

// T-OLIMPIANS – ARKUSZ REFERENCYJNY

# Python Turtle — ściaga

Wszystkie polecenia grafiki żółwia wyjaśnione krok po kroku: ruch, pióro, kolory, okno, zdarzenia, wzorce i najczęstsze błędy. Pod konkurs informatyczny LOGIA i lekcje informatyki.

PYTHON 3 • MODUŁ TURTLE • LOGIA • KLASY IV-VIII

- |                              |                          |
|------------------------------|--------------------------|
| 01 Jak zacząć                | 10 Okno i ekran          |
| 02 Układ współrzędnych       | 11 Tekst i pytania       |
| 03 Ruch i obrót              | 12 Zdarzenia             |
| 04 Skok w miejsce            | 13 Wiele żółwi           |
| 05 Stan żółwia               | 14 Wzorce na LOGIE       |
| 06 Pióro                     | 15 Rekurencja i fraktale |
| 07 Kolory i wypełnianie      | 16 Najczęstsze błędy     |
| 08 Kropki, stemple, szybkość | 17 Skróty                |
| 09 Wygląd żółwia             |                          |

## 01 Jak zacząć — szablon programu

Moduł **turtle** to „żółw” z pisakiem, który chodzi po ekranie i rysuje. Każdy program zaczyna się od importu, potem wydajesz żółwiowi polecenia jedno po drugim.

SZABLON — TAK JAK ZALECA LOGIA

```
from turtle import *      # 1. importujemy wszystkie polecenia żółwia

def kwadrat(bok):         # 2. własne funkcje rysujące
    for i in range(4):
        fd(bok)
        lt(90)

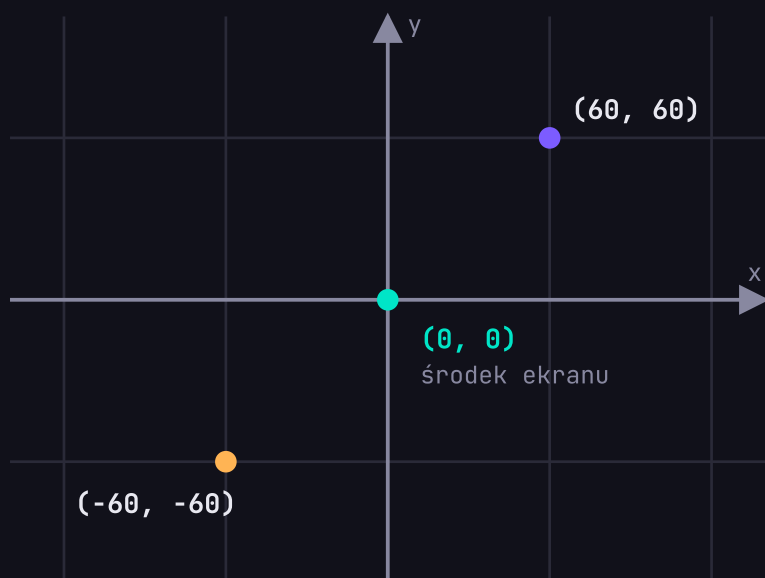
# tracer(0)               # 3. (opcjonalnie) wyłącz animację — rysuje od razu
kwadrat(100)              # 4. wywołanie funkcji = rysunek
# update()                #   po tracer(0) trzeba odświeżyć ekran
done()                   # 5. okno zostaje otwarte po narysowaniu
```

**LOGIA:** organizatorzy polecają `from turtle import *` zamiast `import turtle`. Wtedy piszesz `fd(100)`, a nie `turtle.fd(100)`. Na konkursie oddajesz zwykle **definicję funkcji**, więc testuj ją, wywołując na dole pliku.

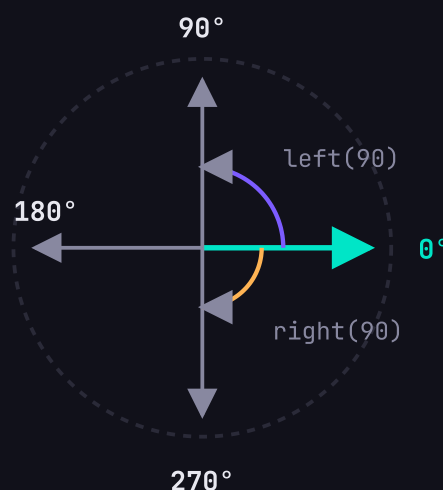
**Uwaga:** nie nazywaj swojego pliku `turtle.py` — Python zaimportuje wtedy Twój plik zamiast modułu i nic nie zadziała.

