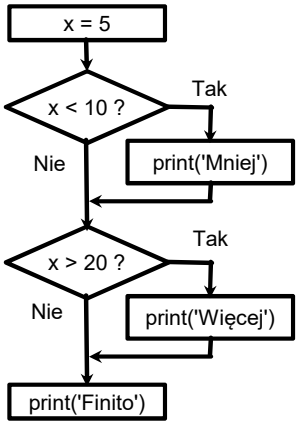
Wykonanie warunkowe

Rozdział 3

Python dla wszystkich
www.py4e.pl



Kroki warunkowe

Program:

```
x = 5
if x < 10:
    print('Mniej')
if x > 20:
    print('Więcej')

print('Finito')
```

Wyjście:

```
Mniej
Finito
```

Operatory porównania

- Wyrażenia logiczne zadają pytania i dają odpowiedzi Tak lub Nie, których używamy do kontroli przepływu programu
- Wyrażenia logiczne używając operatorów porównania, zwracają wartości True/ False, czyli Prawda/ Fałsz albo Tak/ Nie
- Operatory porównania przyglądają się zmiennym, ale ich nie modyfikują

Python	Znaczenie
<	Mniejsze niż
<=	Mniejsze lub równe
==	Równe
>=	Większe lub równe
>	Większe niż
!=	Różne od

Pamiętaj: "=" służy do przypisania.

https://pl.wikipedia.org/wiki/George_Boole

Operatory porównania

```
x = 5
if x == 5 :
    print('Równe 5')
if x > 4 :
   print('Większe niż 4')
if  x >= 5 :
    print('Większe lub równe 5')
if x < 6 : print('Mniejsze niż 6')
if x <= 5 :
    print('Mniejsze lub równe 5')
if x != 6 :
    print('Różne od 6')
```

Równe 5

Większe niż 4

Większe lub równe 5

Mniejsze niż 6

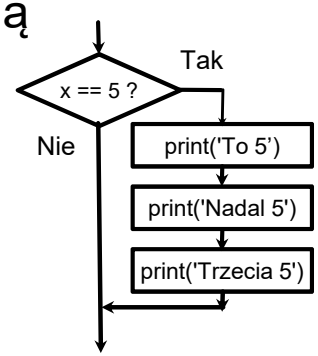
Mniejsze lub równe 5

Różne od 6

Decyzje z jedną opcją

```
x = 5
print('Przed 5')
if  x == 5 :
    print('To 5')
    print('Nadal 5')
    print('Trzecia 5')
print('Po 5')
print(':Przed 6')
if x == 6 :
    print('To 6')
    print('Nadal 6')
    print('Trzecia 6')
print('Po 6')
```

```
Przed 5

To 5
Nadal 5
Trzecia 5
Po 5
Przed 6

Po 6
```

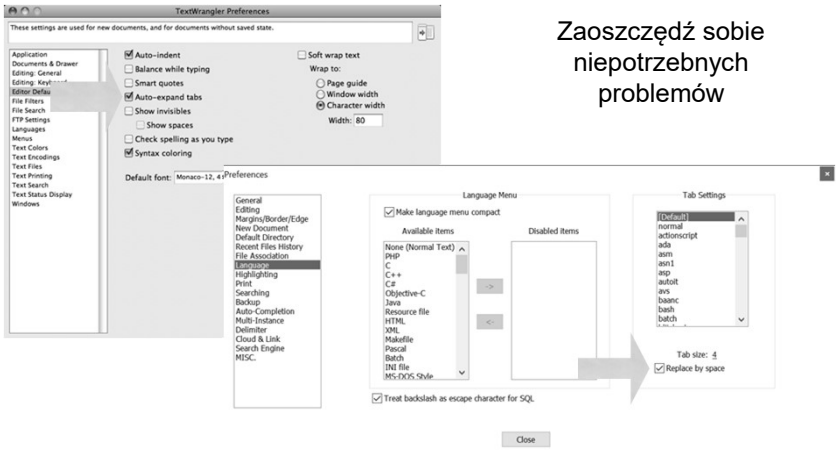


Wcięcia

- Zwiększ wcięcie po instrukcji if lub for (po :)
- Utrzymuj wcięcie, aby oznaczyć zakres bloku (którego linii dotyczą if/for)
- Zmniejsz wcięcie do poziomu instrukcji if lub for, aby oznaczyć koniec bloku
- Puste linie są ignorowane – nie mają wpływu na wcięcia
- Komentarze zajmujące całą linię nie mają wpływu na wcięcia

