



Python



Minecraft Education

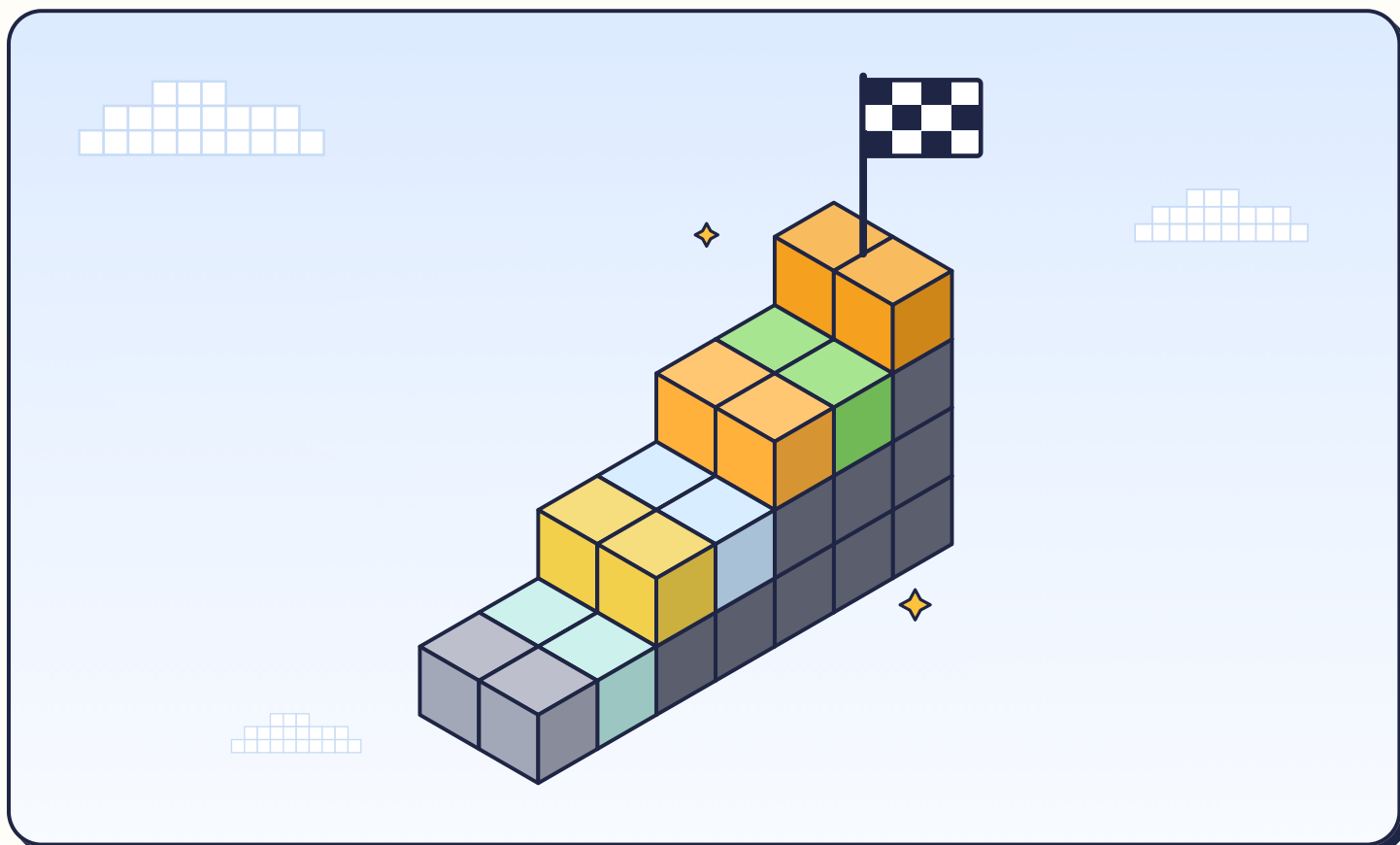


MakeCode

# Tor przeszkód

## Ściągawka młodego programisty

Wszystko z lekcji „Tor przeszkód” i „Uzupełniamy tor z AI” w jednym miejscu: pomysł na grę, najważniejsze komendy, koordynaty i słowniczek programisty.



Ta ściągawka należy do: \_\_\_\_\_

### Co znajdziesz w środku?

01

Co budujemy?

str. 2

02

Przepis na każdy etap

str. 3

03

Koordynaty

str. 4

04

Najważniejsze komendy

str. 6

05

Słowniczek programisty

str. 9

06

Błędy i pomoc AI

str. 11

07

Sprawdź się!

str. 12

## 01

## POMYŚL NA GRĘ

## Co budujemy?

Budujemy **tor przeszkód, który sam się dobudowuje!** Kod piszemy w Pythonie w MakeCode, a Minecraft Education wykonuje go w świecie gry. Każdy etap toru pojawia się dopiero wtedy, gdy dojdiesz do końca poprzedniego.