## 02 Układ współrzędnych i kierunki

Żółw startuje w punkcie **(0, 0)** na środku ekranu i patrzy **w prawo** (kierunek  $0^\circ$ ). Oś x rośnie w prawo, oś y w górę — jak na matematyce. Kąty liczymy przeciwnie do ruchu wskazówek zegara.



1 kratka = 60 kroków żółwia

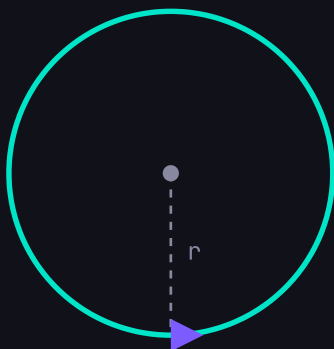


**left(kąt)** obraca żółwia przeciwnie do wskazówek zegara (w lewo), **right(kąt)** — zgodnie z nimi (w prawo). Obrót nie przesuwa żółwia, zmienia tylko kierunek.

### 03 Ruch i obrót — podstawa wszystkiego

| Polecenie                                                        | Co robi                                                                      | Przykład                         |
|------------------------------------------------------------------|------------------------------------------------------------------------------|----------------------------------|
| <code>forward(d)</code> <code>fd(d)</code>                       | idzie <b>d</b> kroków do przodu (w kierunku, w którym patrzy), rysując linię | <code>fd(100)</code>             |
| <code>backward(d)</code> <code>bk(d)</code> <code>back(d)</code> | cofa się <b>d</b> kroków, nie zmieniając kierunku                            | <code>bk(50)</code>              |
| <code>left(k)</code> <code>lt(k)</code>                          | obróć w lewo o <b>k</b> stopni                                               | <code>lt(90)</code>              |
| <code>right(k)</code> <code>rt(k)</code>                         | obróć w prawo o <b>k</b> stopni                                              | <code>rt(120)</code>             |
| <code>circle(r)</code>                                           | okrąg o promieniu <b>r</b> ; środek jest <b>r</b> kroków na lewo od żółwia   | <code>circle(50)</code>          |
| <code>circle(r, kąt)</code>                                      | tylko łuk o podanym kącie                                                    | <code>circle(50, 90)</code>      |
| <code>circle(r, steps=n)</code>                                  | wielokąt foremny o <b>n</b> bokach wpisany w okrąg                           | <code>circle(40, steps=5)</code> |
| <code>circle(-r)</code>                                          | ujemny promień: okrąg w prawo (zgodnie ze wskazówkami)                       | <code>circle(-50)</code>         |

środek po LEWEJ stronie żółwia



`circle(50)`

tylko pół okręgu (łuk)



`circle(50, 180)`

sześciokąt wpisany w okrąg



`circle(50, steps=6)`

KROK PO KROKU — KWADRAT BEZ PĘTLI

```
fd(100) # 1. bok
lt(90)  # 2. obrót o 90°
fd(100)
lt(90)
fd(100)
lt(90)
fd(100)
lt(90) # ostatni obrót: żółw patrzy jak na starcie
```

TO SAMO KRÓCEJ — PĘTLA FOR

```
for i in range(4): # powtórz 4 razy:
    fd(100)        # bok
    lt(90)         # obrót
```

**Krok po kroku — trójkąt równoboczny:** 3 razy: `fd(100)` i `lt(120)`. Skręcasz o **kąt zewnętrzny** ( $180 - 60 = 120$ ), a nie wewnętrzny  $60^\circ$ .