Uwaga: Wyłącz tabulator!!!

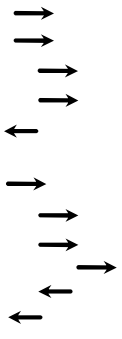
- Atom automatycznie wstawia spacje w plikach z rozszerzeniem ".py" (fajnie!)
- Większość edytorów tekstu umie zamieniać tabulatory na spacje – upewnij się, że ta funkcja jest włączona
  - Notepad++: Ustawienia -> Ustawienia -> Języki/ Rozmiar tabulacji [x] Zastąp spacją
  - TextWrangler: TextWrangler -> Preferences -> Editor Defaults
- Python *bardzo* zwraca uwagę na głębokość wcięcia linii. Jeśli pomieszasz tabulatory i spacje, możesz zobaczyć “błędy wcięć”, nawet jeśli wszystko wygląda ok.



Zaoszczędź sobie niepotrzebnych problemów

zwiększ / utrzymaj po if lub for

zmniejsz, aby oznaczyć koniec bloku



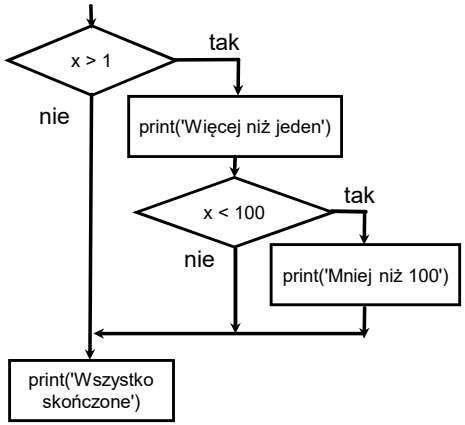
```
x = 5
if x > 2 :
    print('Więcej niż2')
    print('Nadal więcej')
print('Koniec z 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Więcej niż 2')
    print('Koniec z i', i)
print('Wszystko skończone')
```

Pomyśl o początku/ końcu bloków

```
x = 5
if x > 2 :
    print('Więcej niż 2')
    print('Nadal więcej')
print('Koniec z 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Więcej niż 2')
    print('Koniec z i', i)
print('Wszystko skończone')
```

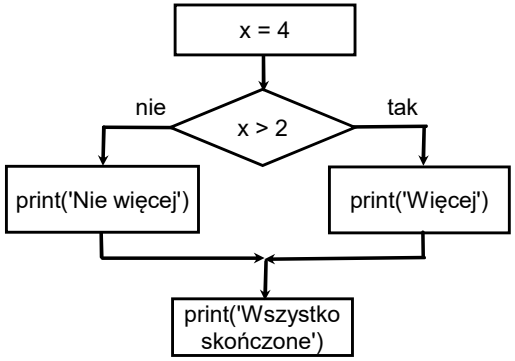
Zagnieżdżone decyzje

```
x = 42
if x > 1 :
    print('Więcej niż jeden')
    if x < 100 :
        print('Mniej niż 100')
print('Wszystko skończone')
```



