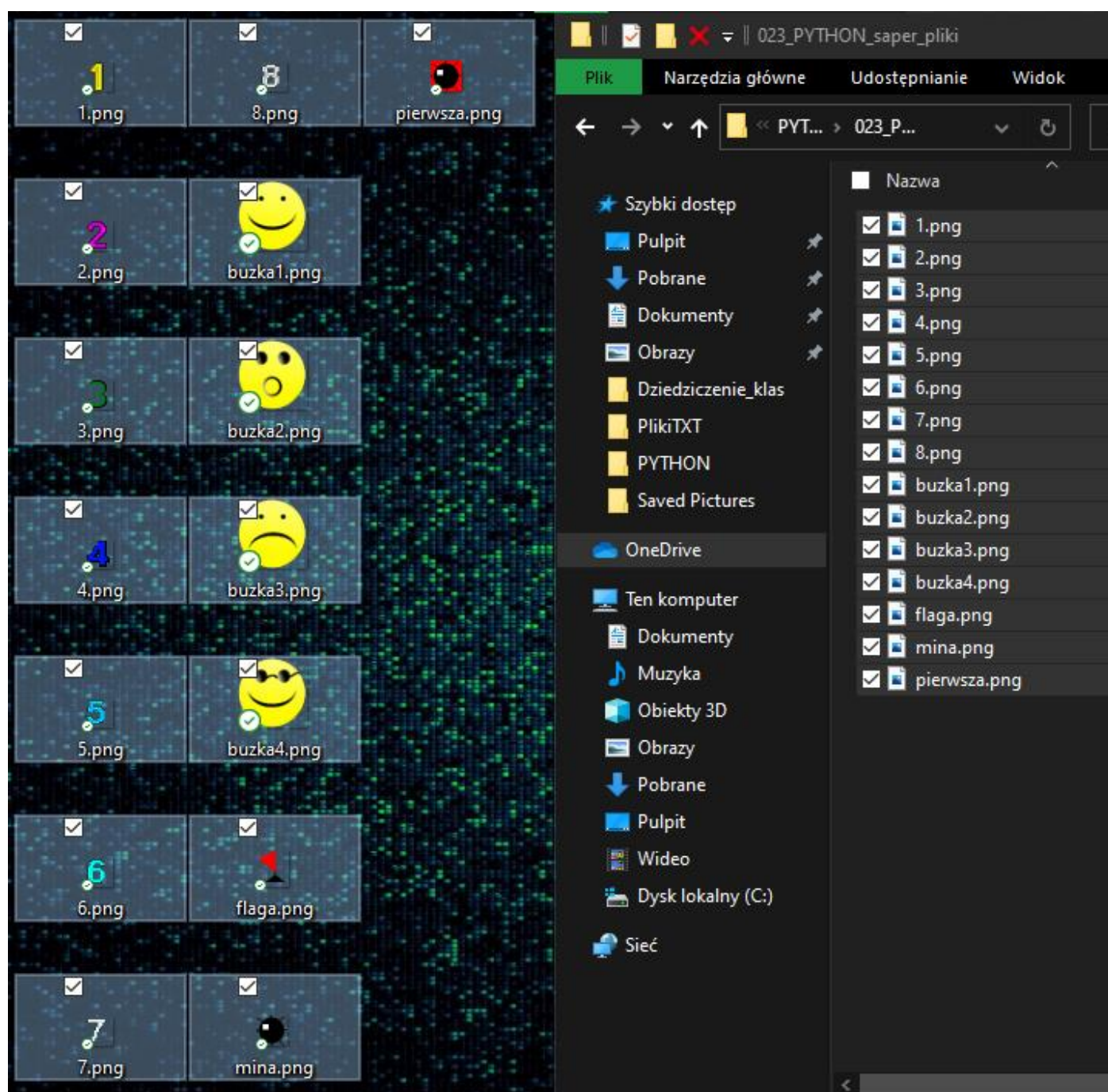


Python – gra Saper

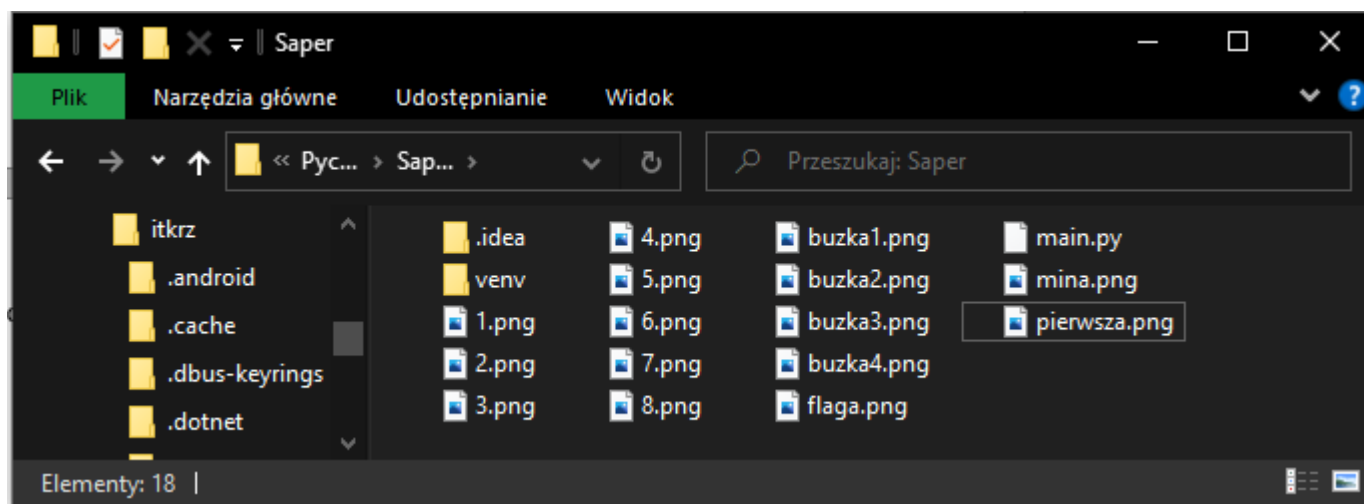
Przed rozpoczęciem wykonywania projektu należy przygotować grafiki:



