Ａ１.値の表示

１.値の表示(出力)

★コードは半角で入力する。行頭に空白を入れず揃え，１行で入力を完了する。
★数値の表示は print(数値) とする。
例）print(5)
≫ 5
★文字列の表示は，
print('文字列')または print("文字列")
シングル「'」 または ダブル「"」
クォーテーションマークで囲む。
例）print('Hello World')
≫ Hello World
print("こんにちは")
≫ こんにちは
print("He's",'so "cool".')
≫ He's so "cool".
★複数データの表示は，記号カンマ「,」で区切る。※半角の空白が出力される。
例）print('時速',40,"km")
★記号「+」で文字列を連結できる
例）print("snow","man")
≫ snow man
print("snow"+"man")
≫ snowman

２.複数文 及び コメント

★記号「#」により，そこから行末までを「コメント」として扱い実行しない。
例）print("Python") #１行目
print("Ver",3) #２行目
≫ Python
Ver 3
★複数の文をセミコロン「;」で区切り，一行で記述できる。
例）#上の２行の命令文を１行で記述
print("Python"); print("Ver",3)
≫ Python
Ver 3

【B01】
プログラムの実行結果を求めなさい。

```
print(1991) ; print("Program")
print('小学',1,"年生")
print("foot"+"ball","team")
print('"Sun" &',"'Moon'")
```

※１行で複数の文を記述しても改行して表示される

[答]

〖類題〗実行結果を求めなさい。
❶ print(3.14) # 円周率
❷ print("令和"+"7年"); print(4,"月")
❸ print('"Wow, that' ↴
+"'s a rain"+'bow!" he said.')

【B02】
≫［答］45 cm
と出力するプログラムを作成しなさい。ただし数値の部分は，整数の「45」を記述すること。

[答]

〖類題〗
❶ ≫ 円周率： 3.14
と出力するプログラムを作成しなさい。ただし数値の部分は，小数の「3.14」を記述すること。
❷ 次のプログラムの誤りを正しなさい。
Print(2025."年')
≫ 2025 年 #期待している出力
❸ ダブルクォーテーションマークで「優勝」を強調するプログラムを作成したい。誤りを正しなさい。
print("やったぁ～'優勝"だ！')

Ａ２.四則演算

１.主な算術演算子と計算規則

★四則計算における主な演算子を示す。

+	加算	例）7+3 → 10
-	減算	7-3 → 4
*	乗算	7*3 → 21
/	除算	7/3 → 2.33･･･
//	除算(切り捨て)	7//3 → 2
%	除算(剰余)	7%3 → 1
**	累乗	7**3 → 343

※除算では，0で割るとErrorとなる。

★計算規則(括弧( )の扱いと優先順位)は，普通の計算と同じである。

例）print(5+2,5-2,5*2,5/2,5//2,5%2,5**2)
≫ 7 3 10 2.5 2 1 25
print(5+2*(1-4))
≫ -1
print(5/0)
≫ ZeroDivisionError

２.誤差

★浮動小数点数を扱うときに「丸め誤差」が生じることがある。

※10進数から2進数へ変換するとき無限に循環する小数となることがある。この小数を，有限の桁で丸める処理を行うことで生じる誤差である。

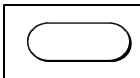
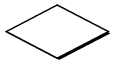
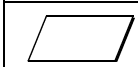
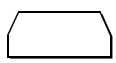
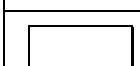
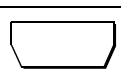
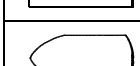
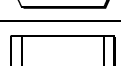
例）print(0.1*3)
≫ 0.30000000000000004 #0.3
※$0.1_{(10)} = 0.000110011001100\cdots_{(2)}$
print(0.2+0.4)
≫ 0.6000000000000001 #0.6
print(1/49*49)
≫ 0.9999999999999999 #1

３.アルゴリズムとフローチャート

★アルゴリズムとは...
計算や問題解決の手順を定めたもの

★フローチャートとは...
アルゴリズムを可視化した流れ図

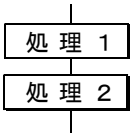
[フローチャートの主な記号]

	開始 終了		判断 分岐
	データ 入出力		反復の 開始
	処理		反復の 終了
	表示		定義済 処理

４.基本制御構造

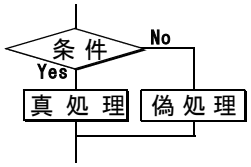
★アルゴリズムは順次構造，選択構造，反復構造の３つの基本制御構造で表すことができる。（構造化定理）