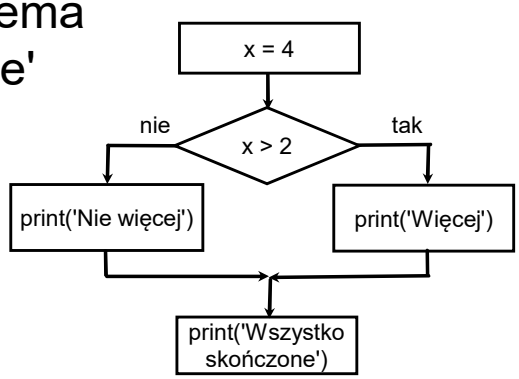
Decyzje z dwiema opcjami

- Czasami chcemy zrobić jedną rzecz, jeśli wyrażenie logiczne jest prawdziwe, a coś innego, kiedy jest fałszywe
- To jak rozstaje dróg: można pojechać jedną albo drugą, ale nie obiema



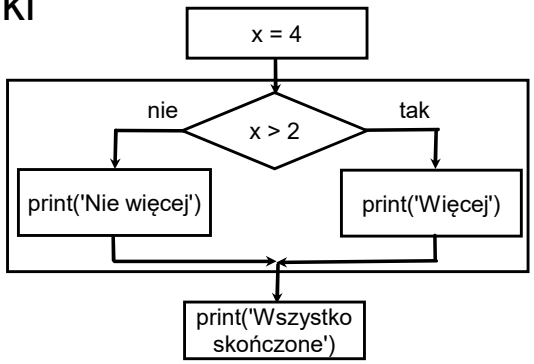
Decyzje z dwiema opcjami 'else'

```
x = 4

if x > 2 :
    print('Więcej')
else :
    print('Mniej')

print('Wszystko
skończone')
```



Zobacz bloki

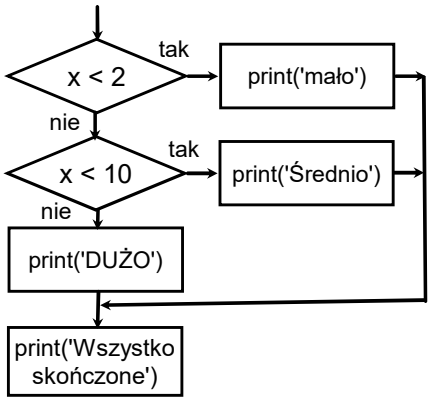
```
x = 4

if x > 2 :
    print('Więcej')
else :
    print('Mniej')

print('Wszystko
skończone')
```



Więcej instrukcji warunkowych

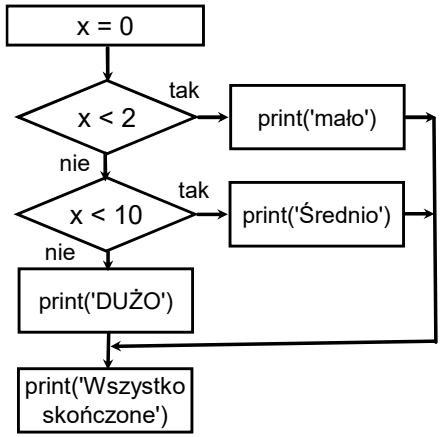
Wiele opcji

```
if x < 2 :
    print('mało')
elif x < 10 :
    print('Średnio')
else :
    print('DUŻO')
print('Wszystko
skończone')
```



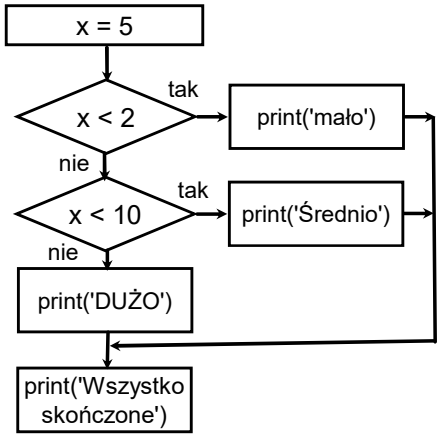
Wiele opcji

```
x = 0
if x < 2 :
    print('mało')
elif x < 10 :
    print('Średnio')
else :
    print('DUŻO')
print('Wszystko
skończone')
```



