praktyce

Dr hab. inż. Gniewomir Sarbicki specjalizuje się w informatyce kwantowej. Zawodowo używa Pythona w obliczeniach numerycznych dotyczących splątania kwantowego i jest opiekunem prac inżynierskich powstających z wykorzystaniem tego języka. Od 2012 roku prowadzi kursy w zakresie Pythona na Wydziale Fizyki, Astronomii i Informatyki Stosowanej Uniwersytetu Mikołaja Kopernika w Toruniu.

Helion 

 helion.pl

 **HELION SA**
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

Sprawdź nasze szkolenia!



HELIONSZKOLENIA.PL

KOD KORZYŚCI
Sięgnij po więcej! ►



ISBN 978-83-283-8239-8



INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 79,00 zł