## 04 Skok w konkretne miejsce

| Polecenie                                                   | Co robi                                                                      | Przykład                      |
|-------------------------------------------------------------|------------------------------------------------------------------------------|-------------------------------|
| <code>goto(x, y)</code>                                     | idzie prosto do punktu (x, y); jeśli pióro jest opuszczone — rysuje linię    | <code>goto(100, 50)</code>    |
| <code>setpos(x, y)</code><br><code>setposition(x, y)</code> | to samo co <code>goto()</code>                                               | <code>setpos(0, -80)</code>   |
| <code>setx(x)</code>                                        | zmienia tylko współrzędną x (y bez zmian)                                    | <code>setx(-120)</code>       |
| <code>sety(y)</code>                                        | zmienia tylko współrzędną y (x bez zmian)                                    | <code>sety(40)</code>         |
| <code>setheading(k)</code> <code>seth(k)</code>             | ustawia kierunek na dokładnie k stopni (0 prawo, 90 góra, 180 lewo, 270 dół) | <code>seth(90)</code>         |
| <code>home()</code>                                         | wraca do (0, 0) i ustawia kierunek 0°                                        | <code>home()</code>           |
| <code>teleport(x, y)</code>                                 | skok bez rysowania linii (Python 3.12+)                                      | <code>teleport(0, 100)</code> |

Różnica: `lt(90)` obraca o 90° względem obecnego kierunku, a `seth(90)` ustawia kierunek na 90°, niezależnie od tego, gdzie żółw patrzył.

## 05 Gdzie jest żółw? — odczytywanie stanu

| Polecenie                                  | Co robi                                                   | Przykład                         |
|--------------------------------------------|-----------------------------------------------------------|----------------------------------|
| <code>position()</code> <code>pos()</code> | aktualna pozycja jako para (x, y)                         | <code>print(pos())</code>        |
| <code>xcor()</code> / <code>ycor()</code>  | sama współrzędna x / y                                    | <code>if xcor() &gt; 200:</code> |
| <code>heading()</code>                     | aktualny kierunek w stopniach                             | <code>print(heading())</code>    |
| <code>distance(x, y)</code>                | odległość żółwia od punktu (x, y)                         | <code>distance(0, 0)</code>      |
| <code>towards(x, y)</code>                 | kąt, pod jakim trzeba się ustawić, żeby patrzeć na (x, y) | <code>seth(towards(0, 0))</code> |
| <code>isdown()</code>                      | True, jeśli pióro jest opuszczone (rysuje)                | <code>print(isdown())</code>     |
| <code>isvisible()</code>                   | True, jeśli żółw jest widoczny                            | <code>isvisible()</code>         |

**Pułapka:** współrzędne po obrotach bywają ułamkowe, np. `99.99999999`. Do porównań używaj `round(xcor())` albo `abs(xcor() - 100) < 0.01`.

## 06 Pióro — kiedy rysuje, a kiedy nie

| Polecenie                                                    | Co robi                                           | Przykład                                        |
|--------------------------------------------------------------|---------------------------------------------------|-------------------------------------------------|
| <code>penup()</code> <code>pu()</code> <code>up()</code>     | podnosi pióro: żółw chodzi, ale <b>nie rysuje</b> | <code>pu()</code>                               |
| <code>pendown()</code> <code>pd()</code> <code>down()</code> | opuszcza pióro: od teraz ruch rysuje linię        | <code>pd()</code>                               |
| <code>pensize(g)</code> <code>width(g)</code>                | grubość linii w pikselach (domyślnie 1)           | <code>pensize(3)</code>                         |
| <code>pen(...)</code>                                        | kilka ustawień naraz                              | <code>pen(pensize=2,<br/>pencolor='red')</code> |

PRZESUNIĘCIE BEZ RYSOWANIA — ZAWSZE TE 3 KROKI

```
pu()          # 1. podnieś pióro
goto(-100, 0) # 2. przejdź w nowe miejsce bez śladu
pd()          # 3. opuść pióro
fd(200)       # 4. dopiero teraz rysujesz
```