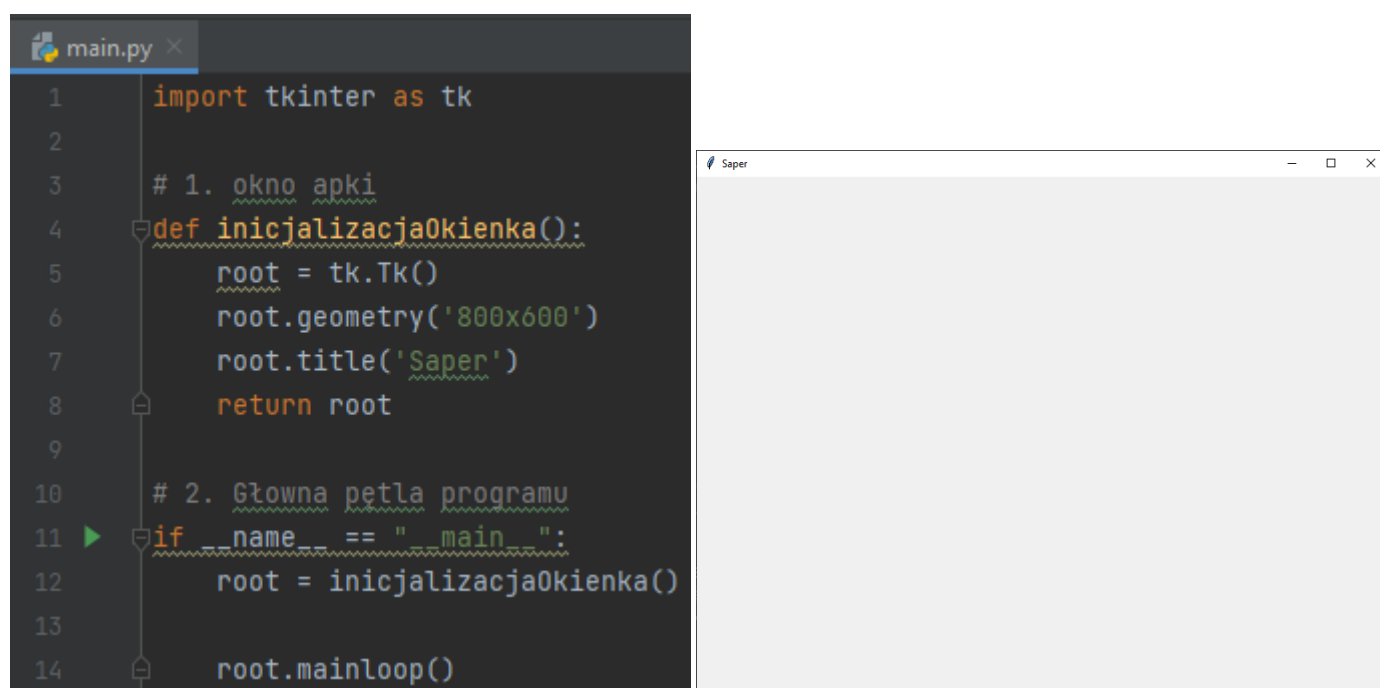
Właściwości grafik:

	Obraz			Obraz			Obraz		
	Wymiary			Wymiary			Wymiary		
1..8 oraz miny:	Szerokość	20 pikseli	; buźki 1..4 :	Szerokość	60 pikseli	; flaga :	Szerokość	22 pikseli	.
	Wysokość	20 pikseli		Wysokość	60 pikseli		Wysokość	25 pikseli	
	Głębina w bitach	32		Głębina w bitach	32		Głębina w bitach	32	

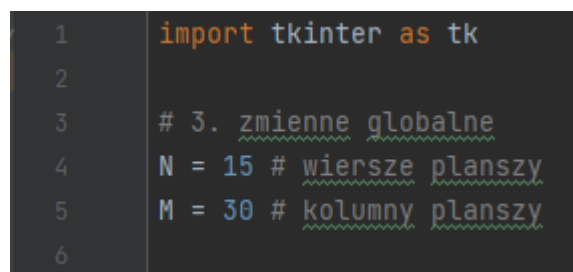
Po przygotowaniu grafik można przejść do tworzenia kodu gry. Po uruchomieniu projektu należy do niego dodać grafiki:



Inicjalizujemy tkinter-a oraz tworzymy okno aplikacji oraz główny program zawierający pętlę:



Dodajemy zmienne globalne które będą określały wielkość panelu dolnego z minami oraz szerokość panelu górnego:



Dodajemy górny panel programu, który będzie wyświetlał buźki oraz wyniki programu:


```
14 # 4. górny panel
15 def inicjalizacjaGornegoPanela(root):
16     licznik_min = tk.Label(root)
17     licznik_min.grid(row = 0, column = 0)
18     licznik_min['text'] = '0000'
19     buzka = tk.Button(root)
20     buzka.grid(row = 0, column = (M//2 - 1))
21     zegar = tk.Label(root)
22     zegar.grid(row = 0, column = (M - 6))
23     zegar['text'] = '0001'
24     gorny_panel = [licznik_min, buzka, zegar]
25     return gorny_panel
```

Oraz dodajemy funkcję do pętli głównej:

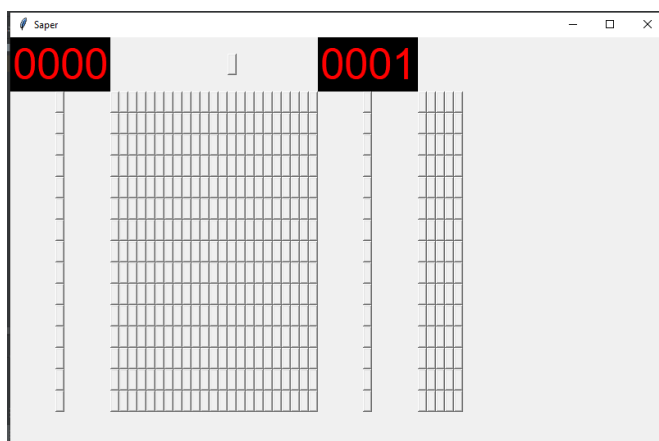
```
28 if __name__ == "__main__":
29     root = inicjalizacjaOkienka()
30     gorny_panel = inicjalizacjaGornegoPanela(root)
31     root.mainloop()
```

Modyfikujemy wygląd panu górnego:

```
14 # 4. górny panel
15 def inicjalizacjaGornegoPanela(root):
16     licznik_min = tk.Label(root, bg = '#000000', fg = '#FF0000', font = ('Digital-7', 40))
17     licznik_min.grid(row = 0, column = 0)
18     licznik_min['text'] = '0000'
19     buzka = tk.Button(root)
20     buzka.grid(row = 0, column = (M//2 - 1))
21     zegar = tk.Label(root, bg = '#000000', fg = '#FF0000', font = ('Digital-7', 40))
22     zegar.grid(row = 0, column = (M - 6))
23     zegar['text'] = '0001'
24     gorny_panel = [licznik_min, buzka, zegar]
25     return gorny_panel
26
```



