

Jméno:

Místnost:

Souřadnice:

0007

list

5 3 5  
0 0 0  
0 0 0

učo

body

Oblast strojově snímatelých informací. Svě UČO vyplňte zleva dle přiloženého vzoru číslic. Jinak do této oblasti nezasahujte.

0123456789

Celá pass/fail část se hodnotí trojstavově – 2 (bez chyb), 1 (jedna drobná chyba), 0 (více chyb nebo jedna větší). Toto hodnocení bude na listu 1.

**Příklad 1**  
pass/fail

Co vypíší následující programy?

```
def magic(num):
    print(num, end="")
    if num > 3:
        return num + 3
    print(num - 3, end="")
```

magic(5)

Program vypíše:

5

```
x = 7
while x * x <= 100:
    x += 3
print(x)
```

Program vypíše:

13

```
def mystery(num):
    if num <= 0:
        return
    if num % 2 == 0:
        print(num, end="")
    mystery(num - 1)
    if num % 2 == 1:
        print(num, end="")
```

mystery(5)

Program vypíše:

42135

```
a = [1, 2]
b = a
a[0] = b[1]
b[1] = b[1] + 5
print(a, b)
```

Program vypíše:

[2, 7] [2, 7]

```
x = 10
if x < 10:
    print(1, end="")
if x < 20:
    print(2, end="")
if x < 30:
    print(3, end="")
else:
    print(4, end="")
```

Program vypíše:

23

Jméno:

Místnost:

Souřadnice:

0007

list

2

učo

body

Oblast strojově snímatelých informací. Své UČO vyplňte zleva dle přiloženého vzoru číslic. Jinak do této oblasti nezasahujte.

0123456789

Je v následující funkci některá podmínka zbytečná (tj. dá se nahradit slovem True se zachováním efektu funkce)? Pokud ano, určete která. (Nemusíte nic zdůvodňovat.)

**Příklad 2**  
pass/fail

```
def sign(num: int) -> int:
```

```
    if num > 0:
```

ANO

```
        return 1
```

```
    if num == 0:
```

```
        return 0
```

```
    if num < 0:
```

```
        return -1
```



zde stačí jen return -1, if je zbytečný

Dojde v následující funkci někde k typové chybě (TypeError)? (Napište jasné ANO nebo NE.) Pokud ano, určete, kde (na kterém řádku) a proč.

```
def middle_char(text: str) -> str:
```

```
    if len(text) % 2 == 0:
```

```
        middle = len(text) / 2
```

ANO

```
    else:
```

```
        middle = (len(text) + 1) / 2
```

```
    return text[middle]
```

middle je typu float,  
místo int

Předpokládejme, že funkce answer vrací číslo typu int. Jsou následující dvě funkce obecně ekvivalentní, tj. vracejí vždy obě stejný výsledek, ať už je funkce answer jakákoliv? (Napište jasné ANO nebo NE, nic víc.)

**Příklad 3**  
pass/fail

```
def fun1() -> int:
```

```
    if answer() > 70:
```

```
        result = answer()
```

NE

```
    else:
```

```
        result = 70
```

```
    return result
```

```
def fun2() -> int:
```

```
    result = answer()
```

```
    if result < 70:
```

```
        result = 70
```

```
    return result
```

Přepište následující funkci tak, aby měla (za předpokladu, že všechny vstupy jsou typu bool, jak naznačuje typová anotace) stejný efekt, a přitom obsahovala pouze jedno jednoduché větvení (tj. jeden if) a neobsahovala žádný výskyt slov True a False. Rovněž se ve výsledku nesmí objevit žádné číslice.

**Příklad 4**  
pass/fail

```
def f(a: bool, b: bool, c: bool) -> None:
```

```
    if a == True:
```

```
        if b == False:
```

```
            print("OK")
```

```
        elif c != False:
```

```
            print("OK")
```

```
    if a and (c or not b):
```

```
        print("OK")
```

Jméno:

Místnost:

Souřadnice:



list



učo

body

Oblast strojově snímatelných informací. Svě UČO vyplňte zleva dle přiloženého vzoru číslic. Jinak do této oblasti nezasahujte.