## 07 Kolory i wypełnianie

| Polecenie                              | Co robi                                                                         | Przykład                           |
|----------------------------------------|---------------------------------------------------------------------------------|------------------------------------|
| <code>pencolor(k)</code>               | kolor linii                                                                     | <code>pencolor('blue')</code>      |
| <code>fillcolor(k)</code>              | kolor wypełnienia                                                               | <code>fillcolor('gold')</code>     |
| <code>color(k)</code>                  | ten sam kolor linii i wypełnienia                                               | <code>color('green')</code>        |
| <code>color(linia, wypełnienie)</code> | dwa kolory naraz                                                                | <code>color('black', 'red')</code> |
| <code>begin_fill()</code>              | start figury do wypełnienia — <b>przed</b> pierwszym ruchem                     | <code>begin_fill()</code>          |
| <code>end_fill()</code>                | koniec figury — wypełnia wszystko od <code>begin_fill()</code>                  | <code>end_fill()</code>            |
| <code>filling()</code>                 | True, jeśli jesteśmy między <code>begin_fill()</code> a <code>end_fill()</code> | <code>filling()</code>             |
| <code>bgcolor(k)</code>                | kolor tła całego okna                                                           | <code>bgcolor('black')</code>      |
| <code>colormode(255)</code>            | kolory jako liczby 0–255 zamiast 0–1                                            | <code>pencolor(255, 128, 0)</code> |

WYPEŁNIONY KWADRAT

```
color('black', 'gold')
begin_fill()
for i in range(4):
    fd(80); lt(90)
end_fill()
```

Kolor zapisujesz na 3 sposoby:

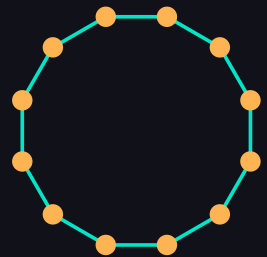
- nazwa: `'red'`, `'orange'`, `'skyblue'`
- kod hex: `'#ff8800'`
- trójka liczb: `(1, 0.5, 0)` albo po `colormode(255)`: `(255, 128, 0)`

## 08 Kropki, stemple, cofanie i szybkość

| Polecenie                         | Co robi                                                         | Przykład                    |
|-----------------------------------|-----------------------------------------------------------------|-----------------------------|
| <code>dot(średnica, kolor)</code> | kropka w miejscu żółwia (rysuje nawet przy podniesionym piórze) | <code>dot(20, 'red')</code> |
| <code>stamp()</code>              | odbija kształt żółwia na ekranie; zwraca numer stempla          | <code>s = stamp()</code>    |
| <code>clearstamp(nr)</code>       | usuwa jeden stempel o danym numerze                             | <code>clearstamp(s)</code>  |
| <code>clearstamps(n)</code>       | usuwa n pierwszych ( $n < 0$ — ostatnich; brak n — wszystkie)   | <code>clearstamps()</code>  |
| <code>undo()</code>               | cofa ostatni ruch żółwia (można wywołać wiele razy)             | <code>undo()</code>         |
| <code>speed(s)</code>             | szybkość 1 (wolno) ... 10 (szybko), <b>0 = najszybciej</b>      | <code>speed(0)</code>       |
| <code>clear()</code>              | czyści rysunek tego żółwia; pozycja i kierunek zostają          | <code>clear()</code>        |
| <code>reset()</code>              | czyści rysunek i wraca żółwia do (0, 0), kierunek 0°            | <code>reset()</code>        |

NASZYJNIK Z KROPEK

```
speed(0)           # najszybciej
for i in range(12):
    fd(30)
    dot(10, 'orange') # kropka co 30 kroków
    lt(30)           # 12 × 30° = 360°
```



12 kropek, obrót 30°

`speed()` przyjmuje też słowa: `'slowest'` (1), `'slow'` (3), `'normal'` (6), `'fast'` (10), `'fastest'` (0). Na konkursie ustaw `speed(0)` albo `tracer(0)`, żeby nie czekać na rysunek.

## 09 Wygląd żółwia

| Polecenie                                   | Co robi                                                        | Przykład                      |
|---------------------------------------------|----------------------------------------------------------------|-------------------------------|
| <code>hideturtle()</code> <code>ht()</code> | ukrywa żółwia (rysuje dalej, a szybciej)                       | <code>ht()</code>             |
| <code>showturtle()</code> <code>st()</code> | pokazuje żółwia                                                | <code>st()</code>             |
| <code>shape(nazwa)</code>                   | 'turtle', 'arrow', 'circle', 'square', 'triangle', 'classic'   | <code>shape('turtle')</code>  |
| <code>shapeseize(w, d)</code>               | powiększa kształt: szerokość, długość (turtle size to to samo) | <code>shapeseize(2, 2)</code> |
| <code>tilt(k)</code>                        | obraca sam kształt, bez zmiany kierunku ruchu                  | <code>tilt(45)</code>         |