Tworzymy planszę z kwadracikami do klikania. Wówczas elementy górnego panelu się rozsuną (na początku nie przejmujemy się tym, że rozsuniecie nas nie zadowoli (zmodyfikujemy je w następnym punkcie):



```

27 # 5. panel z polami do klikania
28 def inicjalizacjaPlanszy(root):
29     przyciski = [tk.Button(root) for i in range(N*M)]
30     #rozmieszczamy przyciski:
31     for i in range(N):
32         for j in range(M):
33             przyciski[i*M+j].grid(row = (i+1), column = j)
34     return przyciski
35
36 # 2. Główna pętla programu
37 if __name__ == "__main__":
38     root = inicjalizacjaOkienka()
39     gorny_panel = inicjalizacjaGornegoPanela(root)
40     przyciski = inicjalizacjaPlanszy(root)
41     root.mainloop()

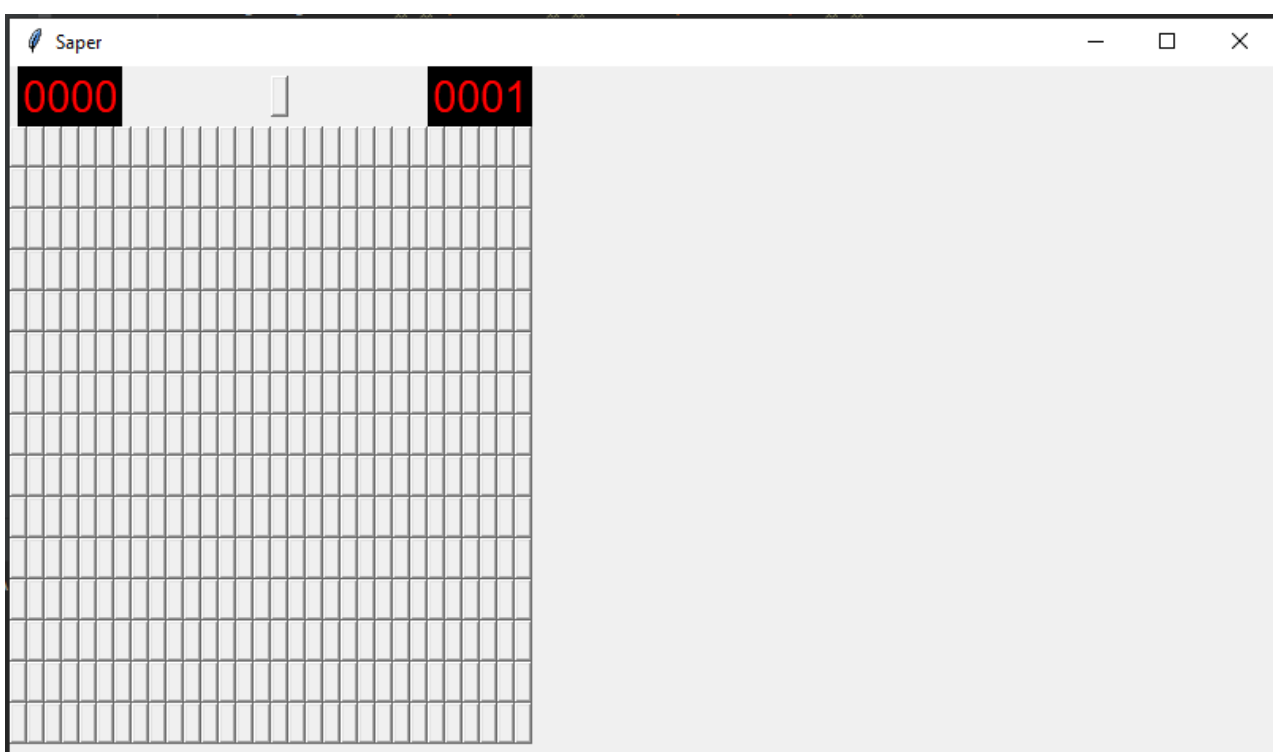
```

Modyfikujemy rozsuniecie elementów górnego panelu za pomocą `columnspan` w definicji panelu górnego(Uwaga: wielkość czcionki może wpłynąć na rozsuniecie kwadracików/prostokątów):

```

14 # 4. górny panel
15 def inicjalizacjaGornegoPanela(root):
16     licznik_min = tk.Label(root, bg = '#000000', fg = '#FF0000', font = ('Digital-7', 20))
17     licznik_min.grid(row = 0, column = 0, columnspan = 7)
18     licznik_min['text'] = '0000'
19     buzka = tk.Button(root)
20     buzka.grid(row = 0, column = (M//2 - 1), columnspan = 3)
21     zegar = tk.Label(root, bg = '#000000', fg = '#FF0000', font = ('Digital-7', 20))
22     zegar.grid(row = 0, column = (M - 6), columnspan = 7)
23     zegar['text'] = '0001'
24     gorny_panel = [licznik_min, buzka, zegar]
25     return gorny_panel

```



Modyfikując wygląd dodajemy marginesy (ipadx – wewnętrzny w poziomie; pady – zewnętrzny w pionie)

```
14 # 4. górny panel
15 def inicjalizacjaGornegoPanela(root):
16     licznik_min = tk.Label(root, bg = '#000000', fg = '#FF0000', font = ('Digital-7', 20))
17     licznik_min.grid(row = 0, column = 0, columnspan = 7, ipadx = 10, pady = 30)
18     licznik_min['text'] = '0000'
19     buzka = tk.Button(root)
20     buzka.grid(row = 0, column = (M//2 - 1), columnspan = 3, pady = 30)
21     zegar = tk.Label(root, bg = '#000000', fg = '#FF0000', font = ('Digital-7', 20))
22     zegar.grid(row = 0, column = (M - 6), columnspan = 7, ipadx = 10, pady = 30)
23     zegar['text'] = '0001'
24     gorny_panel = [licznik_min, buzka, zegar]
25     return gorny_panel
```

Oraz lewy margines całego panelu dolnego:

```
27 # 5. panel z polami do klikania
28 def inicjalizacjaPlanszy(root):
29     przyciski = [tk.Button(root) for i in range(N*M)]
30     #rozmieszczamy przyciski:
31     for i in range(N):
32         for j in range(M):
33             if j == 0:
34                 przyciski[i*M+j].grid(row = (i+1), column = j, padx = (20, 0))
35             else:
36                 przyciski[i*M+j].grid(row = (i+1), column = j)
37     return przyciski
```

Ogólny wygląd gotowy.

