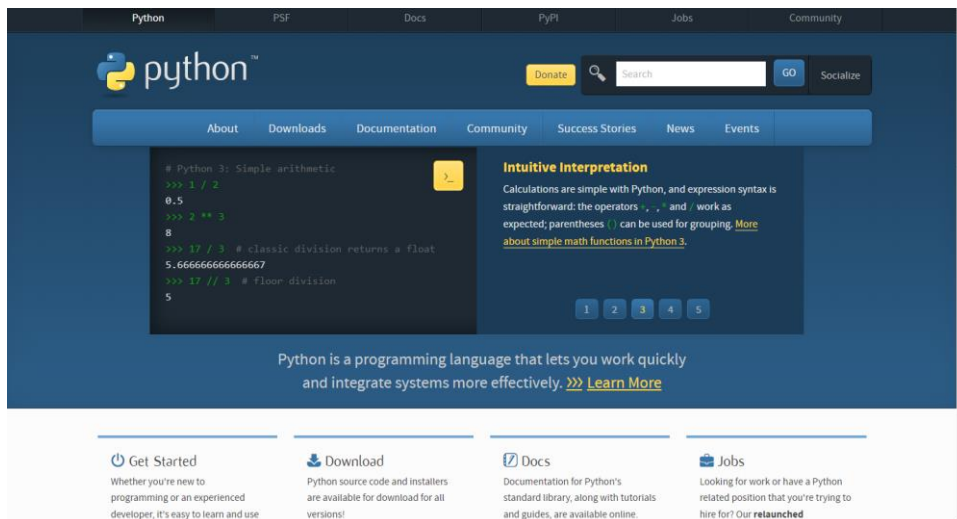


## Sprawozdanie

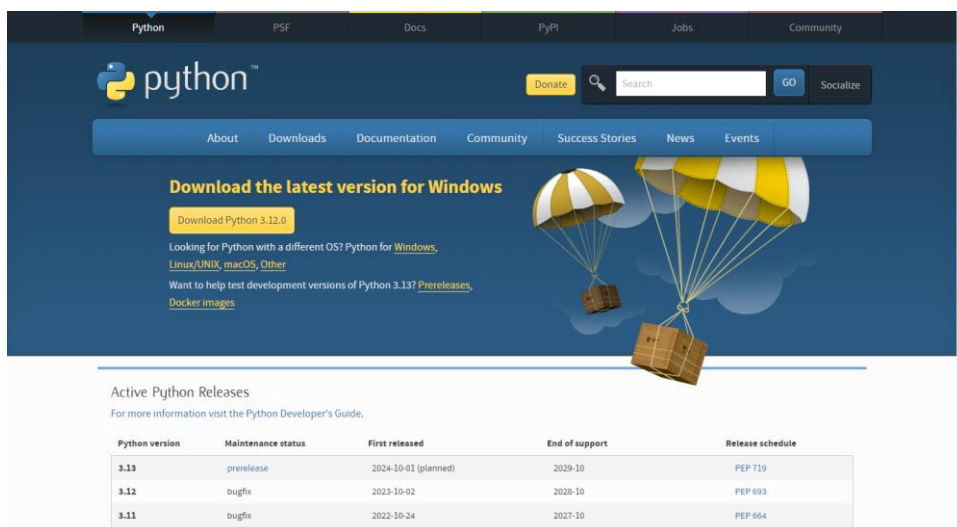
### Zad1

Celem zadania pierwszego było zainstalowanie interpretera Pythona.