1

**Wpisz start**

Program buduje korytarz ze skały macierzystej i kładzie w podłodze linię latarni morskich.

2

**Zniszcz blok z podłogi**

Program to zauważa i dobudowuje **następny etap** tam, dokąd doszedłeś.

3

**I tak w kółko!**

Na końcu każdego etapu leży linia z **innego** bloku. Zniszcz ją, a włączysz kolejny etap.

„Każdy etap jest jak nowy pokój, który dobudowuje się na końcu poprzedniego.”

 **Mapa naszego toru**

ETAP

CO SIĘ DZIEJE?

ZNISZCZ, ŻEBY IŚĆ DALEJ

START

**Wpisujesz start na czacie**

Powstaje korytarz, a ty lądujesz na starcie z pustym ekwipunkiem.


**latarnia morska**  
SEA\_LANTERN

ETAP 1

**Kraty i creepery**

Podłoga z krat żelaznych. Gdy na nią wejdiesz, dostajesz diamentową siekierę i pojawia się 6 creeperów.


**gąbka**  
SPONGE

ETAP 2

**Deszcz zombie**

11 zombie pojawia się wysoko nad podłogą w losowych miejscach — i spada w dół!


**szkło**  
GLASS

ETAP 3

**Diamenty i TNT**

Gdy staniesz na diamentowej podłodze, z góry posypie się 11 podpalonych TNT. Uciekaj!


**jasnogłaz**  
GLOWSTONE

ETAP 4

**Losowa lawa**

11 bloków lawy w losowych miejscach rozlewa się po podłodze. Biegnij!


**blok szlamu**  
SLIME\_BLOCK

ETAP 5

**Przepaść i lewitacja**

Podłoga znika! Gdy stoisz na liniach z lazurytu, zaczynasz lewitować i możesz przelecieć dalej.


**blok miodu**  
HONEY\_BLOCK

META

**Zwycięstwo!** 🏆

Gdy zniszczysz blok miodu, przeniesiesz się za koniec toru i dostaniesz gratulacje na czacie.


**Koniec gry!**

🔑 Bloki z prawej kolumny tworzą w podłodze pas — **linię aktywacyjną**. W każdym etapie leży ona na jego końcu. Zniszcz dowolny jej blok, a włączysz kolejny etap.

# Przepis na każdy etap

Wszystkie etapy robimy według jednego przepisu, czyli **szablonu**. Kopiuj, zmieniaj, testuj!

## A Start programu: zapamiętaj, gdzie stoisz

```
pozycja = player.position()
x = pozycja.get_value(Axis.X)
y = pozycja.get_value(Axis.Y)
z = pozycja.get_value(Axis.Z)

def tor():
    blocks.fill(BEDROCK, world(x-10,y-1,z-2), world(x+10,y+10,z+35), FillOperation.HOLLOW)
    blocks.fill(SEA_LANTERN, world(x-9,y-1,z+3), world(x+9,y-1,z+5))
    player.teleport(world(x, y, z))
    player.execute("/clear")

player.on_chat("start", tor)
```

📍 **x, y, z** to **zmienne** z twoją pozycją na starcie. Są poza funkcjami, bo używamy ich w **całym** kodzie.

## B Funkcja etapu ma zawsze 3 części

```
def etap2(z):
    player.say("ETAP 2")
    # 1. Nowy kawałek korytarza - zawsze ten sam kod
    blocks.fill(BEDROCK, world(x-10,y-1,z-2), world(x+10,y+10,z+35), FillOperation.HOLLOW)
    # 2. Linia aktywacyjna - włączy NASTĘPNY etap
    blocks.fill(GLASS, world(x-9,y-1,z+25), world(x+9,y-1,z+27))
    # 3. Przeszkoda - deszcz zombie!
    for i in range(11):
        mobs.spawn(ZOMBIE, randpos(world(x-8,y+3,z+3), world(x+8,y+3,z+20)))
```

**1** Nowy kawałek korytarza

**2** Linia aktywacyjna (nowy blok!)

**3** Przeszkoda — twój pomysł!

## C Aktywacja — piszemy ją na samym dole kodu

```
def zniszczenie_bloku_e2():
    pozycja = player.position()
    z = pozycja.get_value(Axis.Z)
    etap2(z)

blocks.on_block_broken(SPONGE, zniszczenie_bloku_e2)
```

- 1 Niszczysz **gąbkę** z końca etapu 1.
- 2 Minecraft uruchamia funkcję `zniszczenie_bloku_e2`.
- 3 Ta funkcja bierze twoje **z** i woła `etap2(z)`.
- 4 Etap 2 buduje się tam, dokąd doszedłeś!