Obsługa zegara:

Tworzymy zmienną globalną początku gry

```
3 # 3. zmienne globalne
4 N = 15 # wiersze planszy
5 M = 30 # kolumny planszy
6 CZAS = 0 # czas rozpoczęcia gry
```

Tworzymy funkcję modyfikującą czas:

```
40 # 6. Zegar czasu gry
41 def aktualizujZegar(root, zegar):
42     global CZAS # słowo 'global' umożliwia modyfikowanie zmiennej a nie tylko odczyt
43     CZAS += 1
44     zegar["text"] = "0" * (4 - len(str(CZAS))) + str(CZAS)
45     root.after(1000, aktualizujZegar, root, zegar)
46
```


W celu uruchomienia funkcji pomiaru czasu należy ją wywołać jako argument tekstowy w panelu górnym:

```
15 # 4. górny panel
16 def inicjalizacjaGornegoPanela(root):
17     licznik_min = tk.Label(root, bg = '#000000', fg = '#FF0000', font = ('Digital-7', 20))
18     licznik_min.grid(row = 0, column = 0, columnspan = 7, ipadx = 10, pady = 30)
19     licznik_min['text'] = '0000'
20     buzka = tk.Button(root)
21     buzka.grid(row = 0, column = (M//2 - 1), columnspan = 3, pady = 30)
22     zegar = tk.Label(root, bg = '#000000', fg = '#FF0000', font = ('Digital-7', 20))
23     zegar.grid(row = 0, column = (M - 6), columnspan = 7, ipadx = 10, pady = 30)
24     zegar['text'] = '0001'
25     aktualizujZegar(root, zegar)
26     gorny_panel = [licznik_min, buzka, zegar]
27     return gorny_panel
```

W podobny sposób modyfikujemy ilość min:

- zmienna globalna:

```
6 CZAS = 0 # czas rozpoczęcia gry
7 LICZBA_MIN = 200 # ilość min do odgadnięcia
```

- funkcja modyfikująca ilość min do odgadnięcia:

```
49 # 7. Licznik min do odgadnięcia:
50 def aktualizujLicznikMin(licznik_min):
51     licznik_min["text"] = "0" * (4 - len(str(LICZBA_MIN))) + str(LICZBA_MIN)
```

Teraz trzeba utworzyć tablicę, która będzie przechowywać informację o tym co kryje się pod nie klikniętymi przyciskami. Najpierw tworzymy pustą tablicę wypełnioną zerami i wywołujemy ją kontrolnie w inicjalizacji tablicy (# 5.) oraz w terminalu

```
53 # 8. tablica gry
54 def inicjalizacjaTablicyGry():
55     tablica_gry = [[0 for j in range(M)] for i in range(N)]
56     l_min = LICZBA_MIN
57     print(tablica_gry) # wydruk testowy
58     return tablica_gry
```

```
30 # 5. panel z polami do klikania
31 def inicjalizacjaPlanszy(root):
32     przyciski = [tk.Button(root) for i in range(N*M)]
33     tablica_gry = inicjalizacjaTablicyGry()
34     #rozmieszczamy przyciski:
```

Losujemy miny do rozmieszczenia na planszy:

- dodajemy moduł random

```
main.py x
1 import tkinter as tk
2 import random
3
```

```

55     # 8. tablica gry
56     def inicjalizacjaTablicyGry():
57         tablica_gry = [[0 for j in range(M)] for i in range(N)]
58         l_min = LICZBA_MIN
59         # 9. rozmieszczenie min
60         while l_min:
61             x = random.randint(0, M - 1)
62             y = random.randint(0, N - 1)
63             if tablica_gry[y][x] == 0:
64                 tablica_gry[y][x] = "x"
65                 l_min -= 1
66
67         print(tablica_gry) # wydruk testowy

```

W kolejnym etapie należy w tablicę umieścić liczby określające ilość sąsiednich pól zawierających miny. Utworzymy funkcję wyszukującą sąsiadów z minami (czyli z 'x'):

```

69
70     # 10. Wyszukiwanie sąsiadów:
71     def znajdzSasiadow(tablica, x, y):
72         sasiedzi = [] # pusta tablica przed rozpoczęciem wyszukiwania
73         for i in range(-1, 2):
74             for j in range(-1, 2):
75                 if not (i == 0 and j == 0):
76                     if y + i >= 0 and y + i < N:
77                         if x + j >= 0 and x + j < M:
78                             sasiedzi.append((x + j, y + i)) #dopisuje sąsiadów do tablicy
79         return sasiedzi
80

```

Funkcję wywołujemy w inicjalizacji tablicy gry (modyfikujemy funkcję):

```

55     # 8. tablica gry
56     def inicjalizacjaTablicyGry():
57         tablica_gry = [[0 for j in range(M)] for i in range(N)]
58         l_min = LICZBA_MIN
59         # 9. losowe rozmieszczenie min
60         while l_min:
61             x = random.randint(0, M - 1)
62             y = random.randint(0, N - 1)
63             if tablica_gry[y][x] == 0:
64                 tablica_gry[y][x] = "x"
65                 l_min -= 1
66         # 11. wywołanie funkcji zliczającej miny
67         for i in range(N):
68             for j in range(M):
69                 if tablica_gry[i][j] == 0:
70                     sasiedzi = znajdzSasiadow(tablica_gry, j, i)
71                     l_min = 0
72                     for x, y in sasiedzi:
73                         if tablica_gry[y][x] == "x":
74                             l_min += 1
75                     tablica_gry[i][j] = l_min
76         print(tablica_gry) # wydruk testowy
77         return tablica_gry

```

Łączymy przyciski z odpowiednimi zdarzeniami (po kliknięciu w przycisk będziemy sprawdzać, czy przyciski zawierają miny lub inne elementy). Użyjemy do tego funkcji bind(), którą umieścimy w funkcji inicjalizacjaPlanszy():

```
32 def inicjalizacjaPlanszy(root):
33     przyciski = [tk.Button(root) for i in range(N*M)]
34     tablica_gry = inicjalizacjaTablicyGry()
35     #rozmieszczamy przyciski:
36     for i in range(N):
37         for j in range(M):
38             if j == 0:
39                 przyciski[i*M+j].grid(row = (i+1), column = j, padx = (20,0))
40             else:
41                 przyciski[i*M+j].grid(row = (i+1), column = j)
42                 przyciski[i*M+j].bind('button-1') # lewy p.myszy
43                 przyciski[i*M+j].bind('button-3') # prawy p.myszy
44     return przyciski
```

Dodamy funkcję odpowiedzialną za lewy klawisz myszy i prawy klawisz myszy:

```
92 # 11. lewy klawisz myszy:
93 def lewyKlik():
94     pass
95 # 12. prawy klawisz myszy:
96 def prawyKlik():
97     pass
98
99 # 2. Główna pętla programu
100 if __name__ == "__main__":
101     root = inicjalizacjaOkienka()
```