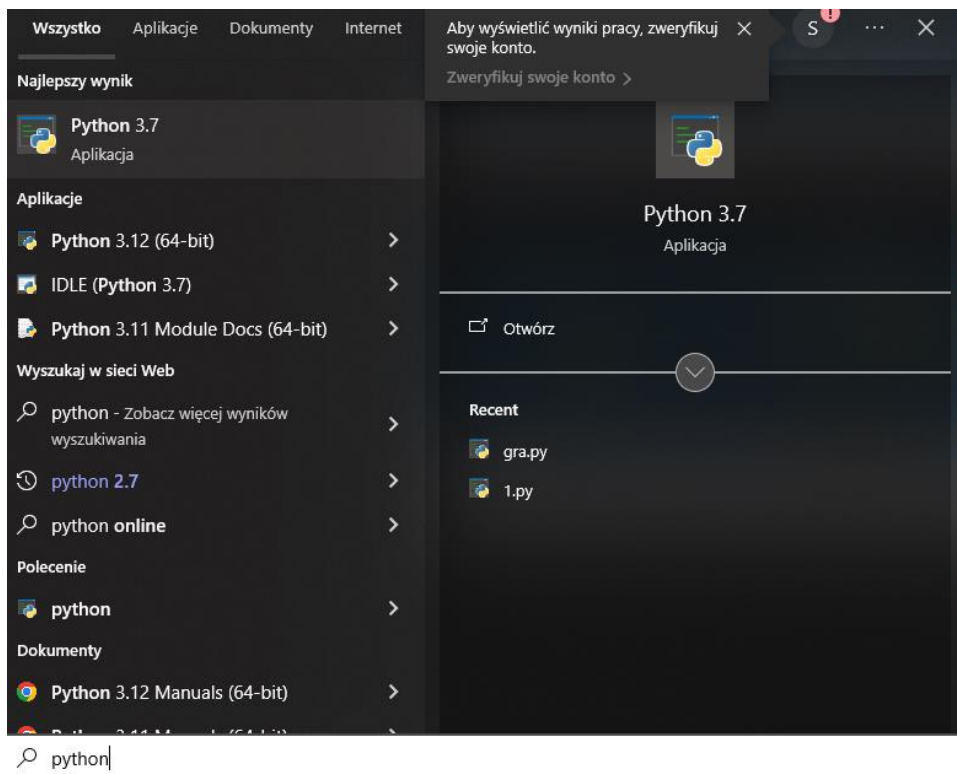
Zadanie to zaczynamy od pobrania wymaganych plików z oficjalnej strony python.org



Przechodzimy do zakładki Downloads



I naciskamy żółty przycisk. W tym przypadku jednak środowisko python jest już zainstalowane na komputerze, możemy to zobaczyć, wyszukując w eksploratorze windows,