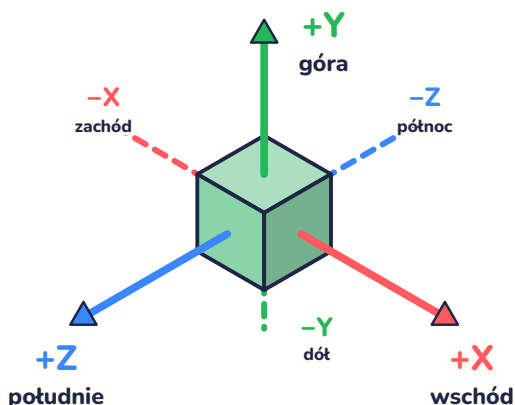
### ✓ Robisz nowy etap? Zmień 4 rzeczy:

- ☐ numer w nazwie funkcji, np. `etap6`
- ☐ napis: `player.say("ETAP 6")`
- ☐ **blok linii aktywacyjnej** — inny niż we wszystkich wcześniejszych etapach
- ☐ **przeszkodę** — wymyśl coś swojego!

...i dopisz na dole kodu aktywację: `on_block_broken` z blokiem z **poprzedniego** etapu.

# Koordynaty

Świat Minecrafta to wielka siatka bloków. Każdy blok ma swój **adres** z trzech liczb: **x**, **y** i **z**. Zapisujemy je zawsze w tej kolejności: **(x, y, z)**.



**W bok:** na wschód (+) albo na zachód (-).



**Wysokość:** w górę (+) albo w dół (-). Im większe y, tym wyżej.



**Do przodu i do tyłu:** na południe (+) albo na północ (-). Nasz tor rośnie w stronę +Z.

**pos(x, y, z)**

LICZYSZ OD SIEBIE

Pozycja **względna** — liczysz od miejsca, w którym stoisz. **pos(0, 0, 0)** to twoje stopy.

- ▶ **pos(0, -1, 0)** → blok pod twoimi nogami
- ▶ **pos(0, 3, 0)** → 3 bloki nad tobą
- ▶ **pos(5, 0, 0)** → 5 bloków na wschód od ciebie

**world(x, y, z)**

STAŁY ADRES

Pozycja **stała** — jak adres domu. Wskazuje zawsze to samo miejsce na mapie, nieważne, gdzie stoisz.

- ▶ **world(0, 4, 0)** → zawsze ten sam blok mapy
- ▶ **world(x, y, z)** → start toru (x, y, z to nasze zmienne ze startu programu)

**Jak przeczytać world(x-9, y-1, z+25)?**

**x-9**

9 bloków na **zachód** od miejsca startu

**y-1**

1 blok **niżej** niż twoje stopy — to poziom **podłogi**

**z+25**

25 bloków na **południe** od początku etapu

**Rozszyfruj! Gdzie jest ten blok?**

1 **pos(0, 2, 0)**

2 **pos(0, 0, 10)**

3 **world(x, y+10, z)**

4 **world(x+9, y-1, z+27)**

Odpowiedzi: 1) blok tuż nad twoją głową • 2) 10 bloków dalej w stronę +Z — tam przenosi cię funkcja zwycięstwa • 3) sufit korytarza nad miejscem startu • 4) róg linii aktywacyjnej przy wschodniej ścianie (podpowiedź: zajrzyj na stronę 5)