Wstępnie utworzone funkcje inicjujemy w linii 42 i 43:

```
41 przyciski[i*M+j].grid(row = (i+1), column = j)
42 przyciski[i*M+j].bind('button-1', lambda: lewyKlik()) # lewy p.myszy
43 przyciski[i*M+j].bind('button-3', lambda: prawyKlik()) # prawy p.myszy
44 return przyciski
```

Funkcja wczytująca tablicę ikon do użycia (tworzymy słownik ikonki z tablicami ikon):

```
92 # 13. wczytanie ikon do użycia:
93 def wczytajIkonki():
94     ikonki = {}
95     ikonki["cyfry"] = [tk.PhotoImage(file=str(i) + ".png") for i in range(1, 9)]
96     ikonki["buzki"] = [tk.PhotoImage(file="buzka" + str(i) + ".png") for i in range(1, 5)]
97     ikonki["flaga"] = tk.PhotoImage(file="flaga.png")
98     ikonki["miny"] = [tk.PhotoImage(file="mina.png"), tk.PhotoImage(file="pierwsza.png")]
99     return ikonki
100
101 # 11. lewy klawisz myszy:
```

Miny mogłyby mieć nazwy mina1.png oraz mina2.png – uprościła by się tablica.

Ikonki wczytujemy w głównej pętli programu:

```
108 # 2. Główna pętla programu
109 if __name__ == "__main__":
110     root = inicjalizacjaOkienka()
111     ikonki = wczytajIkonki()
```

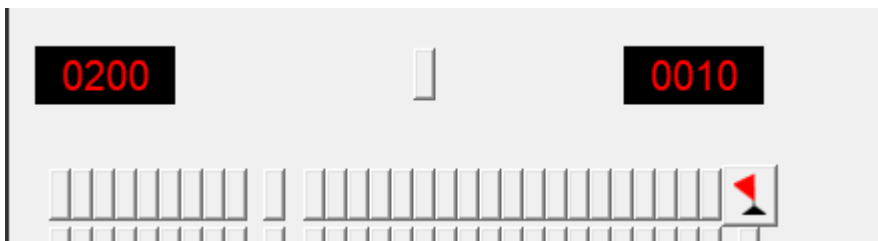
Przechodzimy do funkcji kliknięć myszą:

Prawy - Klik:

```
105 # 12. prawy klawisz myszy
106 def prawyKlik(przycisk, ikonki):
107     # pass # powoduje że funkcja nic nie robi
108     przycisk["image"] = ikonki["flaga"]
109
```

Dodajemy oba atrybuty do lambda:

```
42 przyciski[i*M+j].bind('<Button-1>', lambda event, : lewyKlik()) # lewy p.myszy
43 przyciski[i*M+j].bind('<Button-3>', lambda event, p = przyciski[i*M + j]: prawyKlik(p, ikonki)) # prawy p.myszy
44 return przyciski
```



Wraz z kliknięciem powinien zmodyfikować się licznik min:

Dodamy atrybut funkcji

```
31 # 5. panel z polami do klikania
32 def inicjalizacjaPlanszy(root, gorny_panel, ikonki):
33
34     przyciski = [tk.Button(root) for i in range(N*M)]
35     tablica_gry = inicjalizacjaTablicyGry()
36     #rozmieszczamy przyciski:
37     for i in range(N):
38         for j in range(M):
39             if j == 0:
40                 przyciski[i*M+j].grid(row = (i+1), column = j, padx = (20,0))
41             else:
42                 przyciski[i*M+j].grid(row = (i+1), column = j)
43                 przyciski[i*M+j].bind('<Button-1>', lambda event, : lewyKlik()) # lewy p.myszy
44                 przyciski[i*M+j].bind('<Button-3>', lambda event, p = przyciski[i*M + j]:
45                     prawyKlik(p, ikonki, gorny_panel, ikonki)) # prawy p.myszy
46     return przyciski
```

Oraz w pętli głównej:

```
111 # 2. Główna pętla programu
112 if __name__ == "__main__":
113     root = inicjalizacjaOkienka()
114     ikonki = wczytajIkonki()
115     gorny_panel = inicjalizacjaGornegoPanela(root, gorny_panel, ikonki)
116     przyciski = inicjalizacjaPlanszy(root)
```

i w funkcji

```
106 # 12. prawy klawisz myszy
107 def prawyKlik(przycisk, gorny_panel, ikonki):
108     # pass # powoduje że funkcja nic nie robi
109     global LICZBA_MIN
110     if przycisk.cget('image'): #zwraca wartość
111         przycisk['image'] = ''
112         LICZBA_MIN += 1
113     else:
114         przycisk['image'] = ikonki['flaga']
115         LICZBA_MIN -= 1
116     aktualizujLicznikMin(gorny_panel[0])
```

Lewy - Klik:

```
102 # 11. lewy klawisz myszy:
103 def lewyKlik(przyciski, przycisk, gorny_panel, tablica_gry, ikonki):
104     indeks = przyciski.index(przycisk)
105     pole = tablica_gry[indeks//M][indeks%M]
106     print(pole)
```

Oraz dodajemy atrybuty do event-u w funkcji inicjalizacjaPlanszy():

```
40 else:
41     przyciski[i*M+j].grid(row=(i+1), column=j)
42     przyciski[i*M+j].bind('<Button-1>', lambda event, p=przyciski[i*M+j]:
43         lewyKlik(przyciski, p, gorny_panel, tablica_gry, ikonki)) # lewy p.myszy
44     przyciski[i*M+j].bind('<Button-3>', lambda event, p=przyciski[i*M+j]:
45         prawyKlik(p, gorny_panel, ikonki)) # prawy p.myszy
46     return przyciski
```

W tej chwili prawy klawisz wstawia i usuwa flagi oraz modyfikuje licznik, a lewy klawisz w terminalu wyświetla wartość ukrytą pod przyciskiem.

Przechodzimy do tego co się będzie działo jeżeli odsłoniło się pole.

- Jak będzie to 'X' to gra ma się zakończyć.
- Jak będzie to liczba to ma się pole odsłonić.