★順次構造



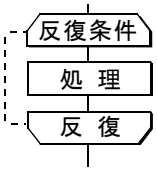
各処理が直線的につながり上から下へ順番に実行

★選択構造 ☞Ａ５



条件により流れが分かれ，それぞれの処理を実行する。

★反復構造 ☞Ａ６



条件が満たされている間，処理を繰し実行する。

【B03】
プログラムの実行結果を求めなさい。

```
print(5+2,5-2,5*2,5/2,5//2,5%2,5**2)
print(3/2/2,3//2//2,3%2%2)
print(2*3*2,2**3*2,2*3**2,2**3**2)
print((7-5)/2-3*(6-4))
print(13%(7-5)*3,13%(7-5)**3)
```

※通常の数式に書き換えてみると分かりやすくなる
３行目は，$2\times3\times2$，$2^3\times2$，2×3^2，2^{3^2}
５行目の右の式では，(7-5)**3 を先に計算する

[答]

※除算 / の結果は浮動小数点型

〖類題〗

❶ 実行結果を求めなさい。

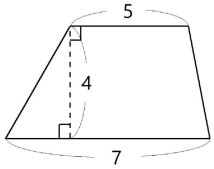
① print(9+3,9-3,9*3,9/3)
② print(9//3,9%3,9**3)
③ print(5//3*2,5%3**2)
④ print(-3-(7*9-8/2**3))

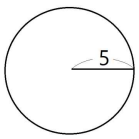
❷ 示した実行結果から，□に当てはまる適当な演算子を推定しなさい。

① print(7□3,7□3,7□3,7□3)
print(7□3,7□3,7□3)
≫ 2 21 4 343
2.33･･･35 10 1

② print(3□5□2, 7□(9□4))
≫ -7 49

❸ 下の□に適当な数式を入れなさい。





① 上底5,下底7,高さ4 の台形の面積
print("[面積] =", □)

② 半径5の円の面積と円周
ただし，円周率を3.14とする。
print("[面積] =", □)
print("[円周] =", □)

③ 時速45kmを秒速mで表す
print("時速 45 km は，")
print("秒速", □ ,"m です。")

④ 500円で1個15円の卵を買う。
買うことができる最大の個数と残金
print("[個数] =", □)
print("[残金] =", □)

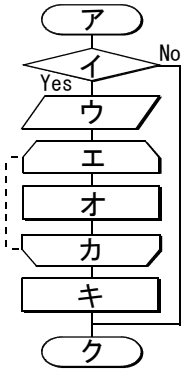
[研究] print(-7/4,-7//4,-7%4) の出力

▶ -7/4 は普通の除算である。
▶ [割られる数]=[割る数]×[商]＋[余り]
ただし 0≦[余り]＜[割る数]である。
従って -7//4,-7%4 について考えると，
4で割り，余りが0以上4未満だから，
-7 = 4×[ア]＋[イ]
▶ -7//4 = [ア]，-7%4 = [イ] となる。

【B04】

食事についての流れ図を作成した。
流れ図のア～クに当てはまる項目を，①～⑧から選びなさい。

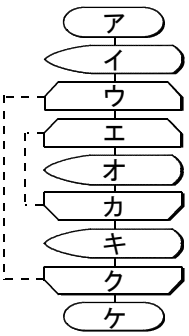
① 調理反復.完了まで
② 食事する
③ 腹が空いているか
④ おわり
⑤ 食材を仕入れる
⑥ 調理反復
⑦ はじめ
⑧ 食材を調理する



［答］

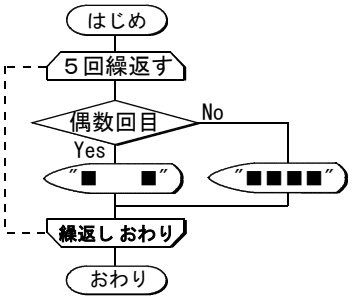
〖類題〗

❶ 右の図形を出力させたい。
流れ図のア～ケに当てはまるものを，①～⑩から選びなさい。



① はじめ
② おわり
③ ２回繰返す
④ ３回繰返す
⑤ "■　　■"と表示
⑥ "■■■■■"と表示
⑦ 繰返し おわり

❷ 下の流れ図を実行したとき，どのような図形が表示されるか。



Ａ３．変数

１．変数

★変数名の規則は以下のとおりである。
- 英文字：a～z，A～Z（大･小文字区別）
  数　字：0～9
  記　号：_（アンダースコア）のみ
- 数字から始めてはいけない。
- 文字数の制限はない。
- 以下の予約語は使えない。