**Włącz współrzędne!** W ustawieniach świata zaznacz „Pokaż współrzędne” — zobaczysz na ekranie swoją pozycję x, y, z.

**Ciekawostka:** **pos(0, 5, 0)** w kodzie to to samo co **~ ~5 ~** w komendach na czacie, np. **/tp ~ ~5 ~**.

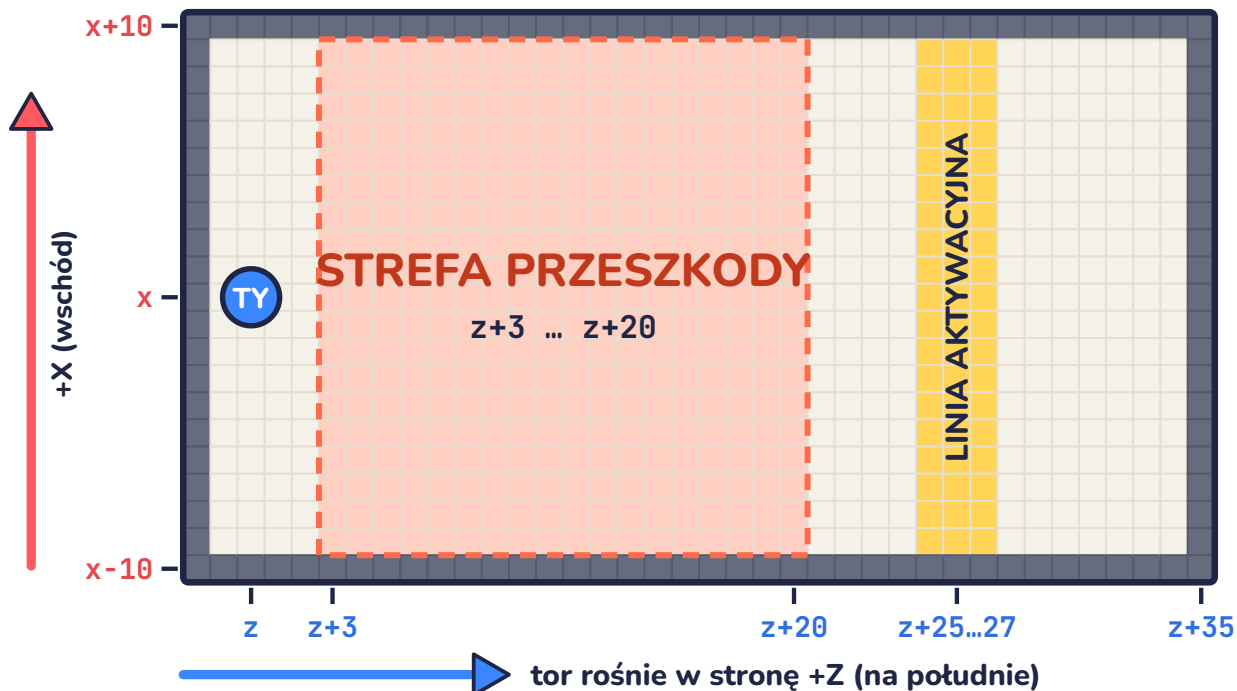
# Jak zbudowany jest etap?

Tak wygląda jeden etap naszego toru. Porównaj rysunki z kodem — każda liczba w `world(...)` ma na nich swoje miejsce!



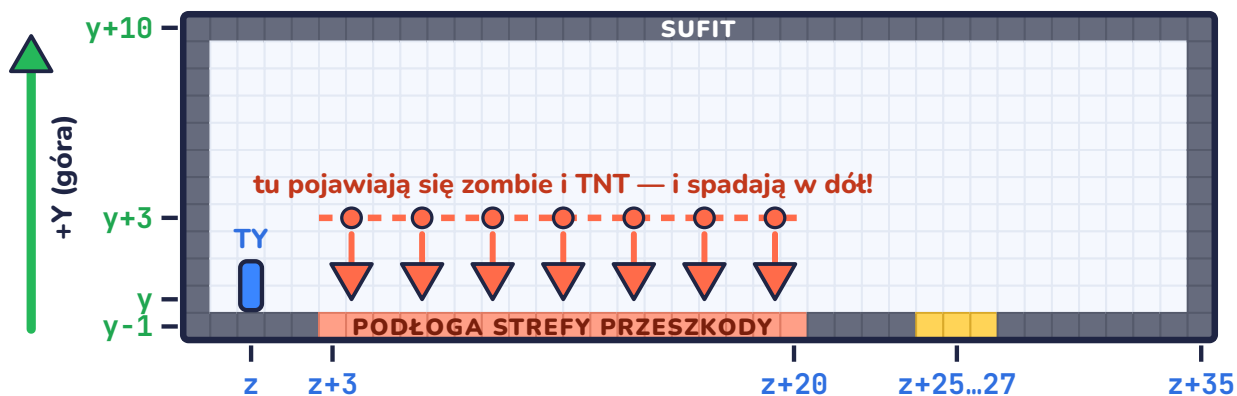
## Widok z góry

każdy kwadracik = 1 blok



## Widok z boku

przekrój korytarza · każdy kwadracik = 1 blok



### Kod z funkcji etapu

### Co buduje?

`world(x-10, y-1, z-2) → world(x+10, y+10, z+35)`

korytarz: ściany, podłogę i sufit (**HOLLOW**)