Sprawdzamy wszystko warunkami:

```
103 # 14. aktualizuj stan przycisku
104 def aktualizujPrzycisk():
105     pass
106
107 # 15. zakończenie gry
108 def koniecGry():
109     pass
110
111 # 11. lewy klawisz myszy:
112 def lewyKlik(przyciski, przycisk, gorny_panel, tablica_gry, ikonki):
113     indeks = przyciski.index(przycisk)
114     pole = tablica_gry[indeks//M][indeks%M]
115     print(pole) # linia kontrolna może zostać wyłączona - zakomentowana
116     if pole == 'x':
117         koniecGry()
118     else:
119         aktualizujPrzycisk()
```

Atrybuty funkcji (linia 119) oraz zawartość funkcji aktualizujPrzycisk() :

```
116     if pole == 'x':
117         koniecGry()
118     else:
119         aktualizujPrzycisk(przyciski, przycisk, indeks, pole, tablica_gry, ikonki)
```

Oraz

```
103 # 14. aktualizuj stan przycisku
104 def aktualizujPrzycisk(przyciski, przycisk, indeks, pole, tablica_gry, ikonki):
105     przycisk['image'] = ikonki['cyfry'][pole - 1] # zaznaczenie pól z liczbami
```

Modyfikujemy miejsce kliknięcia w puste pole (czyli liczba '8'):

Elementy odsłonięte należy zamienić na Label przy czym jeżeli będzie to zero to powinno się zamienić na puste miejsce. W celu zrobienia tego modyfikujemy obsługę przycisków:

```
103 # 14. aktualizuj stan przycisku
104 def aktualizujPrzycisk(przyciski, przycisk, indeks, pole, tablica_gry, ikonki):
105     if pole != 0:
106         #przycisk['image'] = ikonki['cyfry'][pole - 1] # zaznaczenie pól z liczbami
107         przyciski[indeks] = tk.Label(root, image=ikonki["cyfry"][pole - 1])
108         if indeks % M == 0:
109             przyciski[indeks].grid(row = indeks // M + 1, column = indeks % M, padx = (30,0))
110         else:
111             przyciski[indeks].grid(row = indeks // M + 1, column = indeks % M)
112     else:
113         pass
```

Teraz kliknięty przycisk z liczbą zostaje wciśnięty.

Wciśnięty przycisk powinien być nie aktywny i zostać przestłonięty Label'em. Dopisujemy linię:

```
104 def aktualizujPrzycisk(przyciski, przycisk, indeks, pole, tablica_gry, ikonki):
105     if pole != 0:
106         przyciski[indeks].configure(state="disabled", border=1, highlightbackground="black")
```

Oraz ustawiamy odstępnięcie sąsiadów:

```
104 def aktualizujPrzycisk(przyciski, przycisk, indeks, pole, tablica_gry, ikonki):
105     if pole != 0:
106         przyciski[indeks].configure(state="disabled", border=1, highlightbackground="black")
107         #przycisk['image'] = ikonki['cyfry'][pole - 1] # zaznaczenie pól z liczbami
108         przyciski[indeks] = tk.Label(root, image=ikonki["cyfry"][pole - 1])
109         if indeks % M == 0:
110             przyciski[indeks].grid(row=indeks // M + 1, column=indeks % M, padx=(30,0))
111         else:
112             przyciski[indeks].grid(row=indeks // M + 1, column=indeks % M)
113     else:
114         # co gdy trafimy 'zero':
115         sasiedzi = znajdzSasiadow(tablica_gry, indeks % M, indeks // M)
116         for x, y in sasiedzi: #zaznaczaemy i wyłączamy sąsiadów
117             if (isinstance(przyciski[y * M + x], tk.Button) and przyciski[y * M + x]["state"] != "disabled"):
118                 aktualizujPrzycisk(przyciski, przycisk, y * M + x, tablica_gry[y][x], tablica_gry, ikonki)
```

W celu ustalenia kształtu przycisków warto dopisać ich rozmiary początkowe w:

```
32 def inicjalizacjaPlanszy(root, gorny_panel, ikonki):
33     przyciski = [tk.Button(root) for i in range(N*M)]
34     tablica_gry = inicjalizacjaTablicyGry()
35     #rozmieszczamy przyciski:
36     for i in range(N):
37         for j in range(M):
38             if j == 0:
39                 przyciski[i*M+j].grid(row=(i+1), column=j, padx=(20,0), ipadx=8)
40             else:
41                 przyciski[i*M+j].grid(row=(i+1), column=j, ipadx=8)
42                 przyciski[i*M+j].bind('<Button-1>', lambda event, p=przyciski[i*M + j]:
```

Wyłączamy klikalność przycisków po kliknięciu:

```
104 def aktualizujPrzycisk(przyciski, przycisk, indeks, pole, tablica_gry, ikonki):
105     if pole != 0:
106         przyciski[indeks].configure(state="disabled", border=1, highlightbackground="black")
107         #wyłączenie aktywności klikania w przyciski
108         przyciski[indeks].unbind("<Button-1>")
109         przyciski[indeks].unbind("<Button-3>")
110         #przycisk['image'] = ikonki['cyfry'][pole - 1] # zaznaczenie pól z liczbami
```

Funkcja po modyfikacjach i uzupełnieniu przyjmuje postać:

```
103 # 14. aktualizuj stan przycisku
104 def aktualizujPrzycisk(przyciski, przycisk, indeks, pole, tablica_gry, ikonki):
105     przyciski[indeks].configure(state="disabled", border=1, highlightbackground="black")
106     # wyłączenie aktywności klikania w przyciski
107     przyciski[indeks].unbind("<Button-1>")
108     przyciski[indeks].unbind("<Button-3>")
109     if pole != "x":
110         if pole != 0:
111             #przyciski['image'] = ikonki['cyfry'][pole - 1] # zaznaczenie pól z liczbami
112             przyciski[indeks] = tk.Label(root, image=ikonki["cyfry"][pole - 1])
113             if indeks % M == 0:
114                 przyciski[indeks].grid(row=indeks // M + 1, column=indeks % M, padx=(30,0))
115             else:
116                 przyciski[indeks].grid(row=indeks // M + 1, column=indeks % M)
117         else:
118             # co gdy trafimy 'zero':
119             sasiedzi = znajdzSasiadow(tablica_gry, indeks % M, indeks // M)
120             for x, y in sasiedzi: #zaznaczamy i wyłączamy sąsiadów
121                 if (isinstance(przyciski[y * M + x], tk.Button) and przyciski[y * M + x]["state"] != "disabled"):
122                     aktualizujPrzycisk(przyciski, przycisk, y * M + x, tablica_gry[y][x], tablica_gry, ikonki)
```