0123456789

Napište jednoduchou funkci pomocí cyklu `for` (bez rekurze, bez cyklu `while`, bez použití funkce `sum`), která zjistí, zda je součet čísel na sudých pozicích v zadaném seznamu větší než 17, a podle toho vrátí buď `True` nebo `False`. Jasně vyznačte odsazení kódu (kde začínají a končí jednotlivé bloky).

**Příklad 5**  
pass/fail

```
def f1(numbers):
    sum_on_even = 0
    for i in range(0, len(numbers), 2):
        sum_on_even += numbers[i]
    return sum_on_even > 17
```

Přepište následující funkci pomocí cyklu `for` (bez použití cyklu `while`, bez seznamů, bez rekurze) tak, aby měla stejný efekt.

**Příklad 6**  
pass/fail

```
def counter(stop: int) -> None:
    i = 2
    while i < stop:
        i += 1
        print(i)
    for i in range(3, stop+1):
        print(i)
```

Napište definici třídy `DiskDrive`, jejíž objekty budou mít atributy `capacity` a `files`. Objekty se budou inicializovat voláním `DiskDrive(kapacita_disku)`, atribut `capacity` bude po inicializaci nastaven na zadanou kapacitu (celé číslo) a atribut `files` bude nastaven na prázdný seznam.

**Příklad 7**  
pass/fail

```
class DiskDrive:
    def __init__(self, kapacita_disku):
        self.capacity = kapacita_disku
        self.files = []
```

Jméno:

Místnost:

Souřadnice:

0007

list

4

učo

body

Oblast strojově snímatelných informací. Své UČO vyplňte zleva dle přiloženého vzoru číslic. Jinak do této oblasti nezasahujte.

0123456789

Napište příkazy, které v Pythonu vykonají uvedené činnosti. Řešením každého bodu je **jeden příkaz**.

**Příklad 8**  
pass/fail

Vypsát na výstup poslední dvě číslice kladného celého čísla většího než 9 uloženého v proměnné `num` **bez použití řetězců**.

```
print(n % 100)
```

Vypsát na výstup `True` nebo `False` podle toho, jestli má seznam v proměnné `lst` přesně tři prvky.

```
print(len(lst) == 3)
```

Vložit do slovníku v proměnné `dct` položku s klíčem "A" a hodnotou 7.

```
dct["A"] = 7
```

Snížit hodnotu proměnné `num`, která obsahuje sudé celé číslo, na polovinu.

```
num //= 2
```

Zavolat metodu `do_something` bez parametrů na objektu v proměnné `my_object`.

```
my_object.do_something()
```

Která z následujících tvrzení jsou pravdivá? Zakroužkujte ANO, pokud je tvrzení pravdivé; NE, pokud je tvrzení nepravdivé.

**Příklad 9**  
pass/fail

Slovníky v Pythonu jsou modifikovatelné.

☒ ANO ☐ NE

Hodnotou výrazu `5 // 3` je 1.

☒ ANO ☐ NE

Abstraktní datová struktura **zásobník** funguje na principu LIFO (last in, first out), tedy při výběru prvku dostaneme ten naposledy vložený.

☒ ANO ☐ NE

Typ výrazu `3 in [1, 2, 3]` je `bool`.

☒ ANO ☐ NE

Celá čísla jsou v Pythonu reprezentována datovým typem `Integer`.

☐ ANO ☒ NE

Datová struktura slovník (`dict`) smí obsahovat stejný klíč vícekrát.

☐ ANO ☒ NE

Datová struktura slovník (`dict`) smí obsahovat stejnou hodnotu vícekrát.

☒ ANO ☐ NE



Jméno:

Místnost:

Souřadnice:

0007

list

5

učo

body

Oblast strojově snímatelných informací. Své UČO vyplňte zleva dle přiloženého vzoru číslic. Jinak do této oblasti nezasahujte.