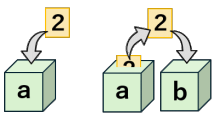
False	None	True	and	as
assert	async	await	break	class
continue	def	del	elif	else
except	finally	for	from	global
if	import	in	is	lambda
nonlocal	not	or	ass	raise
return	try	while	with	yield

★変数への代入は以下のように行う。
変数名 = 数値
変数名 = '文字列' または "文字列"

★print(変数名)で変数の値を出力する。

例）
```
pai = 3.14
print(pai)
```
≫ 3.14

例）
```
a = 2
b = a
print(a,b)
city = "市川"
print(city)
print("city")
```
≫ 2 2
市川　　#cityは変数
city　　#"city"は文字列



２．データ型

★Pythonで扱われるデータの種類のうち代表的な「データ型」は下の通りである。
- 数値:整数型　　int
- 数値:浮動小数点型　float
- DATA:文字列型　　str
- 論理:論理型　　bool(True,False)

★データ型を変換できる場合もある。
- 整数型 = int(文字列型)
- 文字列型 = str(整数型)
- 浮動小数点型 = float(整数型)　など

３．数値や文字列の入力

★キーボードからの入力は，
input(表示文字[必要ないとき空欄])
で行う。入力データは文字列型となる。

例）
```
x=input( )      #ｘは文字列型
y=int(x)        #ｙは整数型
```
例）
```
x = input("金額=")  #「50」と入力
y = int(x)
print(x,y)
print("金額：" , x , "円")
print("金額：" , y , "円")
print("金額：" + x + "円")
print("金額：" + y + "円")
```
≫ 金額=50
50 50
金額： 50 円
金額： 50 円
金額：50円　#ｘは文字列型
TypeError　#ｙは整数型

【B05】
変数名として不適切なものに×をつけよ。

data2(　)	data.2(　)	data_2 (　)
2data(　)	goukei(　)	gou-kei(　)
for　(　)	_for　(　)	For　　(　)
IchiTaro(　)	090_0123_4455(　)	
Senior_Secondry_School_2025 (　)		

［答］

[NOTE]リスト(配列)名の規則も同じ☞Ａ４

【B06】
プログラムの実行結果を求めなさい。

```
a = 5
b = 3
print(a+b, a-b, a*b, a/b, a//b, a%b)
a = a - b
b = a + b
print(a, b, (a+b)**(a-b))
```

※4行目で変数aの値は a=5-3=2 となり，
5行目で変数bの値は b=2+3=5 となる。

[答]

〖類題〗

❶ 下の実行結果となるようプログラムの□に適当な文字を入れなさい。

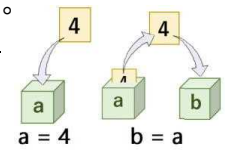
```
a = 4
b = 7
print(a,b)
x = a
a = □
b = □
print(a,b)
```

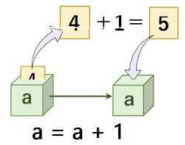
≫ 4 7
7 4　※変数の値が入れ替わった！

❷ 実行結果を求めなさい。

```
a = 4
b = a
print(a,b)
a = a + 1
print(a,b)
b = a + b
print(a,b)
```

a	b
4	/
4	□
□	□
□	□





[変数テーブル]

【B07】
プログラムの実行結果を求めなさい。

```
yes = 1
yes = yes + yes + yes
Yes = "Yes"
Yes = Yes + Yes + Yes
print(yes,Yes)
```

[答]
※変数 yes は整数型，変数 Yes は文字列型

【B08】
プログラムの実行結果を求めなさい。

```
・a = "2"          ・a = "2"
  print(a+3)         print(int(a)+3)
```

[答]

〖類題〗実行結果を求めなさい。

```
・a = "1.5"          ・a = "1.5"
  print(a+2.5)         print(float(a)+2.5)
・a = 3.1            ・a = 3.1
  print(a+"4159")      print(str(a)+"4159")
```

【B09】
時速(km/h)を入力すると秒速(m/s)に変換し表示するプログラムである。
□ に適当なコードを入れなさい。

```
x = input("時速何km : ")
jisoku = float(x)
byosoku = □
print("秒速",byosoku,"m")
```

[答]