Obsługa funkcji koniecGry:

```
124 # 15. zakończenie gry
125 def koniecGry(przyciski, tablica_gry, ikonki):
126     for i in range(N):
127         for j in range(M):
128             if isinstance(przyciski[i*M + j], tk.Button) and przyciski[y*M + x]["state"] != 'disabled':
129                 przyciski[i*M + j].configure(state="disabled", border=1, highlightbackground="black")
130                 przyciski[i*M + j].unbind("<Button-1>")
131                 przyciski[i*M + j].unbind("<Button-3>")
132                 if tablica_gry[i][j] == 'x':
133                     przyciski[i*M + j] = tk.Label(root, image=ikonki['miny'][0])
134
135 # 11. lewy klawisz myszy:
136 def lewyKlik(przyciski, przycisk, gorny_panel, tablica_gry, ikonki):
137     indeks = przyciski.index(przycisk)
138     pole = tablica_gry[indeks//M][indeks%M]
139     print(pole) # linia kontrolna może zostać wyłączona - zakomentowana
140     if pole == 'x':
141         koniecGry(przyciski, tablica_gry, ikonki)
142     else:
```

Dodajemy funkcję:

```
135 # 16. dodawanie przycisków
136 def wstawPrzyciskNaKrate(przyciski, x, y):
137     if x == 0:
138         przyciski[y * M + x].grid(row=y + 1, column=x, padx=(30, 0))
139     else:
140         przyciski[y * M + x].grid(row=y + 1, column=x)
```

Oraz wywołujemy ją:

```
133 przyciski[i*M + j] = tk.Label(root, image=ikonki['miny'][0])
134 wstawPrzyciskNaKrate(przyciski, j, i)
135 # 16. dodawanie przycisków
```


Pojawiły się błędy, które po usunięciu:

```
124 # 15. zakończenie gry
125 def koniecGry(przyciski, tablica_gry, ikonki):
126     for i in range(N):
127         for j in range(M):
128             if isinstance(przyciski[i*M + j], tk.Button) and przyciski[i*M + j]['state'] != 'disabled':
129                 przyciski[i*M + j].configure(state="disabled", border=1, highlightbackground="black")
130                 przyciski[i*M + j].unbind("<Button-1>")
131                 przyciski[i*M + j].unbind("<Button-3>")
132                 if tablica_gry[i][j] == 'x':
133                     przyciski[i*M + j] = tk.Label(root, image=ikonki['miny'][0])
134                     wstawPrzyciskNaKrate(przyciski, j, i)
135 # 16. dodawanie przycisków
```

Blokujemy wyświetlanie nie klikniętych pól. Modyfikujemy linię:

```
124 # 15. zakończenie gry
125 def koniecGry(przyciski, tablica_gry, ikonki):
126     for i in range(N):
127         for j in range(M):
128             if isinstance(przyciski[i*M + j], tk.Button) and przyciski[i*M + j]['state'] != 'disabled':
129                 #przyciski[i*M + j].configure(state="disabled", border=1, highlightbackground="black")
130                 przyciski[i*M + j]['state'] = "disabled" # zmodyfikowana poprzednia linia kodu
131                 przyciski[i*M + j].unbind("<Button-1>")
```

Obsługa buźki restartującej grę

```
17 # 4. górny panel
18 def inicjalizacjaGornegoPanela(root, ikonki):
19     licznik_min = tk.Label(root, bg = '#000000', fg = '#FF0000', font = ('Digital-7', 15))
20     licznik_min.grid(row = 0, column = 0, columnspan = 7, ipadx = 10, pady = 30)
21     licznik_min['text'] = '0200'
22     buzka = tk.Button(root)
23     buzka.grid(row = 0, column = (M//2 - 1), columnspan = 3, pady = 30)
24     buzka['image'] = ikonki['buzki'][0]
25     zegar = tk.Label(root, bg = '#000000', fg = '#FF0000', font = ('Digital-7', 15))
```

```
167 # 2. Główna pętla programu
168 if __name__ == "__main__":
169     root = inicjalizacjaOkienka()
170     ikonki = wczytajIkonki()
171     gorny_panel = inicjalizacjaGornegoPanela(root, ikonki)
172     przyciski = inicjalizacjaPlanszy(root, gorny_panel, ikonki)
173     root.mainloop()
174
```

Łączymy ikonki z buźkami z resztą przycisków (uzależniamy je od siebie).

```
167 # 2. Główna pętla programu
168 if __name__ == "__main__":
169     root = inicjalizacjaOkienka()
170     ikonki = wczytajIkonki()
171     gorny_panel = inicjalizacjaGornegoPanela(root, ikonki)
172     przyciski = inicjalizacjaPlanszy(root, gorny_panel, ikonki)
173     gorny_panel[1].bind("<Button-1>", lambda event: resetujGre(root, gorny_panel, ikonki))
174     root.mainloop()
```

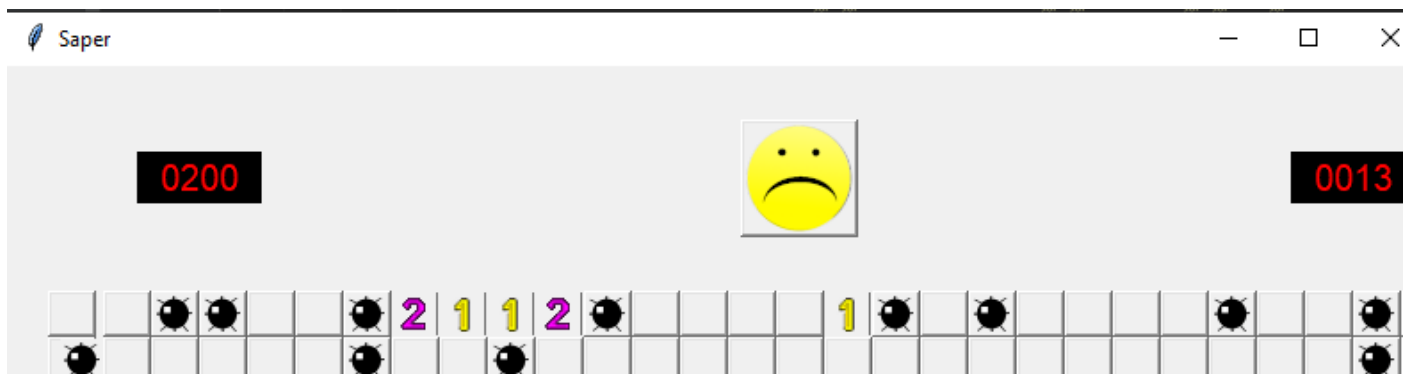
Oraz tworzymy funkcję resetującą grę:

- reset min oraz zegara

```
167 # 17 reset gry
168 def resetujGre(root, gorny_panel, ikonki):
169     gorny_panel[1]["image"] = ikonki["buzki"][0] #ustawiamy uśmiechniętą buźkę
170     przyciski = inicjalizacjaPlanszy(root, gorny_panel, ikonki)
171     global CZAS
172     CZAS = 0 # reset czasu gry
173     global LICZBA_MIN
174     LICZBA_MIN = 100 #reset liczby min
175     aktualizujLicznikMin(gorny_panel[0])
```