`world(x-9, y-1, z+3) → world(x+9, y-1, z+20)`

podłogę strefy przeszkody, np. kraty

`world(x-8, y+3, z+3) → world(x+8, y+3, z+20)`

obszar, w którym losujemy miejsca dla zombie i TNT

`world(x-9, y-1, z+25) → world(x+9, y-1, z+27)`

linię aktywacyjną następnego etapu

# Najważniejsze komendy

Każda komenda jest zbudowana tak samo: **grupa**, kropka, **polecenie**, a w nawiasie **argumenty** — czyli dane, których polecenie potrzebuje. Argumenty oddzielamy przecinkami.

```
mobs.spawn(ZOMBIE, pos(0, 3, 5))
```

- 1 **grupa** — mobs to moby i przedmioty
- 2 **polecenie** — spawn znaczy „przywołaj”
- 3 **1. argument** — kogo? zombie
- 4 **2. argument** — gdzie? 3 bloki nad tobą, 5 w stronę +Z

 **player** — gracz
  **blocks** — bloki
  **mobs** — moby i przedmioty
  **gameplay** — rozgrywka



## Gracz **player**

### player.say

pisze na czacie

```
player.say("ETAP 1")
```

- 1 tekst w cudzysłowie — pojawi się na czacie

💡 Dobry sposób, żeby sprawdzić, czy kod doszedł do danego miejsca.

### player.teleport

przenosi gracza

```
player.teleport(pos(0, 0, 10))
```

- 1 dokąd — tu 10 bloków dalej w stronę +Z

💡 W torze  
 player.teleport(world(x, y, z)) przenosi cię na start.

### player.on\_chat

uruchamia funkcję, gdy wpiszesz słowo na czacie

```
def tor():
    player.say("Buduję tor!")

player.on_chat("start", tor)
```

- 1 słowo, które wpisujesz na czacie (bez ukośnika /)
- 2 nazwa funkcji do uruchomienia — **bez nawiasów!** Minecraft sam ją uruchomi, gdy wpiszesz słowo.

💬 Czat otwierasz klawiszem **T**. Wpisz **start** i naciśnij Enter.

### player.position

gdzie stoisz?

```
pozycja = player.position()
z = pozycja.get_value(Axis.Z)
```

- 1 którą liczbę wyciągnąć z pozycji: Axis.X, Axis.Y albo Axis.Z

### player.execute

komenda Minecrafta

```
player.execute("/clear")
```

- 1 komenda jak na czacie — /clear czyści ekwipunek

🔧 Do testów: /gamemode 1 = tryb kreatywny, /gamemode 0 = przetrwanie.

## blocks.place

stawia JEDEN blok

```
blocks.place(DIAMOND_BLOCK, pos(0, 0, 2))
```

- 1 jaki blok
- 2 gdzie — tu 2 bloki od ciebie w stronę +Z

Z naszego toru (etap 4): lava w losowym miejscu — 11 razy, w pętli `for`.

```
blocks.place(LAVA, randpos(world(x-9,y+1,z+5), world(x+9,y+1,z+20)))
```

## blocks.fill

wypełnia blokami całą bryłę

Ten przykład stawia szklaną ścianę 5 × 4 bloki, 3 bloki przed tobą (w stronę +Z).

```
blocks.fill(GLASS, pos(-2, 0, 3), pos(2, 3, 3))
```

- 1 jaki blok
- 2 pierwszy róg
- 3 drugi róg — ten naprzeciwko, po skosie
- 4 (nieobowiązkowy) sposób wypełniania — patrz niżej

 Licz razem z rogami! Od x = -2 do x = 2 jest 5 bloków: -2, -1, 0, 1, 2.

REPLACE



HOLLOW



- |         |                                                          |
|---------|----------------------------------------------------------|
| REPLACE | wypełnia wszystko (tak też, gdy nie podasz 4. argumentu) |
| HOLLOW  | ściany z bloku, a w środku powietrze — nasz korytarz!    |
| OUTLINE | tylko ściany, środka nie rusza                           |
| KEEP    | wypełnia tylko puste miejsca                             |
| DESTROY | jak REPLACE, ale stare bloki wypadają jak wykopane       |

```
blocks.fill(BEDROCK, world(x-10,y-1,z-2), world(x+10,y+10,z+35), FillOperation.HOLLOW)
```

## blocks.test\_for\_block

sprawdza, jaki blok jest w danym miejscu

```
if blocks.test_for_block(IRON_BARS, pos(0, -1, 0)):
    player.say("Stoisz na kratkach!")
```

- 1 jakiego bloku szukamy
- 2 gdzie patrzymy — tu pod twoimi nogami

 Odpowiada **True** (prawda) albo **False** (fałsz) — dlatego pasuje do `if` i `while`.