ただし，一般的に累乗は掛け算に対して計算速度が遅く，割り算はさらに遅くなります。

〖類題〗100円硬貨を1枚持っている。100円以下の商品を買って100円硬貨で支払ったとき，お釣りの50円,10円,5円,1円の各硬貨は何枚ずつになるだろうか。□に当てはまる適当な数式を，ア～オの中から選び記号で答えなさい。

```
nedan=input("商品の値段 : ")
price=int(nedan)
oturi=100-price
print("50円硬貨 : ",□,"枚")
print("10円硬貨 : ",□,"枚")
print(" 5円硬貨 : ",□,"枚")
print(" 1円硬貨 : ",□,"枚")
```

ア. oturi % 5
イ. oturi // 50
ウ. oturi % 10 // 5
エ. oturi // 50 % 10
オ. oturi % 50 // 10

Ａ４．リスト（配列）

１．リストの構成

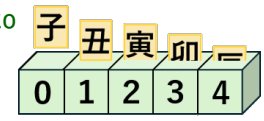
★「リスト」とは，仕切りのある箱に複数のデータを順番に入れていく仕組み。
※他言語の「配列」に近い概念である。

★リストの生成
・生成：リスト名=[値0，値1，値2，･･･]
※リスト名=[ ] ⇒ 空の箱
リスト名=[""]*(整数) ⇒ 個数指定
・[ ]内の各値を「要素」という。
要素は左から0,1,2,･･･と「インデックス(添字)」番号をつけ区別する。
文字列の要素は'」または"」で囲む。

★リスト及び要素の取り出し
・リスト名 ⇒ [値0，値1，値2，･･･]
・リスト名[インデックス]⇒(対応する要素)

★要素の個数
・len(リスト名)⇒要素の個数の取得

例)seireki=[2020,2021,2022,2023,2024]
```
eto = ["子","丑","寅" ,"卯","辰"]
print(seireki)
print(eto)
print(seireki[0],eto[2])
print(seireki[1]+seireki[3])
y = 4
print(str(seireki[y])+"年",eto[y])
```
≫ [2020, 2021, 2022, 2023, 2024]
['子', '丑', '寅', '卯', '辰']
2020 寅
4044
2024年 辰



２．２次元リスト

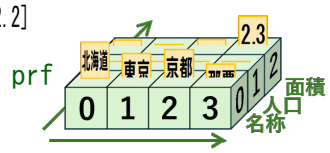
★リストの要素をリスト(入れ子)にして，２次元のリストを考えることができる。
※多次元のリストも同様に考えられる。

★リストの生成
```
リスト名=[
    [値00,値01,値02,値03,･･･],
    [値10,値11,値12,値13,･･･],
        ･･･････････
]
```

★リスト及び要素の取り出し
・リスト名⇒[[値00,値01,･･],[値10,･･],･･]
・※例 リスト名[1]⇒[値10,値11,値12,･･]
・※例 リスト名[0][2]⇒値02

★要素の個数
・※例 len(リスト名[0][2])⇒要素の個数

例)[都道府県,人口(万人),面積(千km²)]
```
prf = [
    ["北海道", 504, 78.4],
    ["東京",  1429,  2.2],
    ["京都",   252, 34.6],
    ["沖縄",   147,  2.3]
]
print(prf)
print(prf[1])
print(prf[0][0],prf[1][2],prf[3][1])
```
≫ [['北海道',504,78.4],['東京',･･],[･･],･･,[･･,2.3]]
['東京', 1429, 2.2]
北海道 2.2 147



【B10】
最初の実行では「0」，２回目は「2」と入入力した。結果はどうなるか。

```
name = ["佐藤","鈴木","高橋"]
club = ["野球","陸上","卓球"]
point = [64, 80, 57]
n = int(input('0～2 の番号：'))
print(name[n],club[n],point[n])
```

[答]

〖類題〗

❶ 下のプログラムの実行すると，どのように表示されるだろうか。

```
point = [64, 80, 57]
sum = point[0]+point[1]+point[2]
print( sum / len(point) )
```

❷ 実行結果が「49 30」となるように
[　]に適当な数値を入れなさい。

```
num=[2, 0, 7, 3, 1, 5]
a1=num[　]**num[　]
a2=num[　]*(num[　]-num[　])
print(a1, a2)
```

≫ 49 30

❸ プログラムと実行結果の□に当てはまる適当な数や文字を入れなさい。

```
w=["ど","う","ぶ","つ","あ","い","ご"]
x=[1,2,0]
wx=w[x[1]]+w[x[2]]+w[x[0]]
y=[□,□,□,□]
wy=w[y[2]]+w[y[0]]+w[y[1]]+w[y[3]]
print(wx,wy)
```