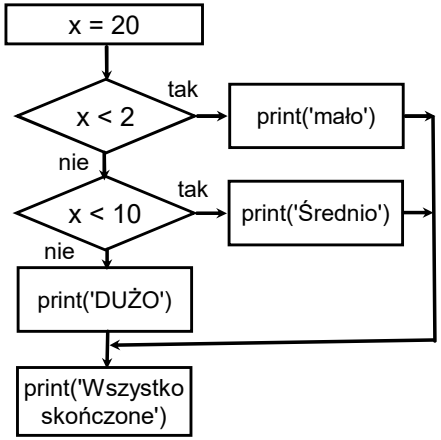
Wiele opcji

```
x = 5
if x < 2 :
    print('mało')
elif x < 10 :
    print('Średnio')
else :
    print('DUŻO')
print('Wszystko
skończone')
```



Wiele opcji

```
x = 20
if x < 2 :
    print('mało')
elif x < 10 :
    print('Średnio')
else :
    print('DUŻO')
print('Wszystko
skończone')
```



Wiele opcji

```
# Brak 'else'
x = 5
if x < 2 :
    print('Mało')
elif x < 10 :
    print('Średnio')

print('Wszystko
skończone')
```

```
if x < 2 :
    print('Mało')
elif x < 10 :
    print('Średnio')
elif x < 20 :
    print('Dużo')
elif x < 40 :
    print('DUŻO')
elif x < 100:
    print('Ogromnie dużo')
else :
    print('Niesamowicie
ogromnie dużo')
```

Zagadki z wieloma opcjami

Która instrukcja print nie zostanie wykonana bez względu na wartość x?

```
if x < 2 :
    print('Poniżej 2')
elif x >= 2 :
    print('Dwa lub więcej')
else :
    print('Coś innego')
```

```
if x < 2 :
    print('Mniej niż 2')
elif x < 20 :
    print('Mniej niż 20')
elif x < 10 :
    print('Mniej niż 10')
else :
    print('Coś innego')
```

Struktura try / except

- Niebezpieczne fragenty kodu obuduj strukturą try / except
- Jeśli kod w try (spróbuj) zadziała, to except (wyjątek) jest pomijany
- Jeśli kod w try zawiedzie, to program przeskoczy do sekcji except

```
$ cat notry.py
astr = 'Hello Bob'
istr = int(astr)
print('First', istr)
astr = '123'
istr = int(astr)
print('Second', istr)
```

```
$ python3 notry.py
Traceback (most recent call last):
  File "notry.py", line 2, in <module>
    istr = int(astr)
ValueError: invalid literal for int() with
base 10: 'Hello Bob'
```