## blocks.on\_block\_broken

reaguje na zniszczenie bloku

```
blocks.on_block_broken(SPONGE, zniszczenie_bloku_e2)
```

- 1 jaki blok trzeba zniszczyć
- 2 jaką funkcję wtedy uruchomić — **bez nawiasów**

 Dzięki niemu działa nasz tor: każdy etap ma linię aktywacyjną z innego bloku.



## Moby i przedmioty

mobs

### mobs.spawn

przywołuje moba

```
mobs.spawn(ZOMBIE, pos(0, 3, 5))
```

1 kogo — zwierzę, potwora, a nawet TNT

2 gdzie — tu 3 bloki w górę i 5 w stronę +Z



W etapie 3 przywołujemy `PRIMED_TNT` — to TNT, które **już się pali!**

! Na poziomie trudności „**Pokojowy**” wrogie moby (zombie, creeper) się nie pojawiają.

### mobs.give

daje przedmiot do ekwipunku

```
mobs.give(mobs.target(NEAREST_PLAYER), DIAMOND_AXE, 1)
```

1 komu — `mobs.target(NEAREST_PLAYER)` to najbliższy gracz, czyli ty

2 jaki przedmiot

3 ile sztuk — przy 0 nic nie dostaniesz!

### mobs.apply\_effect

nakłada efekt

```
mobs.apply_effect(LEVITATION, mobs.target(NEAREST_PLAYER), 2, 1)
```

1 jaki efekt

2 na kogo

3 na ile sekund

4 siła: 0 = poziom I, 1 = poziom II...

⚡ szybkość  
SPEED

🌙 widzenie w ciemności  
NIGHT\_VISION

📍 lewitacja  
LEVITATION

🐢 spowolnienie  
SLOWNESS



Siłę można pominąć — tak jest w kilku miejscach naszego kodu.



### randpos

losuje miejsce w obszarze

```
mobs.spawn(ZOMBIE, randpos(pos(-5, 3, 2), pos(5, 3, 10)))
```

1 jeden róg obszaru

2 drugi róg obszaru



Za każdym razem wychodzi inne miejsce! `randpos(...)` wpisujesz tam, gdzie zwykle jest `pos(...)`.



### gameplay.title

bonus: wielki napis na ekranie

```
gameplay.title(mobs.target(NEAREST_PLAYER), "TOR UKOŃCZONY!", "Brawo!")
```

1 komu pokazać

2 duży napis

3 mniejszy napis pod spodem

# Słowniczek programisty

Najważniejsze słowa, które poznaliśmy do tej pory. Wracaj tu, kiedy coś wyleci ci z głowy!



## Zmienna

Pudełko z naklejką. Na naklejce jest **nazwa**, a w środku **wartość**. Wartość można zmienić.

```
liczba_zombie = 11
z = pozycja.get_value(Axis.Z)
```



## Funkcja def

Nazwany zestaw poleceń — jak przepis. Zapisujesz raz, a uruchamiasz nazwą z nawiasami.

```
def tor():
    player.say("Start!")

tor() # uruchomienie funkcji
```



## Parametr

Dane, które podajesz funkcji w nawiasie. Dzięki nim ta sama funkcja buduje etap w różnych miejscach.

```
def etap1(z): # z to parametr
    player.say("ETAP 1")

etap1(z) # podajemy wartość z
```



## Pętla for

Powtarza kod określoną liczbę razy. `range(6)` daje liczby 0, 1, 2, 3, 4, 5 — czyli **6 powtórzeń**.

```
for i in range(6):
    mobs.spawn(CHICKEN, pos(0, 1, 5))
```

`range(2, 6)` → 2, 3, 4, 5 · `range(0, 10, 2)` → 0, 2, 4, 6, 8



## and • or • not

**and** — oba warunki muszą być prawdziwe („deszcz I wiatr”).  
**or** — wystarczy jeden („deszcz LUB wiatr”).  
**not** — odwraca odpowiedź: prawda ↔ fałsz.



## True i False

Prawda i fałsz. Każdy warunek ma jedną z tych dwóch odpowiedzi. Na przykład `test_for_block` odpowiada **True**, gdy blok się zgadza.