≫ □ どいつご

【B11】

実行後「4」,「6」と入力した。
表示される実行結果はどうなるか。

```
a = [2,0,7,3,1,5]
n = int(input())        #「4」と入力
m = int(input())        #「6」と入力
a[n] = m
print(a)
```

[答]

※インデックス番号「4」の左から5番目の要素「1」に，数値の「6」が上書きされる。
※２，３行目は，input()で入力されたデータの型を変換している ☞Ａ３

〖類題〗プログラム開始後「1」と入力した。出力を参考に□に命令文を埋めなさい。

```
m=["パン","サラダ","drink?"]
print("menu:",m)
d=["珈琲","紅茶","牛乳"]
print("drink:",0,d[0],1,d[1],2,d[2])
k=int(input("番号選択:"))  #「1」と入力
□
print("menu:",m)
```

≫ menu: ['パン', 'サラダ', 'drink?']
drink: 0 珈琲 1 紅茶 2 牛乳
番号選択:1
menu: ['パン', 'サラダ', '紅茶']

【B12】

出席番号と知りたい項目の番号を入力するとその「情報」を表示するプログラムである。出力を参考に□を埋めなさい。

```
data = [
    [1,"佐藤","野球",64],
    [2,"鈴木","陸上",80],
    [3,"高橋","卓球",57],
    [4,"田中","柔道",69],
    [5,"伊藤","剣道",72],
]
n=int(input("出席番号1～5："))
i=int(input("名前0,部活1,点数2："))
print(str(□)+"番",↘
            data[□][□])
```

≫ 出席番号1～5：4　　#「4」と入力
名前0,部活1,点数2：1　#「1」と入力
4番 柔道

[答]

※配列の表現法は言語によって異なる。
配列の配列として表現する方法
A[1][1] ⇒ C,Java,Python 等
添字を「,」で区切り表示する方法
A[1,1] ⇒ Python,Fortran,DNCL等

〖類題〗言葉遊びをしよう。実行後 3⏎4⏎10 と入力すると，「忍者がのり弁を食べながら月面で遊んでいた」と表示された。
[　]内にインデックス番号を入れなさい。

```
word = [
   ["猿","恐竜","ロボット","忍者"],
   ["のり弁","桃","餃子","あんパン"],
   ["東京駅","砂漠","月面","海底"]
]
subject=int(input("subject 0～3:"))
food=int(input("food 4～7:"))
place=int(input("place 8～11:"))
print(word[ ][    ]+"が"↘
     +word[ ][    ]+"を食べながら"↘
     +word[ ][    ]+"で遊んでいた")
```

≫ 忍者がのり弁を食べながら↘
月面で遊んでいた

Ａ５.選択(分岐)構造

１.条件による分岐処理

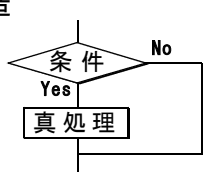
★条件が，成り立つ(真)か，成り立たない(偽)かで流れを変える処理。

★条件は，比較演算子を用いた式で表す。

a == b ... a と b が一致する
a != b ... a と b が一致しない
a ＞ b ... a が b より大きい
a ＜ b ... a が b より小さい
a >= b ... a が b 以上である
a <= b ... a が b 以下である

★基本的なプログラム構造

```
if [条件式]:※1
⇨   [真の場合の処理]
        ブロック※3
※2
```



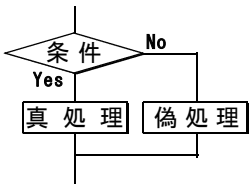
※１.コロン「：」　※２.インデント(字下げ)
※３.インデントでまとめられたコード行

例）
```
x = 7                    #「2」,「5」のときは･･･
if x >= 5:
    print("大きい！")
```
≫ 大きい！

２.その他のプログラム構造

★else を含む構造

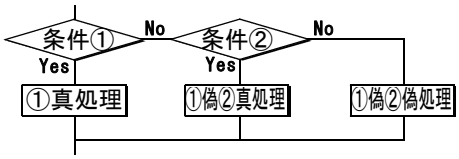
```
if [条件式]:
    [真の場合の処理]
        ブロック
else:
    [偽の場合の処理]
        ブロック
```



例）
```
x = 5                    #「7」,「9」のときは･･･
if x >=7:
    print("大")
else:
    print("小")
```
≫ 小

★elif を含む構造

```
if [条件式①]:
    [①真 の処理]          ブロック
elif [条件式②]
    [①偽,②真 の処理]  ブロック
else:
    [①偽,②偽 の処理]  ブロック
```