## 10 Okno i ekran

| Polecenie                                   | Co robi                                                                     | Przykład                          |
|---------------------------------------------|-----------------------------------------------------------------------------|-----------------------------------|
| <code>setup(szer, wys)</code>               | rozmiar okna w pikselach                                                    | <code>setup(800, 600)</code>      |
| <code>title(tekst)</code>                   | napis na pasku okna                                                         | <code>title('Mój rysunek')</code> |
| <code>bgpic(plik)</code>                    | obrazek .gif jako tło                                                       | <code>bgpic('tlo.gif')</code>     |
| <code>tracer(0)</code>                      | wyłącza animację — rysunek pojawia się od razu (bardzo szybko)              | <code>tracer(0)</code>            |
| <code>update()</code>                       | odświeża ekran; <b>konieczne</b> na końcu, gdy użyto <code>tracer(0)</code> | <code>update()</code>             |
| <code>delay(ms)</code>                      | opóźnienie między klatkami animacji                                         | <code>delay(0)</code>             |
| <code>mode(m)</code>                        | 'standard' (0° w prawo) albo 'logo' (0° w górę, kąty zgodnie z zegarem)     | <code>mode('logo')</code>         |
| <code>clearscreen()</code>                  | czyści wszystko: rysunki i żółwie                                           | <code>clearscreen()</code>        |
| <code>done()</code> <code>mainloop()</code> | zostawia okno otwarte — zawsze na samym końcu programu                      | <code>done()</code>               |
| <code>exitonclick()</code>                  | zamyka okno po kliknięciu                                                   | <code>exitonclick()</code>        |
| <code>bye()</code>                          | zamyka okno od razu                                                         | <code>bye()</code>                |

## 11 Tekst i pytania do użytkownika

| Polecenie                                  | Co robi                                                          | Przykład                                            |
|--------------------------------------------|------------------------------------------------------------------|-----------------------------------------------------|
| <code>write(tekst)</code>                  | pisze tekst w miejscu żółwia                                     | <code>write('Hej!')</code>                          |
| <code>write(t, align=..., font=...)</code> | align: 'left' / 'center' / 'right'; font: (nazwa, rozmiar, styl) | <code>write('A', font=('Arial', 20, 'bold'))</code> |
| <code>textinput(tytuł, pytanie)</code>     | okienko z pytaniem; zwraca tekst (str)                           | <code>imie = textinput('?', 'Imię:')</code>         |
| <code>numinput(tytuł, pytanie)</code>      | okienko z pytaniem o liczbę; zwraca float                        | <code>n = numinput('?', 'Ile boków?')</code>        |

**Pułapka:** `numinput` zwraca liczbę zmiennoprzecinkową, np. `5.0`. Do `range()` zamień ją na całkowitą: `range(int(n))`.

## 12 Zdarzenia — mysz, klawiatura, zegar

| Polecenie                           | Co robi                                                               | Przykład                            |
|-------------------------------------|-----------------------------------------------------------------------|-------------------------------------|
| <code>onclick(f)</code>             | po kliknięciu w okno wywołuje <code>f(x, y)</code>                    | <code>onclick(goto)</code>          |
| <code>onrelease(f)</code>           | po puszczeniu przycisku myszy na żółwiu                               | <code>onrelease(stop)</code>        |
| <code>ondrag(f)</code>              | podczas przeciągania żółwia myszą                                     | <code>ondrag(goto)</code>           |
| <code>onkey(f, klawisz)</code>      | po naciśnięciu (puszczeniu) klawisza wywołuje <code>f()</code>        | <code>onkey(skok, 'space')</code>   |
| <code>onkeypress(f, klawisz)</code> | w chwili wciśnięcia klawisza (powtarza przy przytrzymaniu)            | <code>onkeypress(gora, 'Up')</code> |
| <code>listen()</code>               | okno zaczyna „słuchać” klawiatury — <b>bez tego onkey nie działa</b>  | <code>listen()</code>               |
| <code>ontimer(f, ms)</code>         | wywoła <code>f()</code> za <code>ms</code> milisekund (animacje, gry) | <code>ontimer(ruch, 50)</code>      |