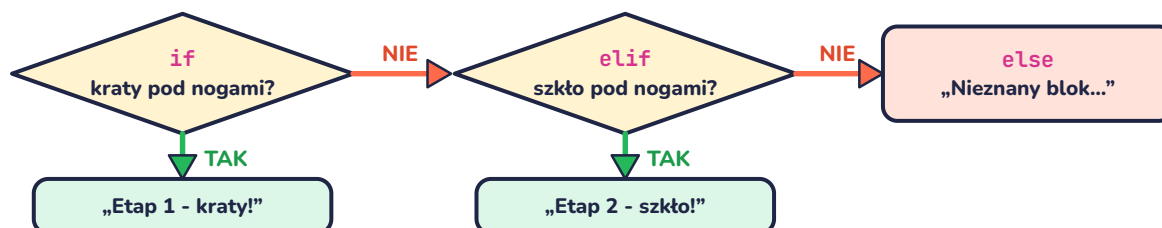
## Instrukcja warunkowa if / elif / else

```
if blocks.test_for_block(IRON_BARS, pos(0, -1, 0)):
    player.say("Etap 1 - kraty!")
elif blocks.test_for_block(GLASS, pos(0, -1, 0)):
    player.say("Etap 2 - szkło!")
else:
    player.say("Nieznany blok pod nogami")
```

Rozwidlenie drogi: **JEŻELI** warunek jest prawdziwy → zrób to. **W PRZECIWNYM RAZIE** → zrób coś innego.

- ▶ **if** — jeżeli...
- ▶ **elif** — a jeśli nie, to może...
- ▶ **else** — w każdym innym przypadku

TAK TO „MYŚLI” KOMPUTER:





## while True + break

Pętla, która kręci się bez końca i **czeka**, aż coś się stanie. **break** to STOP — wychodzi z pętli.

```
while True:
    if blocks.test_for_block(DIAMOND_BLOCK, pos(0, -1, 0)):
        player.say("Uwaga, TNT!")
        break
```



## while not

Pętla kręci się, **dopóki** warunek **nie** jest prawdziwy. Gdy staniesz na bloku miodu — kończy się sama.

```
while not blocks.test_for_block(HONEY_BLOCK, pos(0, -1, 0)):
    if blocks.test_for_block(LAPIS_LAZULI_BLOCK, pos(0, -1, 0)):
        mobs.apply_effect(LEVITATION, mobs.target(NEAREST_PLAYER), 2, 1)
```

→ **Wcięcie** 4 spacje na początku linii. Pokazują, co jest **w środku** funkcji, pętli albo **if**. Złe wcięcie = błąd!

: **Dwukropek** stawiamy na końcu linii z **def**, **if**, **elif**, **else**, **for** i **while**.

# **Komentarz** notatka dla ludzi. Komputer ją pomija, ale tobie pomaga zrozumieć kod.

⚡ **Zdarzenie** coś, co dzieje się w grze: słowo na czacie albo zniszczony blok. Zdarzenie może uruchomić funkcję.



**Nazwy z gry** SPONGE, ZOMBIE, DIAMOND\_AXE  
piszemy WIELKIMI literami i bez literówek.



**Bug** to błąd w kodzie, a **debugowanie** to szukanie i naprawianie bugów.



## Minecraft w pigułce

| Chcę...                            | Komenda — przykład                                                        |
|------------------------------------|---------------------------------------------------------------------------|
| napisać coś na czacie              | <code>player.say("Hej!")</code>                                           |
| uruchomić kod słowem z czatu       | <code>player.on_chat("start", tor)</code>                                 |
| postawić jeden blok                | <code>blocks.place(STONE, pos(0, 0, 2))</code>                            |
| wypełnić obszar blokami            | <code>blocks.fill(GLASS, pos(-2, 0, 3), pos(2, 3, 3))</code>              |
| przywołać moba                     | <code>mobs.spawn(CHICKEN, pos(0, 0, 3))</code>                            |
| dostać przedmiot                   | <code>mobs.give(mobs.target(NEAREST_PLAYER), DIAMOND_AXE, 1)</code>       |
| dostać efekt                       | <code>mobs.apply_effect(SPEED, mobs.target(NEAREST_PLAYER), 10, 1)</code> |
| przenieść się w inne miejsce       | <code>player.teleport(pos(0, 0, 10))</code>                               |
| sprawdzić blok pod nogami          | <code>blocks.test_for_block(GLASS, pos(0, -1, 0))</code>                  |
| zareagować na zniszczony blok      | <code>blocks.on_block_broken(HONEY_BLOCK, zwyciestwo)</code>              |
| wylosować miejsce                  | <code>randpos(pos(-5, 0, 2), pos(5, 0, 10))</code>                        |
| sprawdzić, gdzie stoję             | <code>player.position().get_value(Axis.Z)</code>                          |
| podać miejsce od siebie / na mapie | <code>pos(0, -1, 0) · world(0, 4, 0)</code>                               |

# Błędy i pomoc AI

Każdy programista robi błędy — nawet najlepsi. Szukanie i naprawianie ich to **debugowanie**. Najpierw szukaj sam, a gdy utkniesz, poproś o pomoc trenera albo AI.