例）
```
x = -2                   #「7」,「0」のときは･･･
if x > 0 :
    print("正の数")
elif x == 0 :
    print("零")
else:
    print("負の数")
```
≫ 負の数

３.論理演算子

★複数の条件には，論理演算子を用いる。

・[条件1] and [条件2] ⇒ 論理積:かつ
・[条件1] or [条件2] ⇒ 論理和:または
・not [条件]　　⇒ 否定:ではない

例）
```
x = -2                   #「3」,「-0.5」のときは･･･
if x >=0 and x**2 >= 1:
    print("1以上")
elif x >=0 and x**2 <= 1:
    print("-1以上1未満")
else:
    print("-1未満")
```
≫ -1未満

[NOTE] 比較演算子は連結ができる。
例）x＞0 and x＜=10　⇒　0＜x＜=10

【B13】
次のプログラムの実行結果は「当たり」であった。□に入る可能性のある比較演算子をすべて書き出しなさい。

```
number = 65
if number □ 50:
    print("当たり")
```
≫ 当たり

[答]

【B14】
入力した整数が偶数か奇数かを判定するプログラムである。□に適当な文字や記号を入れよ。

```
n = int(input())      #「6」と入力
hantei = "□"
N = n % 2
if N □ 0 :
   hantei = "奇数"
print(n,"は",hantei)
≫ 6 は 偶数
```

※２行目で初期値を設定し,続くif文の条件により望ましい値に変更していく定石である。

[答]

〖類題〗下のプログラムは，60000票以上は"当選"，他は"落選"と判定し表示するものである。□に適当な文字,式を入れよ。

```
vote = int(input("得票数:"))  #40528
result = "当選"
if □ :
    result = "□"
print(result)
≫ 落選          ※82709と入力すると「当選」
```

【B15】
入力した整数が偶数か奇数かを判定するプログラムである。□に適当な文字や記号を入れよ。

```
n = int(input())      #「6」と入力
N = n % 2
if N □ 0 :
   print(n,"は","偶数")
else :
   print(n,"は","□")
≫ 6 は 偶数
```

※【B14】と等価なプログラムコードである。

[答]

〖類題〗２数を入力すると，その差を表示してくれるプログラムをコーディングした。□に適当な式を入れよ。

```
a = int(input("a:"))      #例「5」を入力
b = int(input("b:"))      #例「8」を入力
if a >= b :
    □
else :
    □
print("2数の差は：",diff)
≫ 2数の差は： 3           #他も考えよ
```

【B16】
入力した点数が,80以上は「優秀」,30以上80未満は「合格」30未満は「追試」と表示したい。□に適当な式を入れよ。

```
point = int(input())  #「67」と入力
if □ :
    eval = "優秀"
elif □ :
    eval = "追試"
else:
    eval = "合格"
print(eval)
≫ 合格
```

[答]

〖類題〗【B16】で，5･7行目を入れ替え
5行目 eval = "合格" , 7行目 eval = "追試"
とすると□にはどのような式が入るか。

【B17】
【B16】〖類題〗と等価なプログラムになるよう□に適当な式を入れよ。

```
point = int(input())  #「67」と入力
if □ :
    eval = "合格"
elif □ :
    eval = "優秀"
else:
    eval = "追試"
print(eval)
≫ 合格
```

[答]

Ａ６.反復(繰り返し)構造

１.規則性のある反復

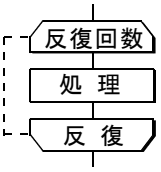
※反復する回数が決まっている場合

```
for [変数] in [反復する仕組み]:
    [反復する処理] ※ブロック
```

《反復する仕組み》

★ `for i in range( ):`

1) 順に整数を生成
2) 整数を変数に代入
3) 繰り返し処理を実行



〖range( )の仕様〗

① range(終了値):
　0から[終了値]-1までの整数を生成
　例) range(5) ⇒ 0, 1, 2, 3, 4
　　range(0) ⇒ null(何もなし)

② range(開始値,終了値):
　[開始値]～[終了値]-1の整数を生成
　例) range(0,5) ⇒ 0, 1, 2, 3, 4
　　range(-1,3) ⇒ -1, 0, 1, 2
　　range(8,4) ⇒ null

③ range(開始値,終了値,増分):
　[開始値]～[終了値]の手前まで
　[増分]ずつとびとびに整数を生成
　例) range(0,10,2) ⇒ 0, 2, 4, 6, 8
　　range(4,-4,-2) ⇒ 4, 2, 0,-2
　　range(10,2,2) ⇒ null