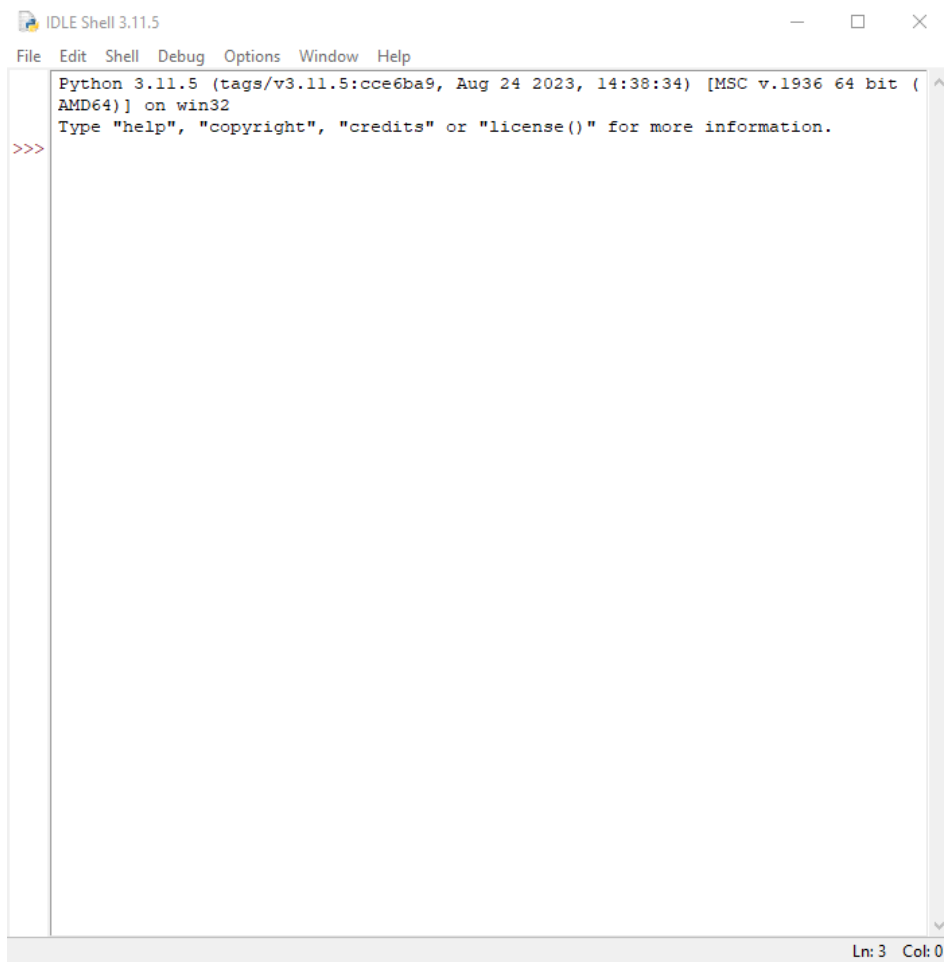
lub pisząc komendę python w konsoli.

```
C:\Users\student>python
Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023, 14:38:34) [MSC v.1936 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

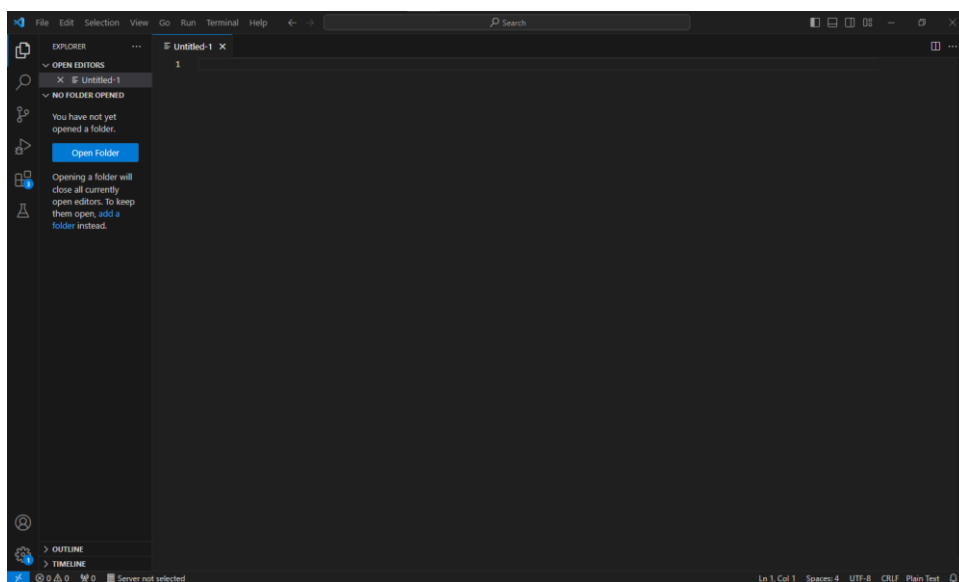
Widzimy też, że zainstalowana wersja pythona, znajdująca się w PATH ma wersję 3.11.5

## Zad 2

Aby stworzyć program w python możemy użyć różnych IDE (Integrated Development Environment). Albo domyślnego środowiska python IDLE który wygląda następująco:



Albo możemy użyć innego środowiska, takiego jak visual studio code, który jest już zainstalowany na komputerze i posiada wsparcie dla języka Python.



Aby utworzyć program wystarczy umieścić następujący kod:

```
HelloWorld.py X
D: > Studenci > Michal Greczkowski > Python projekty
1  print("Hello world")
2
3  zmiennarzeczywista = 1.55
4
5  print(zmiennarzeczywista)
```

Plik po kompilacji wyświetla się następująco:

```
Hello world
1.55
```

Warto zauważyć, że w porównaniu do takich języków jak c++ w pythonie, zmienne nie są wprost definiowane przez piszącego kod, lecz jest to robione dynamicznie. Jeżeli w zmiennej umieścimy typ int to zmienna będzie typu int, jeżeli umieścimy inny typ to ta zmienna zmieni swój typ.

Możemy też potwierdzić, że typem tej zmiennej jest float (liczby rzeczywiste) używając funkcji type zmieniając trochę kod:

```
HelloWorld.py X
D: > Studenci > Michal Greczkowski > Python projekty >
1  print("Hello world")
2
3  zmiennarzeczywista = 1.55
4
5  print(type(zmiennarzeczywista))
6
```

Otrzymujemy:

```
Hello world
<class 'float'>
```

Gdzie druga wydrukowana wartość pokazuje, że typem naszej zmiennej jest float.

### Zad 3

Aby wykonać zadanie trzecie trzeba użyć biblioteki math, którą możemy przywołać za pomocą polecenia:

```
1 import math
```

Użyjemy dwóch ważnych funkcji zawartych w tej bibliotece math.sin oraz math.pi

math.pi użyjemy do przeliczenia kąta na radiany,

```
5 alfa = 30
6 alfarad = (alfa * 2 * math.pi)/360
```

Jest to wymagane, ponieważ funkcja math.sin, potrzebna do wzoru na pole równoległoboku przyjmuje kąty w radianach a nie w stopniach.

Aby obliczyć pole równoległoboku używamy wzoru  $a*b*\sin(\alpha)$  w następujący sposób

```
7 pole = math.sin(alfarad) * a * b
```

Aby zobaczyć wynik po kompilacji używamy print

```
8 print(pole)
```

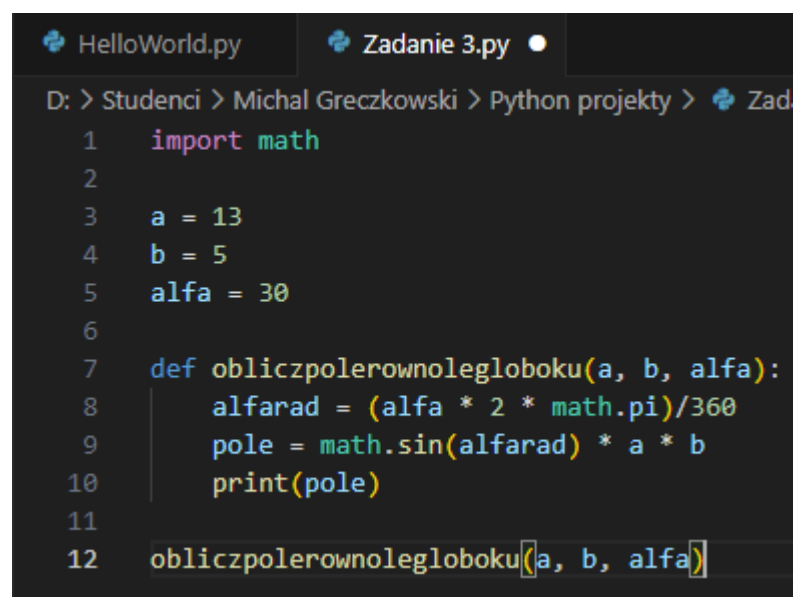
Teraz umieszczamy wszystkie wcześniejsze komendy do funkcji, otrzymujemy:

```
7 def obliczpolerownolegloboku(a, b, alfa):
8     alfarad = (alfa * 2 * math.pi)/360
9     pole = math.sin(alfarad) * a * b
10    print(pole)
```

Aby przywołać funkcję należy teraz użyć polecenia o nazwie funkcji

```
12 obliczpolerownolegloboku(a, b, alfa)
```

Końcowy program wygląda następująco:



The screenshot shows a code editor with two tabs: 'HelloWorld.py' and 'Zadanie 3.py'. The 'Zadanie 3.py' tab is active. The code in the editor is as follows:

```
D: > Studenci > Michal Greczkowski > Python projekty > Zad
1 import math
2
3 a = 13
4 b = 5
5 alfa = 30
6
7 def obliczpolerownolegloboku(a, b, alfa):
8     alfarad = (alfa * 2 * math.pi)/360
9     pole = math.sin(alfarad) * a * b
10    print(pole)
11
12 obliczpolerownolegloboku(a, b, alfa)
```

Program zwraca następujący wynik:

```
32.49999999999999
```

Zad 4

Do wykonania zadania 4 użyto znowu biblioteki math

```
import math
```

Aby obliczyć lambdę utworzono funkcję, która przyjmuje czas połowicznego rozpadu i oblicza za pomocą niej funkcję. Funkcja ta również drukuje ten wynik i zwraca go, aby można go było użyć w następnej funkcji. Funkcja ta wygląda następująco:

```
5 def obliczlamb(tpol):
6     lamb = math.log(2)/tpol
7     print("lambda:", lamb)
8     return lamb
```

Kolejna funkcja oblicza czas oraz sprawdza czy jest możliwe, żeby dany przedmiot był z roku rządów Cesarza Nerona. Funkcja używa do tego polecenia if, aby sprawdzić, czy obliczony czas, zgadza się z czasem przewidzianym. Funkcja przyjmuje 3 wartości, wartość lambdy, wartość minimalna czasu w którym moneta mogła zostać zrobiona, oraz przedział czasu, tak aby obliczyć maksymalny przedział. Końcowa funkcja wygląda następująco:

```
12 def sprawdzprawde(lamb, czasmozliwy, przedzial):
13     czas = -(math.log(9/10)/lamb)
14     print("Obliczony czas:", czas, "lat")
15     if czas < czasmozliwy or czas > czasmozliwy + przedzial:
16         print("Nie jest to możliwe")
```

Teraz aby obliczyć czas wystarczy przywołać tą funkcję w następujący sposób.

```
sprawdzprawde(obliczlamb(czaspol), czasmozliwy, przedzial)
```

Warto zauważyć, że przez użycie wcześniej komendy return, nie musimy przypisywać lambdy do zmiennej, możemy po prostu przywołać tą funkcję, w środku funkcji obliczającej czas.

Po kompilacji, w konsoli otrzymujemy:

```
lambda: 0.00012096809433855938
Obliczony czas: 870.9777254401363 lat
Nie jest to możliwe
```

Końcowy program wygląda następująco:

```
1  import math
2
3  czaspol = 5730
4  czasmozliwy = 2023 - 54
5  przedzial = 68 - 54
6
7  def obliczlamb(tpol):
8      lamb = math.log(2)/tpol
9      print("lambda:", lamb)
10     return lamb
11
12  def sprawdzprawde(lamb, czasmozliwy, przedzial):
13      czas = -(math.log(9/10)/lamb)
14      print("Obliczony czas:", czas, "lat")
15      if czas < czasmozliwy or czas > czasmozliwy + przedzial:
16          print("Nie jest to możliwe")
17
18
19  sprawdzprawde(obliczlamb(czaspol), czasmozliwy, przedzial)
```

## Zad 5

W zadaniu 5 nie potrzebujemy żadnej biblioteki pomocniczej. Całkę możemy najprościej obliczyć za pomocą metody prostokątów. Na początku zaczniemy od definicji funkcji  $f(x)$ :

```
5 def f(x):  
6     return 2*(x**3)+4*(x**2)-12*(x)
```

Potem przystępujemy do tworzenia funkcji obliczającej

Najpierw warto się upewnić, aby nasze wartości były typem float, tak aby przy wykonywaniu działań nie zostały one zmienione automatycznie przez interpreter na typ int

```
10 a = float(a)  
11 b = float(b)  
12 n = float(n)
```

Potem definiujemy parę zmiennych pomocniczych:

```
i = (b-a)/n  
  
x0 = a  
xn = a+i
```

Na sam koniec przystępujemy do obliczania, według wzoru na całkę metodą prostokąta

```
while (a<=(xn)<=b) or (a>=(xn)>=b):  
    pole=f((x0+(xn))/2)*i  
    calka += pole  
  
    x0 += i  
    xn += i
```

I oczywiście na koniec zwracamy wartość całki

```
26     return calka
```

Aby zobaczyć otrzymany wynik używamy polecenia print:

```
28 print("calka=" , oblicz(granicadol, granicagora, 100))
```

W konsoli otrzymujemy:

```
calka= 329.1468749999984
```

Końcowy program wygląda następująco:



```
1  granicadol = 0
2  granicagora = 5
3
4  def f(x):
5      return 2*(x**3)+4*(x**2)-12*(x)
6
7  def oblicz(a, b, n):
8      calka = 0
9      a = float(a)
10     b = float(b)
11     n = float(n)
12
13     i = (b-a)/n
14
15     x0 = a
16     xn = a+i
17
18     while (a<=(xn)<=b) or (a>=(xn)>=b):
19         pole=f((x0+(xn))/2)*i
20         calka += pole
21
22         x0 += i
23         xn += i
24
25     return calka
26
27 print("calka=" , oblicz(granicadol, granicagora, 100))
```