## Najczęstsze błędy

- 1 Brak dwukropka : na końcu linii z `def`, `if`, `for`, `while`.
- 2 Złe wcięcie — za małe albo za duże.
- 3 Literówka w nazwie: `SPONG` zamiast `SPONGE`.
- 4 Pomyłona zmienna: `y` zamiast `z`.
- 5 Zła liczba, np. 0 sztuk przedmiotu.
- 6 Nawiasy w `on_chat`: piszemy `tor`, a nie `tor()`.
- 7 Wielkie i małe litery: `Player` to nie to samo co `player`.
- 8 Ten sam blok aktywacyjny w dwóch etapach — etapy się mieszają.



## Zagadka: znajdź 6 błędów!

Odpowiedzi: strona 12 (do góry nogami 😊)

```
def etap1(z):
    player.say("ETAP 1")
    blocks.fill(BEDROCK, world(x-10,y-1,z-2), world(x+10,y+10,z+35), FillOperation.HOLLOW)
    blocks.fill(SPONG, world(x-9,y-1,z+25), world(x+9,y-1,z+27))
    blocks.fill(IRON_BARS, world(x-9,y-1,z+3), world(x+9,y-1,z+20))
    while True
    if blocks.test_for_block(IRON_BARS, pos(0, -1, 0)):
        mobs.give(mobs.target(NEAREST_PLAYER), DIAMOND_AXE, 0)
        for i in range(6):
            mobs.spawn(CREEPER, randpos(world(x-8,y+1,z+3), world(x+8,y+1,y+20)))
        break
```



## AI — twój pomocnik

AI może pomóc wymyślić przeszkodę, napisać kod i znaleźć błąd. Pamiętaj jednak: **AI to narzędzie, a nie gotowa odpowiedź**. Im lepiej opiszesz problem, tym lepszą dostaniesz pomoc!

### Jak dobrze zapytać?

1. Napisz, w czym programujesz: *MakeCode Python w Minecraft Education*.
2. Wklej swój kod.
3. Opisz, co nie działa, i wklej komunikat błędu.
4. Poproś o wyjaśnienie **prostymi słowami**.

### Zasady

- Najpierw spróbuj sam — to najlepszy trening.
- Sprawdź i przetestuj kod od AI — AI też się myli!
- Nie podawaj AI swoich danych: nazwiska, adresu, haseł.
- Korzystaj z AI na zajęciach albo razem z dorosłym.

### GDY COŚ NIE DZIAŁA

„Uczę się Pythona w MakeCode dla Minecraft Education. Ten kod nie działa — wyświetla błąd: <tu wklej błąd>. Czy możesz znaleźć błąd i wyjaśnić go prostymi słowami?”

### GDY SZUKASZ POMYSŁU

„Tworzę grę w Minecraft Education z MakeCode Python. Mam etap toru przeszkód: <tu wklej kod>. Chcę zmienić przeszkodę — zaproponuj 4 pomysły, które używają podobnych technik.”

## 07

## QUIZ

## Sprawdź się!

Odpowiedz na pytania, a potem obróć kartkę i sprawdź odpowiedzi na dole strony.

1 Co oznacza `pos(0, -1, 0)`?

.....

2 Czym różni się `blocks.place` od `blocks.fill`?

.....

3 Ile razy wykona się pętla `for i in range(4):`? Jakie liczby przyjmie `i`?

.....

4 Czym różni się `and` od `or`?

.....

5 Do czego służy `break` w pętli `while True`?

.....

6 Dlaczego w `player.on_chat("start", tor)` nie ma nawiasów po `tor`?

.....



**Brawo, programisto!**

Twój tor ma już 5 etapów. Wymyśl szósty — przepis znajdziesz na stronie 3.

**Błądy z zagadek!** 🤖

1. `SPONGE` → `SPONGE` (literówka).
2. Linia z `IRON_BARS` ma za duże wcięcie.
3. Po `while True` brakuje dwukropka.
4. `if` musi mieć wcięcie — jest w środku pętli.
5. `mobs.give` daje 0 sztuk — powinno być 1.
6. W `randpos` jest `y+20` zamiast `z+20`.
1. Blok tuż pod twoimi nogami!
2. `place` stawia jeden blok, `fill` wypełnia cały obszar między dwoma rogami.
3. 4 razy: `i = 0, 1, 2, 3`.
4. `and` — oba warunki muszą być prawdziwe, `or` — wystarczy jeden.
5. `break` przerywa pętlę — bez niego kręciłaby się bez końca.
6. Nie uruchamiamy funkcji od razu — podajemy jej nazwę, a Minecraft uruchomi ją, gdy wpiszesz start.