### STEROWANIE ŻÓŁWIEM STRZAŁKAMI

```
def w_prawo(): rt(15)
def naprzod(): fd(10)

onkey(w_prawo, 'Right') # strzałka w prawo -> obrót
onkey(naprzod, 'Up')    # strzałka w górę -> krok
listen()               # bez listen() klawisze nie działają
done()
```

## 13 Wiele żółwi naraz

| Polecenie                | Co robi                                                                                                          | Przykład                       |
|--------------------------|------------------------------------------------------------------------------------------------------------------|--------------------------------|
| <code>Turtle()</code>    | tworzy nowego żółwia; polecenia wydajesz przez kropkę                                                            | <code>zolw = Turtle()</code>   |
| <code>zolw.fd(50)</code> | każde polecenie działa też jako metoda konkretnego żółwia                                                        | <code>zolw.lt(90)</code>       |
| <code>clone()</code>     | kopia żółwia z tą samą pozycją, kierunkiem i kolorem                                                             | <code>z2 = zolw.clone()</code> |
| <code>Screen()</code>    | obiekt okna — ma metody <code>bgcolor</code> , <code>setup</code> , <code>tracer</code> , <code>onkey</code> ... | <code>ekran = Screen()</code>  |
| <code>turtles()</code>   | lista wszystkich żółwi na ekranie                                                                                | <code>len(turtles())</code>    |

### DWA ŻÓŁWIE RYSUJĄ JEDNOCZEŚNIE

```
a = Turtle(); a.color('red')
b = Turtle(); b.color('blue'); b.lt(180)
for i in range(36):
    a.fd(5); a.lt(10)    # czerwony kręci w lewo
    b.fd(5); b.rt(10)    # niebieski - w prawo
```

## 14 Wzorce na LOGIĘ — krok po kroku

Większość zadań graficznych składa się z kilku powtarzających się pomysłów. Poniżej każdy z nich jako gotowa funkcja, a na dole — co dokładnie rysuje.

WIELOKĄT FOREMNY: SKRĘT 360 / N

```
def wielokat(n, bok):
    for i in range(n):
        fd(bok)
        lt(360 / n)
```

GWIAZDA: SKRĘT 180 - 180 / N

```
def gwiazda(n, bok):      # n nieparzyste
    for i in range(n):
        fd(bok)
        rt(180 - 180 / n)
```

SPIRALA: ZMIENNA ZMIENIA SIĘ W PĘTLI

```
def spirala(ile):
    bok = 5
    for i in range(ile):
        fd(bok)
        lt(90)
        bok += 5          # zmienna rośnie
```

ROZETA: FIGURA OBRACANA W PĘTLI

```
def rozeta(n):
    for i in range(n):
        wielokat(6, 40)  # figura...
        lt(360 / n)      # ...i obrót
```

SZLACZEK: TEN SAM MOTYW POWTARZANY W RZĘDZIE

```
def szlaczek(ile, bok):
    for i in range(ile):
        motyw(bok)      # motyw MUSI skończyć w tym samym kierunku,
                        # w którym zaczął — inaczej szlaczek się rozjedzie

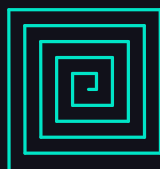
def motyw(bok):
    lt(90); fd(bok); rt(90); fd(bok)
    rt(90); fd(bok); lt(90); fd(bok)
```



wielokat(6, 40)



gwiazda(5, 80)



spirala(20)



rozeta(12)



szlaczek(5, 15)

**Złota zasada:** funkcja rysująca kawałek rysunku powinna kończyć **w tym samym kierunku** (a najlepiej i w tym samym miejscu), w którym zaczęła. Wtedy łatwo składać ją w pętli i większe rysunki. Sprawdź to: suma wszystkich obrotów w funkcji = 0° albo wielokrotność 360°.



**dobrze:** obroty 90 - 90 - 90 + 90 = 0°



**źle:** brak ostatniego lt(90) → rysunek się zawija

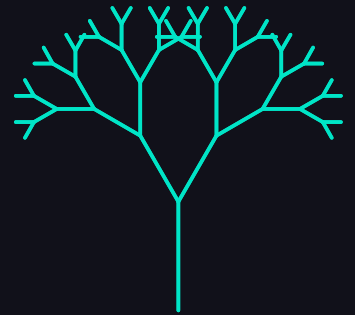
## 15 Rekurencja i fraktale

Funkcja rekurencyjna rysuje mniejszą kopię samej siebie. Zawsze potrzebuje **warunku stopu** (np. gdy poziom = 0 albo bok jest za mały) i musi wracać tam, skąd zaczęła.