★ `for i in リスト名[要素1,要素2,･･･]:`

1) リスト要素を最初から順に取り出す
2) 取り出した要素を変数に代入
3) 繰り返し処理を実行

例)
```
for i in range(5):
    print(i)
for i in range(4,8):
    print(i)
for i in range(10,2,-2):
    print(i)
lst = ["子","丑","寅","卯"]
for i in lst:
    print(i)
```

２.条件による反復

※条件が満たされている間は繰り返す

```
while [条件]:
    [反復する処理] ※ブロック
```

※1.無限ループに陥らないこと
　実行中の中断 → Ctrl+C
※2.break で強制的に脱出できる

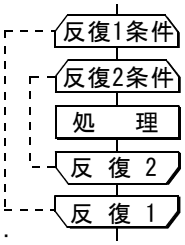
例)
```
n = 0
while n <= 2:
    n = n + 1
print(n)
```

[NOTE] 脱出時の変数の値を意識する

３.入れ子(ネスト)

※制御命令の中に同じ制御命令を入れる

例)
```
for n in range(10):
    for m in range(10):
        print(n, m, n * m)
```



【B18】

①～④の全てで「0,1,2」の結果を得た。

```
for i in range(□):          #①
    print(i)
for i in range(□,□):        #②
    print(i-5)
for i in range(-1,5,□):     #③
    print((i+1)/2)
for i in [□]:               #④
    print(4-i)
```

□に適当な数字･数列を入れなさい。

[答]

〖類題〗①~④で発生する数･文字を答えよ。

① for i in range(4):
② for i in range(-3,-1):
③ for i in range(-2,6,3):
④ for i in ["東京","京都","大阪"]:

【B19】
要素が１桁の数字であるリストで，要素の総和，最大，最小を求めたい。
□内に適当なコードを入れよ。

```
List = [4,7,2,6,9,7,5]
sum = ①              #総和の初期値
max = ②              #最大の初期値
min = ③              #最小の初期値
for i in List:
    sum = ④
    if ⑤ :
        max = ⑥
    if ⑦ :
        min = ⑧
print("総和：",sum)
print("最大：",max)
print("最小：",min)
```

※２～４行目で変数の初期値を設定している。それぞれどのような設定値が適当かを考察すること。（通常は，想定しうる最小または最大の値とする）
④は和を求める常套手段，⑤～⑧では，条件によって変数maxとminの値を変更していく。

[答]

〖類題〗１桁の数字を偶数と奇数に分類し，列挙するプログラムである。
□内に適当なコードを入れよ。

```
odd = ①              #奇数列の初期値
even = ②             #偶数列の初期値
for i in range(10):
    if ③ :
        even = even + str(i) + " "
    else:
        ④
print("偶数：",even)
print("奇数：",odd)
≫ 偶数： 0 2 4 6 8
  奇数： 1 3 5 7 9
```

※文字列の初期値だから，""を付加して設定する必要がある。

【B20】
自然数を入力し，その数の階乗を計算するプログラムを作成した。
□内に適当なコードを入れよ。

```
n = int(input("自然数："))
fact = ①             #累乗の初期値
i = ②                #乗数の初期値
while ③ :
    fact = fact * i
    i = i + 1
print(n,"! =",fact)
≫ 正の整数：4
  4 ! = 24           #例:「4」と入力
```

※Whileで評価する値をWhile内で変化させている。
While文を抜けたとき，iの値は「n+1」である。

[答]

〖類題〗【B12】と等価なコードを作成せよ。

```
n = int(input("自然数："))
fact = ①             #累乗の初期値
i = ②                #乗数の初期値
while ③ :
    fact = fact * i
    i = i - 1        #乗数iが減少
print(n,"! =",fact)
```

【B21】
1!,2!,･･･(入力した自然数)！と表示したい。□内に適当なコードを入れよ。

```
n = int(input("自然数："))
for m in range(①,②):
    fact = m
    for k in range(③,0,④):
        fact = fact * k
    print(m,"! = ",fact)
    ≫ 自然数：3
       1 ! =  1
       2 ! =  2
       3 ! =  6
```