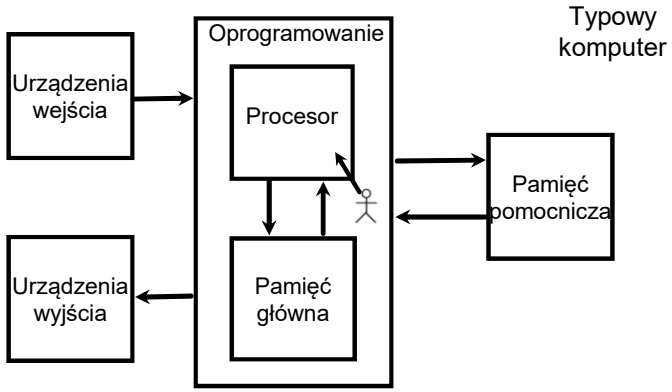
Wszystko skończone

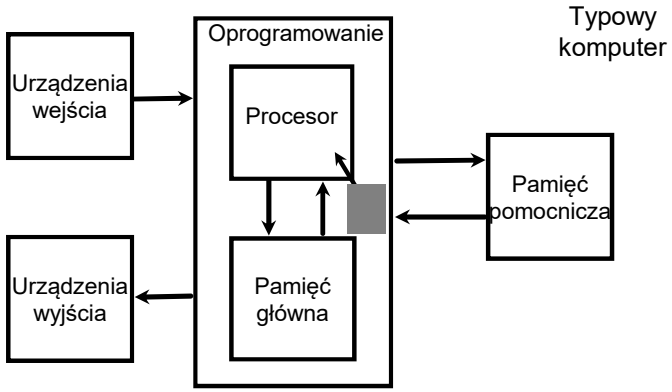
Tu program się zatrzyma →

```
$ cat notry.py
astr = 'Hello Bob'
istr = int(astr)
```


```
$ python3 notry.py
Traceback (most recent call last):
  File "notry.py", line 2, in <module>
    istr = int(astr)
ValueError: invalid literal for int() with
base 10: 'Hello Bob'
```

Wszystko skończone






```
astr = 'Hello Bob'
try:
    istr = int(astr)
except:
    istr = -1

print('First', istr)

astr = '123'
try:
    istr = int(astr)
except:
    istr = -1

print('Second', istr)
```

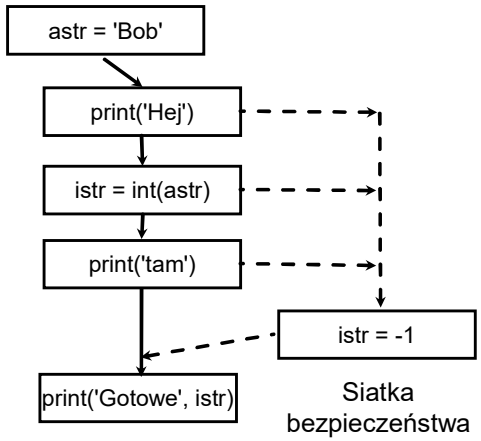
Kiedy pierwsza konwersja nie zadziała, program przechodzi do klauzuli except i działa dalej

```
$ python3 tryexcept.py
First -1
Second 123
```

Kiedy druga konwersja udaje się, program przeskakuje klauzulę except i działa dalej

try / except

```
astr = 'Bob'
try:
    print('Hej')
    istr = int(astr)
    print('tam')
except:
    istr = -1

print('Gotowe', istr)
```



Siatka bezpieczeństwa

Przykład try / except

```
rawstr = input('Podaj liczbę: ')
try:
    ival = int(rawstr)
except:
    ival = -1

if ival > 0 :
    print('Niezła robota')
else:
    print('To nie jest liczba')
```

```
$ python3 trynum.py
Podaj liczbę: 42
Niezła robota
$ python3 trynum.py
Podaj liczbę: forty-two
To nie jest liczba
$
```

Podsumowanie

- Operatory porównania
  == <= >= > < !=
- Wcięcia
- Decyzje z dwiema opcjami
- Decyzje z dwiema opcjami: if: i else:
- Zagnieżdżone decyzje
- Decyzje z wieloma opcjami z elif
- try / except do obsługi błędów

Ćwiczenie

Przepisz ponownie swój program obliczający wynagrodzenie, tak aby dać pracownikowi 1,5 raza większą stawkę godzinową za czas przepracowany powyżej 40 godzin.

```
Podaj liczbę godzin: 45
Podaj stawkę godzinową: 10

Wynagrodzenie: 475.0
```

475 = 40 * 10 + 5 * 15

Ćwiczenie

Przepisz ponownie swój program płacowy, używając try i except, tak aby elegancko obsługiwał nienumeryczne dane wejściowe

```
Podaj liczbę godzin: 20
Podaj stawkę: dziewięćęć
Błąd, podaj wartość numeryczną

Podaj liczbę godzin: czterdzieścieści
Błąd, podaj wartość numeryczną
```

UMSI

Podziękowania dla współpracowników

Autorstwo pierwszej wersji: Charles Severance, University of Michigan School of Information

Polska wersja powstała z inicjatywy Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu

Tłumaczenie: Agata i Krzysztof Wierzbiccy, EnglishT.eu

... wstaw tu nowych współpracowników i tłumaczy

...