- reset i zmiana buziek:

```
145 # 11. lewy klawisz myszy:
146 def lewyKlik(przyciski, przycisk, gorny_panel, tablica_gry, ikonki):
147     indeks = przyciski.index(przycisk)
148     pole = tablica_gry[indeks//M][indeks%M]
149     print(pole) # linia kontrolna może zostać wyłączona - zakomentowana
150     if pole == 'x':
151         gorny_panel[1]['image'] = ikonki['buzki'][2]
152     koniecGry(przyciski, tablica_gry, ikonki)
```



Dodajemy funkcjonalność buźki podczas klikania:

```
45 przyciski[i*M+j].bind('<Button-3>', lambda event, p = przyciski[
46     prawyKlik(p, gorny_panel, ikonki)) # prawy p.myszy
47     przyciski[i * M + j].bind('<ButtonRelease>', lambda event:
48     zwykłaBuzka(gorny_panel[0])) # zwykła
49     return przyciski
```

Oraz definicja funkcji zwykłaBuzka() :

```
180 # 18 zwykła buźka
181 def zwykłaBuzka(buzka):
182     gorny_panel[1]['image'] = ikonki['buzki'][0]
```

```

147 # 11. lewy klawisz myszy:
148 def lewyKlik(przyciski, przycisk, gorny_panel, tablica_gry, ikonki):
149     gorny_panel[1]['image'] = ikonki['buzki'][1]
150     indeks = przyciski.index(przycisk)

```

```

159 # 12. prawy klawisz myszy
160 def prawyKlik(przycisk, gorny_panel, ikonki):
161     gorny_panel[1]['image'] = ikonki['buzki'][1]
162     # pass # powoduje że funkcja nic nie robi

```

Oprogramowujemy wygraną gry. Dodajemy/modyfikujemy zmienne globalne:

```

4 # 3. zmienne globalne
5 N = 15 # wiersze planszy
6 M = 30 # kolumny planszy
7 CZAS = 0 # czas rozpoczęcia gry
8 LICZBA_MIN = 100 # ilość min do odgadnięcia
9 POZOSTALA_LICZBA_FLAG = LICZBA_MIN
10 LICZBA_TRAFIONYCH_MIN = 0
11 CZY_KONIEC_GRY = False
12

```

Modyfikujemy odwołania do tych zmiennych:

```

175 # 17 reset gry
176 def resetujGre(root, gorny_panel, ikonki):
177     gorny_panel[1]["image"] = ikonki["buzki"][0] #ustawiamy uśmiechniętą buźkę
178     przyciski = inicjalizacjaPlanszy(root, gorny_panel, ikonki)
179     global CZAS
180     CZAS = 0 # reset czasu gry
181     global LICZBA_MIN #reset liczby min
182     global POZOSTALA_LICZBA_FLAG
183     POZOSTALA_LICZBA_FLAG = LICZBA_MIN
184     aktualizujLicznikMin(gorny_panel[0])

```

Uzupełniamy atrybuty funkcji:

```

49 prawyKlik(przyciski, p, gorny_panel, tablica_gry, ikonki) # prawy p.myszy

```

Tworzymy nową funkcję

```

201 # 19. wygrana
202 def wygranaGra(przyciski, tablica_gry):
203     for i in range(N):
204         for j in range(M):
205             if (isinstance(przyciski[i * M + j], tk.Button) and przyciski[i * M + j]["state"] != "disabled"):
206                 przyciski[i * M + j]["state"] = "disabled"
207                 przyciski[i * M + j].unbind("<Button-1>")
208                 przyciski[i * M + j].unbind("<Button-3>")
209

```

I zmieniamy obsługę prawego kliknięcia:

```
162 # 12. prawy klawisz myszy
163 def prawyKlik(przyciski, przycisk, gorny_panel, tablica_gry, ikonki):
164     gorny_panel[1]["image"] = ikonki["buzki"][1]
165     indeks = przyciski.index(przycisk)
166     pole = tablica_gry[indeks // M][indeks % M]
167     global POZOSTALA_LICZBA_FLAG
168     global LICZBA_TRAFIONYCH_MIN
169     if przycisk.cget("image"):
170         przycisk["image"] = ""
171         POZOSTALA_LICZBA_FLAG += 1
172         if pole == "x":
173             LICZBA_TRAFIONYCH_MIN -= 1
174     else:
175         przycisk["image"] = ikonki["flaga"]
176         POZOSTALA_LICZBA_FLAG -= 1
177         if pole == "x":
178             LICZBA_TRAFIONYCH_MIN += 1
179             if LICZBA_TRAFIONYCH_MIN == LICZBA_MIN:
180                 gorny_panel[1]["image"] = ikonki["buzki"][3]
181                 wygranaGra(przyciski, tablica_gry)
182     aktualizujLicznikMin(gorny_panel[0])
```

Aktualizujemy liczbę min :

```
20 # 4. górny panel
21 def inicjalizacjaGornegoPanela(root, ikonki):
22     licznik_min = tk.Label(root, bg = '#000000', fg = '#FF0000', font = ('Digital-7', 15))
23     licznik_min.grid(row = 0, column = 0, columnspan = 7, ipadx = 10, pady = 30)
24     aktualizujLicznikMin(licznik_min)
25     #licznik_min['text'] = '0200'
26     buzka = tk.Button(root)
```

Zatrzymanie licznika gry:

- zmiana buźki tylko na koniec gry:

```
196 # 18 zwykła buźka
197 def zwyklaBuzka(buzka):
198     if not CZY_KONIEC_GRY:
199         gorny_panel[1]['image'] = ikonki['buzki'][0]
```

- zakończenie gry:

```
201 # 19. wygrana
202 def wygranaGra(przyciski, tablica_gry):
203     global CZY_KONIEC_GRY
204     CZY_KONIEC_GRY = True
205     for i in range(N):
```

```
131 # 15. zakończenie gry
132 def koniecGry(przyciski, tablica_gry, ikonki):
133     global CZY_KONIEC_GRY
134     CZY_KONIEC_GRY = True
135     for i in range(N):
```

```
187 # 17 reset gry
188 def resetujGre(root, gorny_panel, ikonki):
189     gorny_panel[1]["image"] = ikonki["buzki"][0] #ustawiamy uśmiechniętą buźkę
190     przyciski = inicjalizacjaPlanszy(root, gorny_panel, ikonki)
191     global CZY_KONIEC_GRY
192     CZY_KONIEC_GRY = False
193     global CZAS
194     CZAS = 0 # reset czasu gry
```