[答]
※ プリグラムの終了時点で，m = n，k = 1 でなる。

Ａ７.関数

１.関数の構造

★「関数」とは，まとめて一定の処理を行う仕組みのこと。

・組み込み関数⇒システムに最初から用意されている関数

例)print(),input(),int(),range()
len(リスト･文字列)：要素の個数

・ユーザ定義関数⇒自分で作成する関数

★関数の定義

```
def 関数名(引数 ※1):
    [処理]※1          ※ブロック
    return [戻り値]※2
```

※1 カンマ「,」区切りで複数の引数を取る場合もある。必要なければ空欄となる。

※2 必要なければこの行は不要である。

・関数は呼び出す前に定義しておく。

★関数の呼び出し

```
関数名(引数 ※1)
```

・この命令文で戻り値が取得できる。

```
例)def circle(): #引数:無,戻り値:無
       print("円")
   def area(r):  #引数:有,戻り値:有
       s = r * r * 3.14
       return s
  circle()
  print(area(5))
  ≫ 円
    78.5
```

２.モジュールのインポート

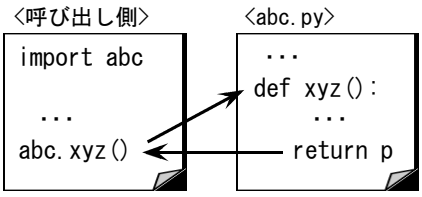
★「モジュール」とは，様々な関数が定義されているPythonプログラムのこと。
他で作られた関数を取り込み利用可能。

★モジュールの取り込み

```
import モジュール名
```

★取り込んだモジュールの利用方法

```
モジュール名　関数名(引数)
```



[研究]・from モジュール名 import 関数名
・import モジュール名 as 別名

```
例)import random
   import math
   print(random.random())
   print(random.choice(["甲","乙","丙"]))
   print(math.pi)
   print(math.sqrt(2.0))
   print(math.sin(math.pi/4))
   ≫0.5905371959216885  #0以上1未満の乱数
     丙                 #無作為抽出
     3.141592653589793  #円周率(定数)
     1.4142135623730951 #√
     0.7071067811865476 #正弦
```

【B22】

和と差を行う関数を作った。
実行結果を求めなさい。

```
def wa(x, y):
    ans = x + y
    return ans
def sa(x, y):
    ans = x - y
    return ans
print(wa(wa(-2,3),sa(-4,-1)))
```

［答］

〖類題〗 □ を埋めて，実行結果を示せ。

```
def seki( ① ):
    ans = x * y
    return ②
③ sho(x, y):
    if y == 0:
        ans = 888888888888888  #Error
    else:
        ans = x / y
    ④
print(seki(-4,sho(6, -3)))
```

【B23】
階乗 n! を求める関数 fact(n) 及び順列 nPr [=n!/(n-r)!] を求める関数 P(n,r) を作成し，確認計算も実施した。□を埋めなさい。

```
        ①        :
    val = ②
    if n ③ 0 : #「0」以外のとき
        for k in range(n):
            val = ④
    ⑤
        ⑥        :
    val = fact(n)/fact(n-r)
    ⑦
print(P(7,3),P(4,4))
```

[答]

〖類題〗
❶【B23】のプログラムに追加して，組合せ nCr [=n!/((n-r)!*r!)] を求める関数 C(n,r) を作成しなさい。
❷ リスト data の要素の値を「*」の棒グラフで表すプログラムを作成した。□を埋めなさい。

```
        ①        :
    hight = ②
    for i in range(x):
        hight = ③
    print(hight)
data=[2,5,8,1]
for i in data:
    bar(i)
```

≫ **
*****
****
*
※関数bar(x)には戻り値がない。

【B24】
A,B,C の3文字を無作為に続けて並べ，合計100文字表示する。それぞれ30%，50%，20% の割合で出現するようにしたい。□を埋めなさい。

```
import random
n=10
letter=""
prob=[0.7, 0.2, 0.0]
data=["A", "B", "C"]
for i in range( ① ):
   x=random.random()
   for j in range( ② ):
      if x >= ③ :
         letter=letter+ ④
         break
print(letter)
```

※ n回繰り返す中で，各回とも初めて70%以上，80%以上，100%以上の割合に達したとき，それぞれ"A","B","C"の文字を追加していく。
※ 例えば「BABCBBABAC」のように表示される。
[答]
〖類題〗
❶ サイコロを続けて振ったとき，初めて６の目が出るまで何回サイコロを振ったかを表示したい。□を埋めよ。

```
import random
n = ①
dice = [1, 2, 3, 4, 5, 6]
while random.choice(dice) ② 6:
        ③
print(n)
```

❷ 変数x,y,zはそれぞれ何を意味するか。

```
from random import random
x = random() * 5
y = random() * 5 + 1
z = int(random() * 5 + 1 )
print(x, y, z)
```