0123456789

Co vypíše následující program? Svou odpověď zdůvodněte.

Příklad 10

6 bodů

```
def foo(a_list):
    a_list.append(10)
    a_list = a_list[:]
    a_list.append(7)
    print(a_list)
```

```
[1,2,10,7]
[1,2,10]
```

```
my_list = [1, 2]
foo(my_list)
print(my_list)
```

Na 2. řádku se změní odkaz v proměnné `a_list` a dále už neukazuje na vstupní seznam, ale na nový, zkopírovaný seznam. Dále se tedy `my_list` nijak nezmění.

Zapište následující intenzionální seznamy explicitním výčtem prvků.

Příklad 11

6 bodů

```
[2 ** x for x in range(1, 6) if x % 2 == 1]
```

```
[2,8,32]
```

```
[(y, x) for x in range(1, 10) for y in range(x) if x + y == 7]
```

```
[(3,4), (2,5), (1,6), (0,7)]
```

```
[x + y for x in "abc" for y in ["ab", "cd"]]
```

```
["aab", "acd", "bab", "bcd", "cab", "ccd"]
```

Jméno:

Místnost:

Souřadnice:

0007

list

6

učo

body

Oblast strojově snímatelných informací. Své UČO vyplňte zleva dle přiloženého vzoru číslic. Jinak do této oblasti nezasahujte.

0123456789

K následujícím funkcím doplňte typové anotace (tak, aby odpovídaly tomu, co se ve funkcích děje a aby prošla typová kontrola). Nepoužívejte anotaci Any.

**Příklad 12****6 bodů**

```
def is_good_password(passw: str) -> Tuple[bool, Optional[str]] :
```

```
    if passw.islower():
        return (False, "Needs at least one uppercase character")

    return (True, None)
```

```
def run(mem: List[int], inputs: List[int]) -> Optional[int] :
```

```
    ip = 0
    while mem[ip] != 99:
        if mem[ip] != 3:
            return None
        mem[mem[ip] + 1] = inputs.pop()
        ip += 2
    return mem[0]
```

```
def average(nums: List[int]) -> int :
```

```
    if len(nums) == 0:
        return -1

    return sum(nums) // len(nums)
```

Jméno:

Místnost:

Souřadnice:



list



učo

body

Oblast strojově snímatelných informací. Svě UČO vyplňte zleva dle přiloženého vzoru číslic. Jinak do této oblasti nezasahujte.

```
0 1 2 3 4 5 6 7 8 9
```

Uvažujme algoritmus binárního vyhledávání v seznamu celý čísel. Popište, jaké tento algoritmus očekává vstupy (jejich typ, podmínky, které na ně jsou kladeny apod.) a jaké jsou jeho výstupy. Následně co nejpřesněji (ideálně Pythonovským kódem) popište fungování algoritmu.

**Příklad 13**  
6 bodů

**Parametry:**

`n` - číslo, které hledáme

`numbers` - seřazený seznam intů

**Výstup:**

`True` - pokud se `n` v `numbers` nachází

`False` - jinak

```
def binary_search(n: int, numbers: List[int]) -> bool:
    lower_bound = 0
    upper_bound = len(numbers) - 1
    while lower_bound <= upper_bound:
        middle = (lower_bound + upper_bound) // 2
        if n == numbers[middle]:
            return True
        if n < numbers[middle]:
            upper_bound = middle - 1
        else:
            lower_bound = middle + 1
    return False
```

Uvažujme následující (náhodnostní) program. Při opakovaném spouštění tohoto programu můžeme dostat různé výstupy. Napište všechny možné výstupy, které je možno při spuštění tohoto programu získat.

**Příklad 14**  
6 bodů

```
import random
n = 3
s = 0
x = 0
while x < n:
    s += random.randint(1, 3)
    x += s
print(x)
```

3  
4  
5  
6  
7