DRZEWO BINARNE

```
def drzewo(dl, poziom):
    if poziom == 0:          # 1. warunek stopu
        return
    fd(dl)                   # 2. pień
    lt(30)
    drzewo(dl * 0.7, poziom - 1) # 3. lewa gałąź (mniejsza)
    rt(60)
    drzewo(dl * 0.7, poziom - 1) # 4. prawa gałąź
    lt(30)
    bk(dl)                   # 5. powrót na start – obowiązkowo!

lt(90)
drzewo(80, 6)
```



drzewo(80, 6)

- 1 Narysuj ręcznie na kartce poziom 1 i 2 — zobacz, co się powtarza.
- 2 Napisz przypadek najmniejszy ( `poziom == 0` → nic nie rób).
- 3 Narysuj „pień”, wywołaj funkcję dla mniejszych części, **wrót na start**.
- 4 Testuj od małych poziomów: 1, 2, 3 — dopiero potem większe.

## 16 Najczęstsze błędy

- 01 **Skręt o kąt wewnętrzny zamiast zewnętrznego.** Trójkąt to `lt(120)`, nie `lt(60)`; kwadrat `lt(90)`; sześciokąt `lt(60)` ( $360 / n$ ).
- 02 **Brak `pu()` przed przejściem.** Żółw zostawia niechcianą kreskę — przed `goto` podnieś pióro, a potem je opuść.
- 03 `begin_fill()` za późno. Musi być przed pierwszym `fd` figury, a `end_fill()` po ostatnim.
- 04 `tracer(0)` bez `update()`. Okno zostaje puste albo niedokończone.
- 05 **Funkcja nie wraca do kierunku startowego.** Pętle i szlaczki „rozjeżdżają się” lub kręcą w kółko.
- 06 `range(n)` liczy od 0. `for i in range(4)` daje  $i = 0, 1, 2, 3$ .
- 07 **Dzielenie daje float.**  $360 / 7 = 51.43$  — dla kątów to w porządku, ale `range()` potrzebuje liczby całkowitej: użyj `//` albo `int()`.
- 08 `left` i `right` pomyłone w lustrzanym rysunku. Rysuj najpierw jedną połowę, potem skopiuj funkcję i zamień wszystkie `lt` ↔ `rt`.

## 17 Skróty — pełne nazwy i aliasy

| Pełna nazwa              | Skrót                                   | Pełna nazwa             | Skrót                               |
|--------------------------|-----------------------------------------|-------------------------|-------------------------------------|
| <code>forward</code>     | <code>fd</code>                         | <code>penup</code>      | <code>pu</code> , <code>up</code>   |
| <code>backward</code>    | <code>bk</code> , <code>back</code>     | <code>pendown</code>    | <code>pd</code> , <code>down</code> |
| <code>left</code>        | <code>lt</code>                         | <code>pensize</code>    | <code>width</code>                  |
| <code>right</code>       | <code>rt</code>                         | <code>hideturtle</code> | <code>ht</code>                     |
| <code>setheading</code>  | <code>seth</code>                       | <code>showturtle</code> | <code>st</code>                     |
| <code>setposition</code> | <code>setpos</code> , <code>goto</code> | <code>shapsize</code>   | <code>turtlesize</code>             |
| <code>position</code>    | <code>pos</code>                        | <code>mainloop</code>   | <code>done</code>                   |

### Kwadrat

```
for i in range(4): fd(a); lt(90)
```

### Wielokąt n-kąt

skręt `360 / n`

### Gwiazda n-ramienna

skręt `180 - 180 / n` (n nieparzyste)

### Koło / łuk

```
circle(r) , circle(r, kąt)
```

### Przejście bez śladu

```
pu(); goto(x, y); pd()
```

### Wypełnienie

```
begin_fill() ... end_fill()
```

### Rysuj od razu

```
tracer(0) ... update()
```

### Okno nie znika

`done()` na końcu